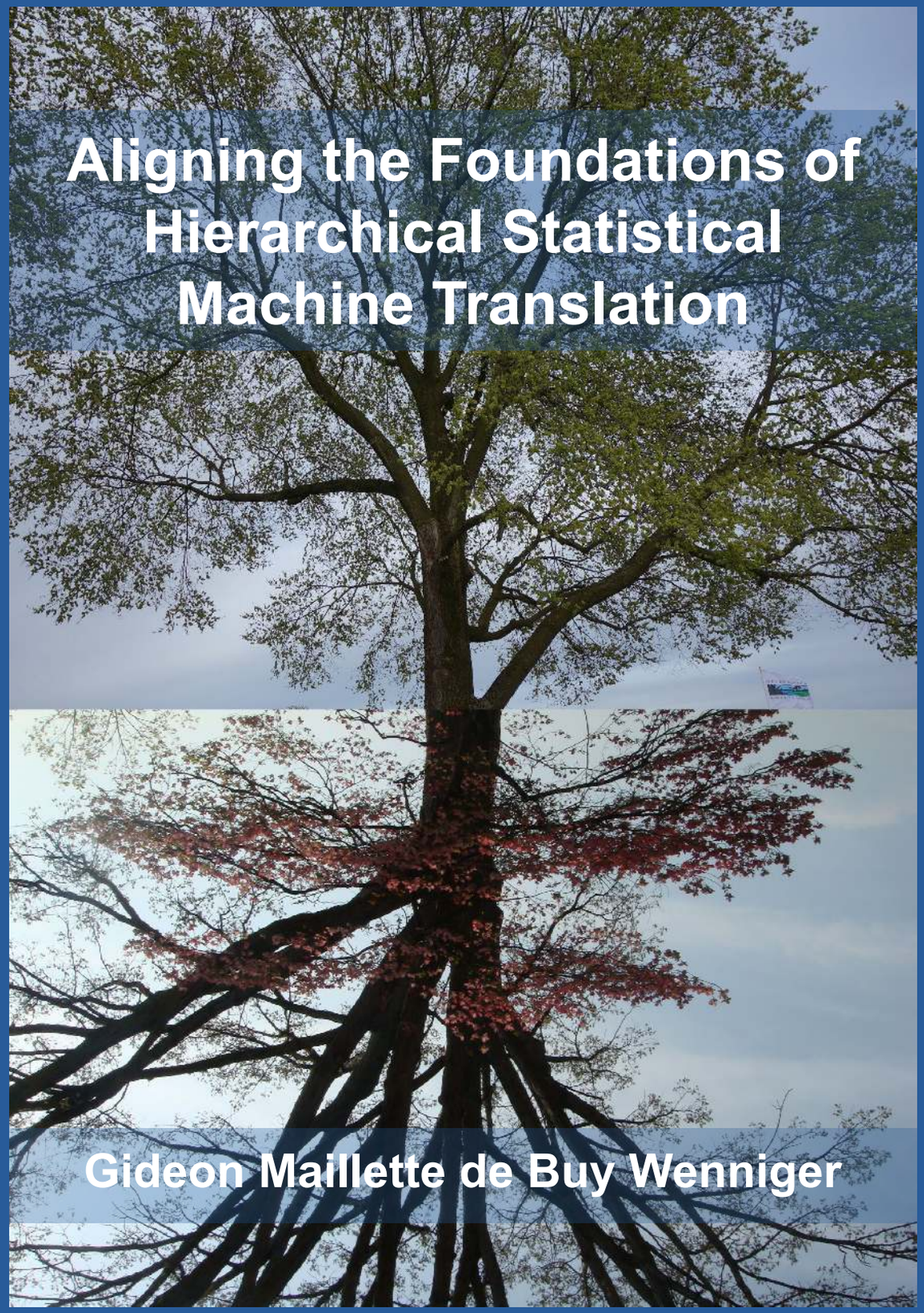
INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Aligning the Foundations of Hierarchical Statistical Machine Translation

Gideon Maillette de Buy Wenniger

Aligning the Foundations of Hierarchical Statistical Machine Translation



Gideon Maillette de Buy Wenniger

Aligning the Foundations of Hierarchical Statistical Machine Translation

Gideon Maillette de Buy Wenniger

Aligning the Foundations of Hierarchical Statistical Machine Translation

ILLC Dissertation Series DS-200X-NN

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

The investigations were supported by The Netherlands Organization for Scientific Research (NWO) under grant nr. 612.066.929 and VICI grant nr. 277-89-002 and Stichting voor de Technische Wetenschappen (STW) grant nr. 12271.

Cover design by Ranjita Bose.
Printed and bound by Ipskamp Drukkers.

ISBN: 978-94-028-0193-4

Aligning the Foundations of Hierarchical Statistical Machine Translation

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. D.C. van den Boom
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in de Aula der Universiteit
op vrijdag 10 juni 2016, te 11.00 uur

door

Gideon Emile Maillette de Buij Wenniger

geboren te Amsterdam

Promotiecommissie:

Promotor:	Prof. dr. K. Sima'an	Universiteit van Amsterdam
Overige leden:	Prof. dr. D. Wu	Hong Kong University of Science and Technology
	Prof. dr. J. van Genabith	Universität des Saarlandes
	Prof. dr. L.W.M. Bod	Universiteit van Amsterdam
	Dr. C. Monz	Universiteit van Amsterdam
	Dr. I.A. Titov	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Contents

Acknowledgments

As I am at the end of the Ph.D. and thesis writing phase, I owe heartfelt thanks to a number of people without whom this work would not have been possible.

I first met Khalil as a Masters student in his course on Natural Language Processing. Since my interview with his group, I was excited to work on statistical machine translation and more importantly with the wonderful and knowledgeable members of his group. I am immensely grateful for the opportunity to embark on this long journey together on the research of hierarchical translation, reordering and hierarchical translation equivalence. Khalil's continued support and energy, the very personal contact, his dedication to the research, to making it work together patiently, persistently, they have made this thesis possible. I learned a lot from Khalil's unique ability to propose groundbreaking ideas that go beyond and sometimes against the established wisdom, and his confidence in pursing them over many years until after inevitable initial failures big successes are obtained.

I would like to thank my current and former colleagues and friends at ILLC for all their support and time, and all the fun we had together: Sophie, Miloš, Wilker, Phong, Raquel, Joachim, Philip, Joost, Hoang, Aaron, Stella, Bushra, Amir, Desmond, Andreas, Federico, Gideon Borensztajn, Dieuwke, Tejaswini, Markos, Maxim, Bart, Christos, Reut, Diego, Ehsan, Carlos, Umberto, Inés, Facundo, Sumit, Nina, Davide, Lena, Raúl.

I deeply appreciate the support of and wonderful interaction with all the current and former members of our LACO group at ILLC, in particular Ivan Titov, Remko Scha, Rens Bod, Raquel Fernández and Henk Zeevat. I am also grateful to Adam Lopez, Dekai Wu, David Chiang, Josef van Genabith, Andy Way, Christof Monz, Anders Søgaard and Kristina Toutanova for their interest in my research and our stimulating conversations and their time for answering my questions about specific details of papers.

The interaction I had with Andreas Zollmann about labeling *Hiero* has been of great importance to the work in this thesis. Many thanks to Matt Post for working together on *Joshua* and discussing technical and theoretical issues. The project we did at the MT Marathon project gave me the necessary insights in the core Joshua decoder code, to implement soft matching in Joshua, a crucial component in my eventual success. Thanks to Jonathan Weese, Juri Ganitkevitch and Yuan Cao from John Hopkins for answering many of my technical questions about different Joshua components. I want to thank Alexander Fraser for organizing the Dagstuhl Seminar on translating to morphologically rich languages rich languages and inviting me there, and spending time to discuss ideas at the MT Marathon. To Ondřej Bojar and Loïc Barrault I am grateful for their help and support, particularly around MT marthon 2010 and 2013. Miriam Kaeshammer and Matthias Huck have both inspired me in their own ways with their work, which is very relevant to the work in my thesis, and I am grateful for the contact we had by mail and in person during conferences on several occasions.

I want to thank all members of faculty and staff of the ILLC who have created

an outstanding, excellent and very supportive research environment. I'd like to give special thanks to Jenny Batson, Tanja Kassenaar, Peter van Ormondt, Karin Gigengack, Leen Torenvliet and Yde Venema. I am immensely grateful for all the help and support you have given me over so many years and in so many different ways.

This work might not have been possible without the prelude of my thesis work in the LASA group at EPFL Lausanne. Thanks go to my supervisors and friends at EPFL: Aude Billard, Basilio Noris and Sylvain Calinon. Special thanks to Aude Billard for welcoming me as a foreign student in the LASA group, and to Basilio for being an inspiring direct supervisor and good friend in Switzerland. I also want to thank Theo Gevers and Nicu Sebe for being my supervisors at UvA for the Masters. Arnoud Visser has been my mentor around the time of finishing my Master and together we wrote my first academic paper, I am thankful for that.

I deeply appreciate the dedication of my former teachers at the Bachelor and Master AI have instilled in me the passion for research in Artificial Intelligence, and the solid foundation that allowed me to pursue my PhD research. I particularly want to thank Maarten van Someren for being my thesis supervisor during the Bachelor, and Leo Dorst for helping to make my exchange to ETH Zürich possible.

I also would like to thank my previous supervisor Sven Behnke and former colleagues from my time as a researcher in Bonn. The period in Bonn taught me a lot about doing research and working in a bigger team, and these lessons helped me make my PhD a success.

I would like to thank my former colleagues and friends from the Bachelor and Master AI for cooperation, fun and companionship: Attila, Barbara, Andrea, Harpreet, Nikolaj, Arvid, Chris, Remi, Maarten, Jacco, Bernardo and Bas. Attila in particular I want to thank for being my main partner in AI, friend and flatmate for several years.

I want to thank my friends: Daan, Jelle, Bas, Thomas, Joris, Sankh, Katya, Mireia, Roos. You have endured my wild tree chase over many years, and always made time to do fun things together, be it a simple dinner, a barbecue in the park, hiking, skiing, or just talking over a few beers. It helped share my PhD journey and made the difficult phases much easier.

I want to thank my many friends at toastmasters Laverne, Jennifer, Paolo, Gillian, Joao, Pierre, Chinan, Riaan, Bill Monsour, Richard, Michal, Yoko, Giacomo, Samrin, Philip, Kees, Perle, Annemijn and others for being a constant source of inspiration and learning apart from academia over many years. I want to thank my friends and mentors over many years Donald and Irénke. I also want to thank my former French teacher Arlette Peintaux. I am indebted to my teachers at the Montessori lyceum, in particular Cor Mak and Huib Schwab and my teachers from the Amsterdamse Montessori primary school Wout Geel and Coby Jelsma, as well as to my nanny Tineke Jansen. They always encouraged my curiosity and made me the person I am.

I want to thank my parents in-law and Kalyan, Sunanda, my brother-in-law Udayan, aunt-in-law Manjula and grandmother-in-law Pratima for being the most welcoming in law family. My large family has always supported and welcomed me, including my more distant family in US, and this has made a big difference.

I am deeply grateful to my parents, Liesbeth and Willem, for always supporting me and encouraging me to persist in pursuing my dreams, academic and other. Your example of how to live a balanced and fulfilling life and stay in close contact with a large variety of people you love is admirable. My brother Lucas has always been a great support and example for me, both allowing me to benefit from his experience and wisdom to do tested things much easier but also freeing me to explore different paths. His partner Laura is always so supportive to me and Ranjita and with my nieces Rosa and Anne life is always full of excitement. I am grateful to my parents' partners Cees and Annelieke for giving so much joy to my parents and the entire family, and for having made me and Ranjita always feel so welcome in your lives from the very beginning. Your advise and wisdom is very much appreciated. Finally I want to thank my wife Ranjita, who is the flower in my life and partner in everything. Her love, support and wit has given me so much joy and her sharp mind, courage and confidence are something I learn from every day.

1

Introduction

> *Thus, the central theme that runs through my remarks is that complexity frequently takes the form of hierarchy, and that hierarchic systems have some common properties that are independent of their specific content. Hierarchy, I shall argue, is one of the central structural schemes that the architect of complexity uses.*
>
> – Herbert A. Simon, *The Architecture of Complexity*

The automatic translation of sentences and texts relies heavily on the reuse and composition of translations for shorter, previously observed patterns.[1] Automatic translation systems are typically trained on large collections of translated sentence pairs, so-called parallel corpora. When such parallel corpora are used, two important questions arise:

1. What atomic translation equivalence units (TEUs) follow from the translated sentence pairs?

2. How are TEUs composed together to form longer TEUs up to the sentence level?

Inducing translation equivalence units The first task associated with the first question consists in establishing translation equivalence at the word level given translation equivalence at the sentence level. This task is known as *word alignment* and its solutions form an important part of the foundations of modern statistical machine translation (SMT) (Brown et al., 1988). But the first question goes further. Because once translation equivalence at the word level is established, it is desirable to use it to form larger translation fragments and use those larger fragments directly for the

[1] In fact such patterns play not only an important role in fully automated translation but also in human translation in the form of translation memories. Such translation memories are databases that store "segments" that have been previously translated, and can be re-used to bootstrap faster human translation of new text.

translation of new sentences. As it turns out, under the restriction of working with contiguous TEUs, there is a finite and intuitive set of larger TEUs that can be composed from the translation equivalence relations at the word level. These TEUs are known as *phrase pairs* and form the core of modern phrase-based SMT (Koehn et al., 2003; Zens et al., 2002). Going one step beyond contiguous TEUs, subsequences of phrase pairs that are themselves phrase pairs may be replaced by variables, yielding *hierarchical phrase pairs*. These hierarchical phrase pairs are the basis of hierarchical SMT, which is the focus of this thesis.

Reusing and composing translation equivalence units The second question mentioned above can be reformulated as "how to compose TEUs extracted from a parallel corpus?" and forms a major challenge in building a machine translation model. Composing TEUs into new TEUs is both a syntactic and a semantic challenge, because the new TEUs are expected to be semantically coherent as well as syntactically acceptable. Even so, the large majority of models, apart from a few recent *deep neural network* models (Bengio et al., 2003; Schwenk et al., 2006; Mikolov et al., 2010; Auli et al., 2013; Kalchbrenner and Blunsom, 2013)[2] , focus on improving word order and lexical choice without a view into semantics. The popular *phrase-based* models use algebraic composition operations based on simple concatenation, possibly extended by restricted local reordering of elements. Alternatively, *hierarchical* models base reordering on grammar formalisms that use recursive decomposition to deal with long-range order differences between languages. The work in this thesis is concerned with the latter kind of composition as explained in more detail later in this chapter.

Concatenating two or more adjacent TEUs to form new TEUs has been used to implement composition in both the original IBM models as well as more recent phrase-based models. Often, the option to permute the positions of the involved target side TEUs (within a limited distance from each other), is added to this basic strategy. Even with this addition, the resulting algebraic operation of composition remains impoverished. This is compensated for by using dedicated components that score the possible compositions such as the ordering model, translation model and language model, as shown by the following formula that is optimized during translation:

$$\arg\max_{\mathbf{e}} P(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \frac{P(\mathbf{f}|\mathbf{e}) \cdot P(\mathbf{e})}{P(\mathbf{f})} = \arg\max_{\mathbf{e}} P_O(\mathbf{f}|\mathbf{e}) \cdot P_{TM}(\mathbf{f}|\mathbf{e}) \cdot P_{LM}(\mathbf{e}) \quad (1.1)$$

Where the target sentence $\mathbf{e}$ is translated from the source sentence $\mathbf{f}$. This formula is derived from the noisy channel model (Shannon, 1948), performing Bayesian inversion of optimization for the most likely source sentence first, and then splitting into an ordering model $P_O(\mathbf{f}|\mathbf{e})$ (order of $\mathbf{f}$ given order of $\mathbf{e}$), a lexical translation model

[2]This is only a small selection of relevant work. Furthermore it should be noted that Deep Neural Network models deal with semantics only in an implicit way, by creating distributed representations of input-output mappings. Whether these representations succeed in adequately dealing with composition, as typically required by theories of semantics, is not clear.

$P_{TM}(\mathbf{f}|\mathbf{e})$ and a language model $P_{LM}(\mathbf{e})$. Unfortunately, the impoverished algebraic operations have a high price, because the space of possible compositions is quite large while it still misses long-range order differences since the operations are local. The hierarchical models, inspired by syntactic grammars for parsing, aim to improve the algebraic operations. They do so by embedding them within a formal device (synchronous grammar) and defining scoring functions based on component models similar to those used by phrase-based models.

One of the earliest attempts for grammar based SMT are possibly the use of inversion transduction grammars (ITGs) (Wu, 1997) . ITG is a grammar formalism where TEUs are lexicon entries (synchronous source-target lexical rules) that are combined using binary grammar rules such as $X \rightarrow X_1 X_2$. These grammar rules can implement either the straight order of composition (also known as monotone) or the inverted order. In the straight order the nonterminals X_1 and X_2 are in the same order on the source and target side, whereas in the inverted order the target side order $X_2 X_1$ inverts the source order $X_1 X_2$.

A major issue in the use of grammars is how to label the nonterminals of the grammars. Since labels are usually seen as refinements of the composition operations allowed by the grammar, this is an important issue. Thus, a grammar can be better fitted to the data by using adequate labels. Labeling grammars, is at the core of the problem addressed in this thesis.

The hierarchical phrase-based model (Hiero) (Chiang, 2005) is a successful variant of ITG and is used as the base model in this thesis. Next, we will discuss Hiero in detail. "How to label Hiero using syntactic or semantic labels given monolingual parsing tools?", is the challenge addressed in the existing literature. "How to label Hiero without such monolingual resources and extract labels directly from the parallel corpus?", is the challenge addressed in this thesis. We will discuss our approach directly after we introduce Hiero and the problem statement in more detail.

1.1 Hiero grammars and syntactic composition

Hiero grammars (Chiang, 2005) are a particular form of synchronous context-free grammars (SCFGs), which were originally known as *syntax directed transduction grammars* (Lewis and Stearns, 1968) or *syntax-directed translation schemata* (Aho and Ullman, 1969). These grammars use only one type of non-terminal: X.[3] Hiero grammar rules are lexicalized synchronous grammar rules, with at least one word on the source and target side of the rule and up to two nonterminals. Hiero rules, also known as hierarchical phrase pairs, generalize normal phrase pairs by replacing embedded, smaller phrase pairs with nonterminals. For example take the phrase pairs "⟨we love hats, nous aimons les chapeaux⟩" and "⟨except ugly hats, sauf les chapeaux

[3]More precisely, in addition to X and a *START* symbol, one more non-terminal S is used. But the use of this second non-terminal is restricted to a small set of *glue rules* used to monotonically combine incomplete derivations into complete derivations.

$X \rightarrow \langle X^{\boxed{1}} \text{ love } X^{\boxed{2}}, X^{\boxed{1}} \text{ aimons } X^{\boxed{2}} \rangle$ (R101)

$X \rightarrow \langle \text{how many } X^{\boxed{1}} \text{ clients } X^{\boxed{2}}, \text{combien } X^{\boxed{2}} \text{ de clients } X^{\boxed{1}} \rangle$ (R102)

$X \rightarrow \langle \text{we love } X^{\boxed{1}}, \text{nous aimons } X^{\boxed{1}} \rangle$ (R103)

$X \rightarrow \langle X^{\boxed{1}} \text{ hats, les chapeaux } X^{\boxed{1}} \rangle$ (R104)

$X \rightarrow \langle X^{\boxed{1}} \text{ hats}, X^{\boxed{1}} \text{ les chapeaux} \rangle$ (R105)

$X \rightarrow \langle \text{except } X^{\boxed{1}}, \text{sauf } X^{\boxed{1}} \rangle$ (R106)

$X \rightarrow \langle \text{ugly, laids} \rangle$ (R107)

$X \rightarrow \langle \text{particularly } X^{\boxed{1}}, \text{en particulier } X^{\boxed{1}} \rangle$ (R108)

$X \rightarrow \langle \text{scientific, scientifiques} \rangle$ (R109)

$X \rightarrow \langle X^{\boxed{1}} \text{ publications, les publications } X^{\boxed{1}} \rangle$ (R110)

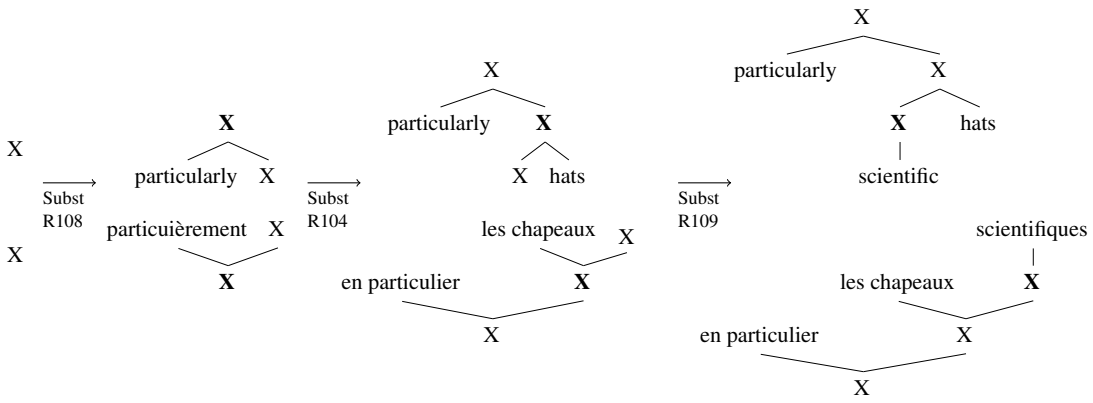
Figure 1.1: Example hierarchical phrase-based translation, as first proposed by (Chiang, 2005) (Hiero) rules for English–French translation.

laids⟩". Some Hiero rules, partly based on generalizing these two example phrase pairs, are shown in Figure 1.1. The nonterminals of rules with two gaps can be in monotone (an example is rule R101) or inverted orientation (an example is rule R102).[4] This makes Hiero grammars effectively a special, lexicalized form of ITGs (Wu, 1997). In figure 1.2 we show two Hiero derivations yielding translations for the English sentence "particularly scientific hats". Both these derivations can be formed using the rules in Figure 1.1. The first derivation yields the translation "en particulier les chapeaux scientifiques" while the second derivation yields the translation "en particulier scientifiques les chapeaux". Only the first translation is correct. This illustrates how substitution under Hiero, which without labels lacks context, can yield wrong word order and wrong translations. This motivates the syntactic enrichment of grammars, discussed next.

1.1.1 Does syntactic composition improve over Hiero?

Plain phrase-based systems (Koehn et al., 2003; Zens et al., 2002) and hierarchical phrase-based systems (Chiang, 2005) are unable to directly encourage global coherence during the composition of rules. This has been known by the research community for a long time. It has been one of the driving factors behind research on syntactic

[4] In fact English-French translation has mostly local reordering compared to other language pairs. This makes it somewhat difficult to find good examples of sentences containing non-contiguous inverted words, yielding inverted Hiero rules. One example of a sentence pair from which this particular inverted rule can be derived is "⟨how many [young]$_1$ clients [do they have]$_2$?, combien [ont-ils]$_2$ de clients [jeunes]$_1$?⟩".



(a) Derivation yielding correct translation.

(b) Derivation yielding wrong translation.

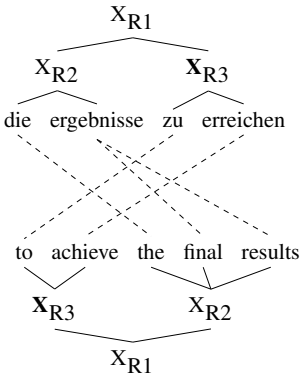
Figure 1.2: HIERO example derivations yielding different French translations for the English input sentence "particularly scientific hats". At every derivation step, we indicate the substituted (Subst) rule number, and mark the root node of the substituted synchronous tree pair with bold.

translation. Syntactic translation systems (Yamada and Knight, 2001; Hassan et al., 2006; Zollmann and Venugopal, 2006; Galley et al., 2004; Liu et al., 2006; Almaghout et al., 2010) use syntax on the source and/or target side, typically enforcing it as a hard constraint, turning translation into a task that has much similarity with classical monolingual parsing. In this view the source sentence is parsed, producing the translation (target side) as a byproduct of this parsing task.[5] Unfortunately syntactic translation approaches that apply syntax as a hard constraint often suffer loss of coverage, as valid translations may be blocked by syntactic constraints. Syntactic translation approaches have historically had problems improving over phrase-based and hierarchical phrase-based systems (Koehn et al., 2003; Almaghout et al., 2011).[6] Translation coverage loss by syntactic systems can be considered one of the main reasons for this. In contrast to syntactic systems, phrase-based and hierarchical phrase-based systems do not enforce any formal syntactic constraints, and therefore do not suffer the same problems related to coverage loss. These challenges have lead to the development of methods to introduce syntax to translation systems in a softer, less obstructive way such as (Marcu et al., 2006; Hassan et al., 2007). Here we focus on approaches that extend Hiero with syntax. A well known example of this is a relaxed, heuristic syntactic labeling approach called syntax-augmented machine translation (SAMT) (Zollmann and Venugopal, 2006). This labeling approach assures that all phrases can be labeled and therefore no TEUs need to be discarded because of not matching proper syntactic categories. But in parallel these challenges also yielded approaches that treat syntactic or other labels as soft constraints, whose satisfaction is preferred but not required (Marton and Resnik, 2008; Chiang et al., 2008; Cherry, 2008; Venugopal et al., 2009; Chiang, 2010). These two approaches, in particular when combined, yielded significant improvements over phrase-based models and unlabeled hierarchical translation (Venugopal et al., 2009; Chiang, 2010) for certain language pairs.

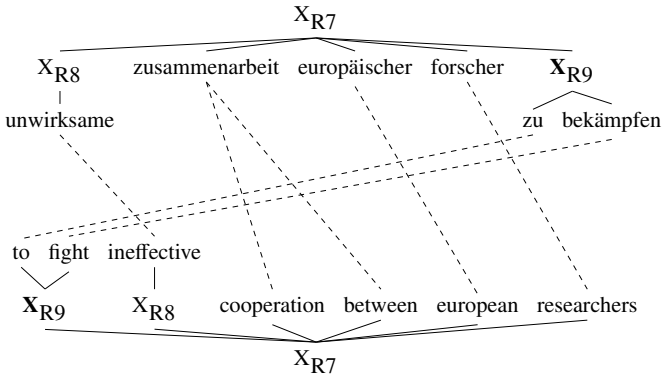
Syntax provides one viable way to improve composition of Hiero. But reliable syntactic parsers may not be available, and syntax may be incompatible with the alignment structure. We will now show another example of Hiero failing to perform coherent composition, and explain how labels based on the alignment structure can solve this.

[5] One important difference of (hierarchical) translation with monolingual parsing is the fact that the presence of the language model which necessitates approximation in the form of beam search with cube pruning.

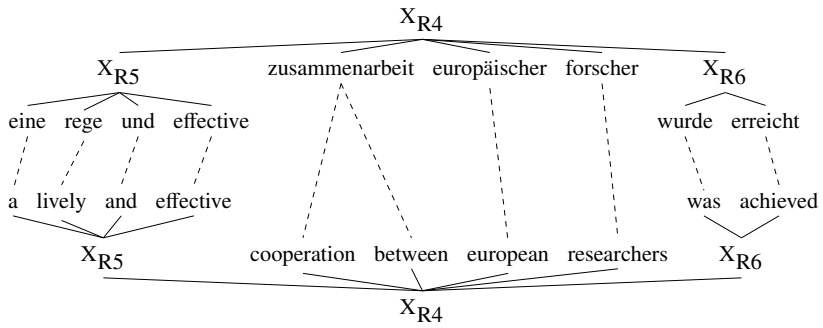
[6] Many publications concerning syntactic systems, including recent ones, unfortunately fail to report a direct comparison against Hiero. Hence, we think that the here reported cases where Hiero outperforms syntactic systems or performs equally to them should be considered only a conservative lower-bound on the actual number of cases where Hiero performs equal or better.



(a) (Left-binding) Inverted training example 1.



(b) (Left-binding) Inverted training example 2.



(c) Monotone training example.

Figure 1.3: Training examples, the labeled and indexed nodes represent (some of the) phrase pairs that can be extracted from the aligned sentence pairs.

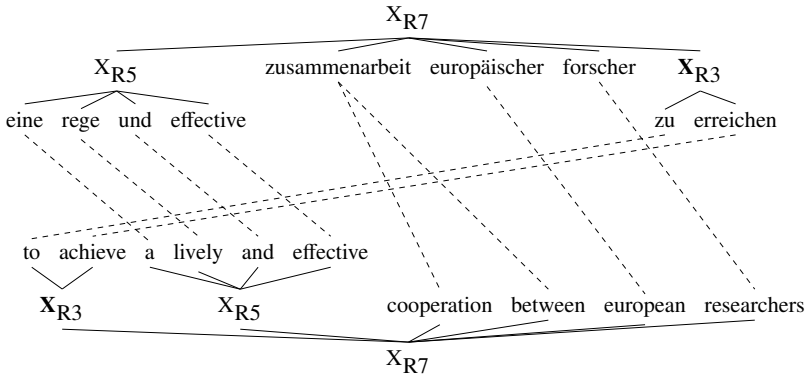
1.2 Improving composition in Hiero with reordering labels

Figure 1.3 shows a toy training parallel corpus of three word-aligned sentence pairs, decomposed into Hiero rules (hierarchical phrase pairs); the boxed *Ri* indices at the nodes stand for rule identities placed on the left-hand side of every rule. For example, in Figure 1.3c we find rule *R*5

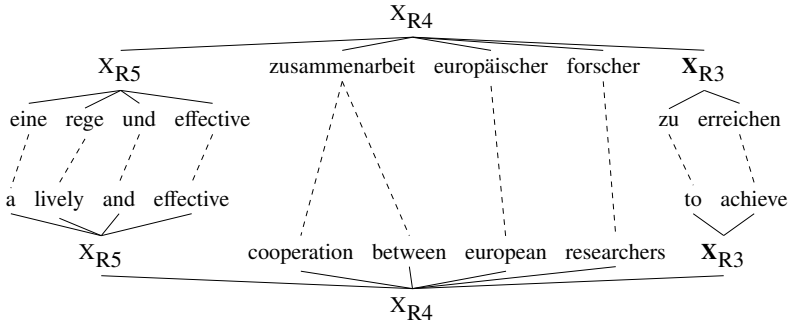
$$X \rightarrow \langle \text{eine rege und effective}, \text{a lively and effective} \rangle$$

and in Figure 1.3a, rule *R*3

$$X \rightarrow \langle \text{zu erreichen}, \text{to achieve} \rangle$$



(a) Correct translation for new sentence that produces the right word order.



(b) Wrong alternative translation that can be produced by HIERO.

Figure 1.4: Translations of the new sentence “eine rege und effektive zusammenarbeit europäischer forscher zu erreichen”.

By cutting out some of the embedded phrase pairs, we obtain HIERO rules with gaps. As an example, from the phrase pair at the root of the aligned sentence pair in Figure 1.3c, the hierarchical rule $R4$

$$X \rightarrow \langle \mathrm{X}_{\boxed{1}} \text{ zusammenarbeit europäischer forscher } X_{\boxed{2}}, \\ X_{\boxed{1}} \text{ cooperation between european researchers } X_{\boxed{2}} \rangle$$

can be extracted by cutting out the two embedded phrase pairs $R5$ and $R6$ as gaps labeled X. Similarly, we obtain rule $R7$

$$X \rightarrow \langle \mathrm{X}_{\boxed{1}} \text{ zusammenarbeit europäischer forscher } X_{\boxed{2}}, \\ X_{\boxed{2}} \; X_{\boxed{1}} \text{ cooperation between european researchers } \rangle$$

from the root phrase pair in Figure 1.3b. Note how the training examples for the English verbs “to fight" in Figure 1.3b and “was achieved" in Figure 1.3c are embedded

within, respectively, monotone and inverted reordering patterns when translated into German.

We now exemplify how HIERO risks missing the correct word order and how labels from the surrounding word alignment context may help. In Figure 1.4a, translation rule $R7$ is combined with rule $R5$ and rule $R3$ to translate the new sentence "eine rege und effektive zusammenarbeit europäischer forscher zu erreichen". Here starting from translation rule $R7$ and then substituting $R5$ and $R3$ on the two X nonterminals, the correct word order can be obtained. However, the rules extracted during training also permit a different translation of this sentence that produces the wrong word order, shown in Figure 1.4b. This translation is formed by combining $R4$ with $R5$ and $R3$. Both Hiero derivations are eligible, and the independence assumptions between rules suggest that there is no reason why Hiero's synchronous grammar should be able to select the correct word order. The independence assumptions between the rules suggest also that the burden of selecting the correct reordering is left over to the target language model.

How could the use of word alignment context help produce preference for correct reordering in HIERO? Contrast the reordering structures in Figure 1.3a and Figure 1.3b to the structure in Figure 1.3c. In the first two the verb units, "to achieve" and "to fight" (labeled with a bold **X**), are inverted with respect to the embedding context, whereas in the latter example, the verb "was achieved" is monotone with respect to the embedding context. In this simple example, two types of verbs can be discriminated using word alignment types from the embedding rules, which can be used as HIERO labels. Such labeling can be obtained during HIERO rule extraction from the word-aligned training sentence pairs without need for other resources. By extracting such *reordering labels*, the incorrect substitution in Figure 1.4b could be either prevented or made far less likely than the correct alternative.[7] Phrases induce a certain reordering pattern with respect to their sub-phrases and with respect to the parent phrase that embeds them. We note that in a sentence-aligned, word-aligned parallel corpus, it turns out, there are many more reordering patterns than the binary choice of monotone/inverted.

1.3 What if syntactic composition is not available or effective?

Unlabeled HIERO does not describe context for its rule composition. Syntactic SMT has tried to overcome this problem by adding syntactic information to HIERO. While strictly enforced syntactic labels often give problems, using syntax as a soft constraint works quite well. However, in the early stages of this thesis research, we explored

[7] One might wonder about the frequency of verbs that show such preferences for reordering: in the filtered test grammar (see experimental section) there are more than 27,000 phrase pairs, each with 2 words on both sides, that show such a preference for inversion relative to their embedding context. A large fraction of these phrase pairs corresponds to such verbal constructs. This itself is just a part of one of many types of reordering phenomena, selected for this example.

the frequent incompatibility of syntax and alignment structure. We published a paper about automatic ways to identify and visualize this incompatibility (Maillette de Buy Wenniger and Sima'an, 2014b). Some existing work tried to adapt the syntax to make it more compatible with the word alignment structure and vice versa (Gildea, 2003; Wang et al., 2010). Generally, the frequent incompatibility of syntax and alignment structure is dealt with by adapting the syntax and word alignments, and using the syntactic information only as soft constraints in translation.

We now summarize the situation sketched so far:

- The composition of rules in HIERO is weak, due to the lack of labels.
- Syntax is often used to enrich hierarchical translation grammars, promoting more coherent composition of rules into translations.

But we observe that the application of syntax leads to two clear problems:

1. Monolingual syntax and bilingual translation equivalence are not naturally compatible.
2. Reliable, high quality syntactic parsers are not available for all languages.

A third observation points us in the direction of a possible solution:

3. Hierarchical translation equivalence is induced by word alignments, and therefore by definition compatible with any extracted TEUs.

Noting these two problems, and the direction for a possible solution suggested by the third observation, for this thesis we decided to adopt a different approach by not forcing syntax to match with the alignment structure. Instead we take the hierarchical alignment structure itself as the core structure of translation and the basis for reordering. We would create a representation of the hierarchical word alignment structure called hierarchical alignment trees (HATs) (Sima'an and Maillette de Buy Wenniger, 2011a). This would provide a basis for a line of work enabling hierarchical translation with context sensitive reordering, without requiring syntax. We now take a closer look at these three observations.

The first observation is supported by much empirical evidence in the literature, beyond our own observations. For example by the fact that allowing only phrases compatible with syntax has been shown to lead to significant losses of coverage and translation quality in phrase-based translation (Koehn et al., 2003). Furthermore, the coverage problems are not restricted to just strict constituency parse syntax: Almaghout et al. (2010) for example reports that with SAMT labels 50% of all rules remain unlabeled i.e. are incompatible with syntax, and for combinatory categorial grammar (CCG) labels this is 30%. Another source of evidence comes from (Wang et al., 2010), an influential work on syntactic MT. This work shows that re-structuring (binarization) and re-labeling of syntactic trees, increase performance for string-to-tree translation. These operations are essentially concerned with adapting the syntactic

structure and syntactic information respectively, so as to increase its compatibility with the alignment structure. This therefore indirectly implies that syntax, at least in its raw form, is often incompatible with the alignment structure.

The second observation is supported by the fact that most linguistic resources like parsers and taggers have been developed for European languages as well as some popular Asian languages such as Chinese and Japanese. For many other languages little or no such resources exist or at least the training material on which the resources are based is very limited; as illustrated by a sample of publications on this topic (Garrette et al., 2013; Irvine and Callison-Burch, 2013; Scherrer and Sagot, 2013; Kim et al., 2015; Duong et al., 2015).

The third observation will be developed in chapter 4, but here we briefly sketch the argumentation: *Hierarchical translation equivalence* is defined in terms of the TEUs induced by a word alignment and their (hierarchical) subsumption relations.[8] When translation equivalence is established at the word level, and from there at the phrase level, the generated (contiguous) TEUs must be nested in a certain way. The specific, applicable nesting of these TEUs implies a specific set of subsumption relations. These subsumption relations in turn determine a hierarchical embedding structure with associated reordering relations. The embedding structure and reordering relations together with the set of TEU constitute hierarchical translation equivalence. And since hierarchical translation equivalence is derived from the extracted TEUs it is by definition also compatible with them.

1.4 Problem statement and hypothesis

Summarizing our observations from the last section, we saw that syntax is not necessarily compatible with word alignments and that high quality parsers are not always available. Furthermore, our third observation suggested that word alignments could be an important part of the solution. This leads to the following problem statement:

Problem Statement. *Is it possible to obtain sufficient information from word alignments only to facilitate coherent composition in hierarchical* SMT*, more specifically in* HIERO*?*

The third observation suggests that this may be possible, and leads to the following, more specific hypothesis:

[8]The term *subsumption* as used in this thesis requires some explanation. Generally, the term subsumption is defined as the act of considering or including something as part of a more general category. In our particular case, subsumption is mainly used for TEUs, in particular phrase pairs. Subsumption of a phrase P_s by a larger phrase P_b implies that P_s has source and target spans that are a subspan of the corresponding spans of P_b. But stronger than this, it implies that P_b can be composed from P_s and additional non-overlapping phrases and/or loose words on the source and target side.

Hypothesis. *Hierarchical translation equivalence relations induced from word alignments provide the means to facilitate more coherent composition in hierarchical* SMT. *These relations directly and precisely inform about bilingual hierarchical reordering structure, in contrast to monolingual syntax. Additionally, these relations are inherently compatible with induced* TEUs. *This allows them to be used to form effective (reordering) labels or other soft reordering constraints that are used to promote coherent composition and better reordering, with possible advantages over monolingual syntax.*

1.5 Hierarchical Alignment Trees and their applications

In this thesis we propose structures called hierarchical alignment trees (HATs) which provide a compact, complete and unambiguous representation of hierarchical translation equivalence. These structures are motivated by the desire to exactly model the composition of small TEUs into bigger ones up to the sentence level, as induced from the word alignment. In chapter 4 we discuss HATs, their relation to existing representations of translation equivalence and word order as well as examples, formal definitions and algorithms. One way to think about HATs is to follow analogy with treebanks in parsing. Treebanks in parsing allow the straightforward extraction of many types of grammars, including basic context-free grammars (CFGs) as well as CFGs enriched with parent annotation or head-word annotation, and grammars containing larger fragments (tree-substitution grammars). HATs are designed to fulfill a similar role in facilitating the straightforward extraction of many different types of reordering labeled grammars. HATs provide a compact representation of translation equivalence in the form of sets of recursive bilingual trees that exactly represent the set of contiguous TEUs induced by word alignments and their recursive reordering relations.

1.5.1 Hierarchical Alignment Trees as a basis for rule extraction and labeling

Extraction of Hiero rules is intuitively based on HATs, but does not require them. However, HAT-based rule extraction is not restricted to Hiero rules. For example, HATs also support the extraction of more complex non-binary reordering rules, such as complex non-lexicalized permutation rules. Inducing such rules without HATs is not straightforward. Such rules can have various successful applications, for example for preordering, as shown by (Stanojević and Sima'an, 2015).[9]

[9] In that work a subset of HATs called permutation trees (PETs) (Gildea et al., 2006) is used, which is restricted to alignments with bijective mappings.

In the work for this thesis we have focused on the application of HATs for adding reordering labels to HIERO rules. In the previous section we established the potential merit of reordering labels, and now we come back to the question how such labels need to be formed. HATs naturally allow extraction of hierarchical translation rules enriched with many different types of reordering labels. In particular the presence of explicit reordering labels on the nodes of HATs make this possible. Note that one could also form reordering labels without first extracting HATs. But with HATs all reordering relations are explicitly represented in one representation, making the formation of reordering labels simpler and more intuitive.

1.5.2 Analyzing empirical translation equivalence with hierarchical alignment trees

In chapter 6 we will see that HATs have another role in supporting empirical analysis of word alignment. In particular we will see how word alignments have an identical representation in the form of a mathematical framework called *set-permutations*, which generalizes permutations to include many-to-many mappings between elements. HATs allow to exactly determine what category of *set-permutations* is necessary to cover a particular word alignment. In doing so they allow to precisely answer certain questions regarding the complexity of empirical word alignments, for example what fraction of the word alignments can be covered by binarizable permutations and consequently by a popular type of grammars know as inversion transduction grammars (ITGs).

1.6 Thesis focus and contributions

We now return to the initial two questions of this chapter. Note how HATs bridge the two questions regarding the inducible set of TEUs and how they are to be reused and composed, by representing not only TEUs but also their hierarchical reordering relations. These reordering relations, in the form of labels, readily provide a reordering context to extracted rules that allows these rules to be re-composed in a way that yields more coherent word order. As such the role of reordering labels is similar to that of syntactic labels in treebanks. But the important difference between reordering labels and syntactic labels that are adopted for translation, is that while the former are by definition compatible with TEUs, the latter frequently are not since translation equivalence and monolingual syntax are not necessarily compatible. In chapter 5 we discuss the adoption of reordering labels in hierarchical translation. These labels target global coherence with respect to word order, and seek to resolve problems arising from a blind reliance on the language model far beyond its intended use or capability. We will see how these labels in combination with soft constraints significantly improve the quality of automatic translation of various language pairs.

We end this section with an overview of contributions:

Contributions

- We propose hierarchical alignment trees (HATs), bilingual trees that fully capture the hierarchical translation equivalence structure induced by word alignment, extending normalized decomposition trees (NDTs) and permutation trees (PETs). HATs are designed to serve a function that is somewhat comparable to that of treebanks in parsing. They facilitate many applications, including rule extraction, rule labeling, hierarchical alignment structure visualization and preordering. HATs were first proposed in (Sima'an and Maillette de Buy Wenniger, 2011b) and are the foundation for the work on labeling HIERO (Maillette de Buy Wenniger and Sima'an, 2013b, 2014a), measuring alignment complexity (Maillette de Buy Wenniger and Sima'an, 2013a, 2014b) and visualizing hierarchical alignment structure (Maillette de Buy Wenniger and Sima'an, 2014b) done in the context of this thesis.

- Based on HATs we propose two types of reordering labels for HIERO:

  - 0^{th}-order labels, that describe the reordering at child phrases relative to the current phrase.
  - 1^{st}-order labels, that describe the reordering of the current phrase relative to an embedding parent.

  We combine these new labels with decoding with relaxed label matching constraints and special features marking types of label substitutions used in derivations. This approach which we call *elastic-substitution decoding* is empirically tested on Chinese–English, German–English and English–German translation and gives significant improvement over HIERO for these language pairs while also performing favorably against a well known syntactic baseline (SAMT).

- We show that our proposed labeling schemes are superior to simplified reordering labels restricted to the cases Monotone, Inverted and Discontinuous from inversion transduction grammar (ITG), while even those simplified labels already give an improvement over no reordering labels.

- We propose a theoretical framework to exactly answer the question if given an SCFG and a word alignment the SCFG can be said to cover the word alignment. This framework is based on the intersection of the sets of TEUs inducible from the word alignment and producible by the grammar. Based on this framework and HATs we report an exact empirical analysis of alignment complexity. As part of this analysis we compute the proportion of word alignments that can be covered by ITGs and PETs respectively, and the proportion of word alignments

that can only be represented by grammatical frameworks equivalent to the full class of HATs. Experiments on both manually and automatically aligned parallel corpora for three language pairs give new insights into the complexity and structure of empirical word alignments.

1.7 Thesis Overview

1.7.1 Chapter 2: Preliminaries: SMT and Translation Equivalence

In this chapter we cover the foundations of modern statistical machine translation (SMT). We start by mentioning some of the historic roots in *rule-based* (Vauquois, 1975; Toma, 1977) machine translation (MT) and *example-based* MT (Nagao, 1984; Sato and Nagao, 1990), followed by a more systematic discussion of how translation models can be categorized. After this we give an overview of the core problems of SMT: 1) Translation model definition, 2) training, 3) decoding. We discuss the relation between these core problems and additional important subproblems, such as reordering and feature weights learning.

Following this, methods to establish translation equivalence relations at the word level (word alignments) are covered. Word alignments play a central role in this thesis. As usual, they serve as the input for extracting phrase pairs and from those hierarchical phrase pairs, used by hierarchical SMT. Additionally, they yield the concept of hierarchical translation equivalence as made explicit by HATs, covered in chapter 4. HATs in turn are used to extract reordering labels in chapter 5. Finally, through HATs, word alignments provide the input for the empirical analysis of hierarchical translation equivalence, covered in chapter 6.

Next, we discuss motivations for using composed TEUs and give some formal definitions of translation equivalence in MT, with phrase pairs as one important special case. We then discuss phrase-based SMT, starting with a brief overview of the alignment template approach (Och and Ney, 2004) followed by a more complete discussion of mainstream phrase-based translation (Koehn et al., 2003; Zens et al., 2002). Phrase-based SMT is not itself used in our experiments, but is still important for two reasons. First, hierarchical phrase-based SMT is the focus of this thesis. It is based on phrase-based SMT and borrows many features and techniques from it. Second, the limitations of phrase-based SMT with respect to long-distance reordering are important motivations for hierarchical translation.

This chapter ends with a brief overview of feature weights training methods, focusing on the minimum error rate training (MERT) (Och, 2003) and margin infused relaxed algorithm (MIRA) (Crammer and Singer, 2003) methods.

1.7.2 Chapter 3: Background Hierarchical SMT and Synchronous Grammars

Continuing from the discussion of phrase-based SMT in chapter 2, this chapter introduces SCFGs (Aho and Ullman, 1969). After first describing formal foundations, the most important SCFG algorithms are introduced: 1) parsing, 2) decoding, 3) expectation maximization. The discussion zooms in on binary SCFGs, particularly on one popular variant of these known as inversion transduction grammars (ITGs). This grammar formalism is of particular importance, since it is the foundation for an influential lexicalized variant, known as Hiero (Chiang, 2005), which forms the basis of modern hierarchical phrase-based SMT and its variants. After discussing hierarchical phrase-based SMT (Hiero), we continue the chapter with an in-depth discussion of a popular syntactically labeled variant known as syntax-augmented machine translation (SAMT). SAMT is particularly relevant, as it was one of the first approaches that succeeded in marrying phrase-based hierarchical SMT with syntax, while improving translation performance. Two important factors in its success are: 1) A heuristic labeling regime that avoids the rejection of Hiero rules when no syntactic labels can be found, 2) smoothing of phrase weights, to avoid problems with sparsity of labeled rule variants.

After introducing SAMT, we continue with the discussion of various other labeling approaches for hierarchical SMT. Almaghout et al. (2010) use CCG (Steedman, 1987, 2000) to label Hiero grammars. Li et al. (2012b) use dependency parse information to select part-of-speech (POS)-tags to form labels that encode a form of *syntactic head* information. We also cover approaches to automatically coarsen SAMT label (Hanneman and Lavie, 2013; Mino et al., 2014) and a method to automatically learn labels using the Cross-Validated EM algorithm (Mylonakis and Sima'an, 2011). We end the discussion of other labeling approaches with a review of latent-variable SCFGs for hierarchical SMT (Saluja et al., 2014).

Before ending the chapter we briefly discuss approaches to *soft constraints*. These approaches are crucial in overcoming problems with sparsity and loss of coverage when adding syntax or other information to hierarchical SMT through labeling approaches. We focus in the discussion on *preference grammars* (Venugopal et al., 2009) and soft syntactic constraints as used by Chiang (2010).

At the end of the chapter we review the work concerning the adoption of (hierarchical) lexicalized orientation models in hierarchical SMT (Huck et al., 2013; Nguyen and Vogel, 2013a). These models condition the reordering of phrase pairs relative to surrounding phrase pairs on lexical context. This work has particular relevance in the context of this thesis, as it shares a partially common motivation and somewhat similar mechanisms with the reordering labels that are proposed in chapter 5.

1.7.3 Chapter 4: Representing Hierarchical Translation Equivalence: Hierarchical Alignment Trees

How can hierarchical translation equivalence be induced from word alignments and represented in a compact and unambiguous way? In this chapter we build further upon existing representations of maximal decompositions of permutations called *permutation trees* (PETs) (Gildea et al., 2006; Zhang and Gildea, 2007) and representations of general word alignments called *normalized decomposition trees* (NDTs) (Zhang et al., 2008a). We research how these representations can be extended in such a way that:

- The recursive reordering structure at every node is explicitly represented.
- It explicitly represents all induced contiguous TEUs, instead of only a single canonical maximal decomposition of the word alignment.

The result is a new representation called HATs that facilitates:

- The extraction of reordering labels, used to significantly improve the quality of hierarchical translation, in particular with respect to word order (covered in chapter 5).
- The intuitive visualization of the hierarchical translation equivalence structure of aligned sentence pairs (discussed in (Maillette de Buy Wenniger and Sima'an, 2014b), not covered in this thesis).
- The systematic and exact quantitative analysis of empirical translation equivalence (covered in chapter 6).

Chapter 4 motivates the need for HATs and explains intuitively what they are, based on examples. We then explain the relation between HATs and the existing frameworks of normalized decomposition trees (NDTs) and permutation trees (PETs); extensively discussing the former work in these frameworks. Having established these foundations of existing work, we discuss the concept of *hierarchical translation equivalence*. This concept consists of a full representation of all contiguous TEUs as well as all subsumption, mapping and reordering relations between them.

Set permutations are then introduced as a simple and conservative extension to permutations, that enables the representation of arbitrary many-to-many word alignments. These set permutations are used to form node operators that exactly represent the mapping relations between the source and target-side children of HAT nodes.

The last part of the chapter discusses algorithms to compute HATs based on a variant of CYK-parsing. An important part of the algorithm is the discussion of set-permutation labels. The chapter ends with a summary and outlook on the applications discussed in later chapters.

1.7.4 Chapter 5: Bilingual Reordering Labels

Chapter 5 shows how grammars for hierarchical SMT (Hiero) can be enriched with labels that are not derived from syntax, in such a way that the word order of produced translations is improved. While the lexicalized rules of Hiero grammars embed the reordering decisions within rules in a lexical context, this lexicalization does not improve the coherence of reordering decisions across rules. It also does nothing to inform the reordering decisions made for plain, fully lexicalized rules (phrase pairs).

Syntactic labeling schemes such as SAMT (Zollmann and Venugopal, 2006) and Hiero enriched with CCG labels (Almaghout et al., 2010, 2012) or syntactic head information (Li et al., 2012b,a) have provided one successful approach to improve word order and fluency in hierarchical SMT for some language pairs. But these approaches have two disadvantages:

- They require good quality syntactic annotations (such as consistuency parses), which are not available for all languages.
- The used syntactic annotations are not necessarily compatible with the structure of hierarchical translation equivalence as induced by the word alignments.

While scarce syntactic annotations mostly restrict the applicability of these schemes, the incompatibility of syntax with the structure of hierarchical translation equivalence leads to more fundamental problems. One such problem is the lack of appropriate labels for many phrase pairs. This problem can be reduced by relaxing the syntax to have higher compatibility with the alignment structure, as is done for example in (Zollmann and Venugopal, 2006; Hassan et al., 2006, 2007; Cherry, 2008). But the price of this is often a high number of rare alternative labels, leading to data sparsity.

The approach proposed in this chapter is to rather use labels that are directly induced from the hierarchical translation equivalence structure induced by word alignments as represented by hierarchical alignment trees (HATs). This avoids the need for syntax, and assures that all rules can be assigned non-trivial labels. Furthermore, by applying a smart form of (heuristic) bucketing over the type of node labels (set permutations) of HAT nodes (i.e. phrase pairs), it is possible to adequately limit the total number of labels to a few meaningful categories. This benefits effective learning.

Following the introduction, the chapter shortly reviews closely related work on lexicalized orientation models (Xiao et al., 2011; Nguyen and Vogel, 2013a; Huck et al., 2013) and decoding with relaxed matching constraints (Chiang, 2010). After this short overview, it discusses how two types of reordering labels can be formed by bucketing node operators. The first label type (0^{th}-order) characterizes the decomposition of a phrase pair and reordering of its children. The second label type (1^{st}-order) characterizes the reordering of a phrase pair relative to an embedding parent phrase pair.

While these labels can be used in a standard decoding setting, with strict matching of labels, it turns out that working with relaxed label matching constraints during

decoding (elastic-substitution decoding) is important for getting the best results. This approach requires features that represent the substitution of the specific labels for the left-hand side of rules to the specific labels of the right-hand-side nonterminals of the rules these labels are extending. We accordingly present *label-substitution features* to allow the system to learn preferences for label substitution.

The proposed labels are tested on German–English, English–German and Chinese–English translation. They yield significant improvements over HIERO for all three language pairs when working with elastic-substitution decoding and also outperform SAMT. With extensive experimentation, we systematically investigate the influence on the results of the labels and label granularity, the label-substitution features and decoding type. The chapter ends with an overview of more distantly related work and conclusions.

1.7.5 Chapter 6: Empirical Analysis Hierarchical Translation Equivalence

Given an SCFG and a word alignment, does the SCFG cover the word alignment? In the first part of this chapter we focus on the problem of answering this question in a formally well defined way. As it turns out, the solution to this problem depends on what it means for a SCFG to cover a word alignment. This problem turns out to be non-trivial since:

- Arbitrary SCFGs do not generate word alignments as such.
- Terminals on the right-hand side of SCFG rules are not aligned together.

The question is then how to bind the terminals in order to check the coverage of word alignments. One important criterion for a proper solution is that it must capture the compositional translation equivalence structure. This implies that for general SCFGs, lexicalization of a TEU to form a flat structure should only be allowed if no further decomposition of that TEU is possible.

Existing work focuses on ITG and works mainly by counting TEUs induced by word alignment that are not covered by ITG (Zens and Ney, 2003; Wellington et al., 2006; Søgaard and Wu, 2009; Søgaard and Kuhn, 2009a; Søgaard, 2010). The existing literature disagrees on how to measure ITG word alignment coverage, and so reported results are diverging, see (Søgaard and Wu, 2009). Furthermore, ITG algorithms are ITG specific and do not generalize well to other SCFGs.

In this chapter, we propose a different, and more general approach. To test if a word alignment is covered by a SCFG, we take the set of TEUs extracted from the word alignment and check that it is a subset of the TEUs generated by that SCFG. We additionally require that the SCFG generates the TEUs of the word alignment with identical subsumption relations as in the word alignment.

In experiments with different language pairs and different parallel corpora we compute the fraction of TEUs covered by normal form inversion transduction grammar

(NF-ITG) using the precise measure of intersection. We also study the influence of the definition of TEUs (continuous/discontiguous) on alignment coverage; as well as the differences in alignment coverage between hand-aligned and corpora and automatically aligned corpora.

Working with discontiguous TEUs lowers ITG coverage because it causes discontiguous TEUs that are embedded in atomic TEUs and ignored in the continuous interpretation to obstruct ITG coverage. The effects of this are small in hand aligned corpora and somewhat bigger in automatically aligned corpora. On a general level, coverage of NF-ITG is considerably higher in manually aligned corpora than in automatically aligned corpora. This can be attributed to the different criteria and methods used to form manual versus automatic alignments.

In the second part of the chapter we explain how HATs are constructed in such a way that a minimally complex structure is formed, necessary to capture the hierarchical translation equivalence structure induced by the word alignment. This guarantees that:

- HATs maximally decompose the word alignment recursively into the set of induced TEUs ($\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$).[10]
- The structure of HATs ensures that all subsumption relations are represented.

Since the constructed HATs are the minimally complex structures required to capture hierarchical translation equivalence, it follows that grammars need to be minimally as complex as the HATs to be able to cover the word alignments. This allows HATs to be used as a shortcut to directly determine the required complexity of the grammars, without performing explicit set comparisons of TEUs for grammars and word alignments.

Using HATs we then measure alignment coverage and other alignment complexity metrics for three language pairs and both automatically and hand-aligned parallel corpora. The results reveal that many word alignments are neither binarizable nor bijective, and require the full set of arbitrary mappings (set permutations) to be produced.

[10]Here $\mathbf{f}$ and $\mathbf{e}$ are a source and target sentence and $\mathbf{a}$ is their corresponding word alignment, as described in chapter 2, subsection 2.2.1.

Sources of the Chapters

Some of the chapters in this thesis are partly based on earlier published work. Chapter 4 presents hierarchical alignment trees (HATs) and is partly based on material earlier published in (Maillette de Buy Wenniger and Sima'an, 2014b) and (Sima'an and Maillette de Buy Wenniger, 2013). Chapter 5 presents the work on bilingual Markov reordering labels. It is based on the following publications:(Maillette de Buy Wenniger and Sima'an, 2013b, 2014a, 2016). Finally, chapter 6 presents work on the empirical analysis of hierarchical translation equivalence. The first part of this chapter is mostly concerned with formal considerations of what it means to parse a word alignment using a synchronous grammar, and is largely based on (Maillette de Buy Wenniger and Sima'an, 2013a). The second part of this chapter is concerned with the actual empirical analysis of word alignments and alignment coverage, using HATs as a tool in doing so. This second part is partly based on (Maillette de Buy Wenniger and Sima'an, 2014b) and to a small extend on (Sima'an and Maillette de Buy Wenniger, 2013). The work described in (Maillette de Buy Wenniger et al., 2010) is not a direct source for any of the chapters in this thesis, but has been relevant as an early inspiration and motivation to pursue the performed research.

Maillette de Buy Wenniger, G., Khalilov, M., and Sima'an, K. (2010). A toolkit for visualizing the coherence of tree-based reordering with word-alignments. *The Prague Bulletin of Mathematical Linguistics*, pages 97–106.

Maillette de Buy Wenniger, G. and Sima'an, K. (2013a). A formal characterization of parsing word alignments by synchronous grammars with empirical evidence to the itg hypothesis. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 58–67. Association for Computational Linguistics.

Maillette de Buy Wenniger, G. and Sima'an, K. (2013b). Hierarchical alignment decomposition labels for hiero grammar rules. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 19–28.

Maillette de Buy Wenniger, G. and Sima'an, K. (2014a). Bilingual markov reordering labels for hierarchical SMT. In *Proceedings of the Eight Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 11–21.

Maillette de Buy Wenniger, G. and Sima'an, K. (2014b). Visualization, search and analysis of hierarchical translation equivalence in machine translation data. *The Prague Bulletin of Mathematical Linguistics*, (101):43–54.

Maillette de Buy Wenniger, G. and Sima'an, K. (2016). Labeling hiero grammars without linguistic resources. *Machine Translation*, pages 1–41.

Sima'an, K. and Maillette de Buy Wenniger, G. (2013). Hierarchical alignment trees: A recursive factorization of reordering in word alignments with empirical results. Internal Report.

2

Preliminaries: SMT and Translation Equivalence

One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

– Warren Weaver, *In letter to Norbert Wiener, 4 march 1947*

Machine translation (MT) aims to find the most plausible target sentence (translation) in some language, given a source sentence (input) in some other language. But this seemingly simple problem turns out to be hard in practice. It triggers many very complex subproblems, that are at the core of artificial intelligence (AI). To start, the number of possible translations, essentially the number of possible sentences in natural languages, is unbounded. This means that even if it were possible to perfectly assess the quality of translations, choosing a translation within reasonable time would still require being very selective about what translations to generate.

In the past a proven method to produce high quality translations was to build a system using linguistic rules often extracted from dictionaries, grammars and other monolingual and bilingual linguistic resources. This line of work, focusing mostly on hand-crafted rules, is known as *rule-based* machine translation (Vauquois, 1975; Toma, 1977; Vauquois and Boitet, 1985; Arnold and Tombe, 1987). An alternative approach, called *example-based* machine translation (Nagao, 1984; Sato and Nagao, 1990; Sumita and Iida, 1991) forms translations by adapting stored examples of source sentences with their translations to translate the new input. In contrast to the early rule-based approaches, this framework explicitly tries to find the best matching translation from data, by scoring multiple competing alternatives. Yet the methods of determining the similarity of matching examples to the input remains largely heuristic. Both rule-based and example-based machine translation lack a solid mathematical and statistical foundation. This has made it very hard for these approaches to offer a general, structured solution to the fundamental ambiguity of translation. The introduction

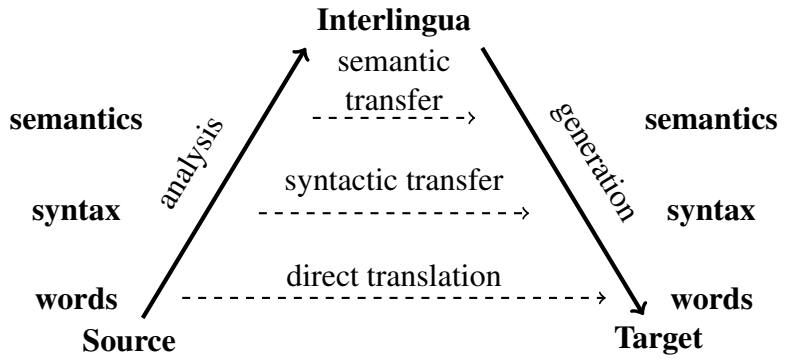


Figure 2.1: The Machine Translation pyramid

of SMT (Brown et al., 1988), marked the beginning of a new era in MT research, distinguished by a more structured and statistically grounded approach to MT. This thesis broadly concerns SMT and specifically focuses on improvement of the word order and coherence of translations for hierarchical SMT.

This chapter and the next one lay the foundation for the rest of the work in the thesis by presenting the required basic concepts and essential frameworks. We start by giving a categorization of translation models, after which we zoom in on Statistical Machine Translation (SMT), and discuss some of its important high level elements. We then continue in section 2.2 with a description of word alignment, the process by which translation equivalence on the word level is learned. In this we focus on the IBM models. In the second part we then give a description of translation equivalence in MT. followed by a description of phrase-based translation. We end the chapter with a brief discussion of feature weights training: methods to learn the weights of different components used by a translation model.

Categorization of translation models Translation systems can be classified based on how far they abstract from the lexical surface form of source and target words (Vauquois, 1968). Words can be directly translated into words, which is known as *direct translation*, and SMT has clearly exemplified the success of this approach (Brown et al., 1988, 1993; Och and Weber, 1998; Koehn et al., 2003; Ittycheriah and Roukos, 2007). Alternatively words can be analyzed as syntactic or even semantic representations, which are then transferred to the target side, followed by a generation step that produces the more concrete representations, ending with the target words (Bennett and Slocum, 1985; Johnson et al., 1985; Kaplan et al., 1989; Dorr, 1994; Dorna et al., 1998). Figure 2.1 shows these alternative paths in a historic scheme known as the *machine translation pyramid*. Theoretically, source input can be mapped to an extremely abstract meaning representation called *interlingua*, followed by downwards generation of the target words from that representation. This approach of extreme abstraction has been popular in the past before the advance of SMT. But this approach has been elusive in practice, and it has not yielded working MT systems.

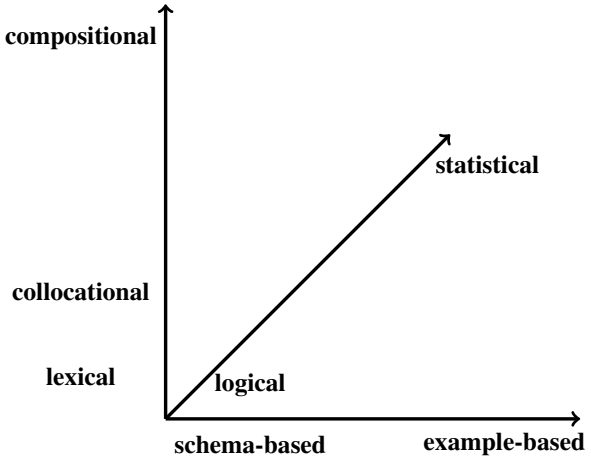


Figure 2.2: The space of MT models (Wu, 2005)

The formulation of a formal language that can effectively capture all the possible meanings of natural language effectively has proven to be problematic. Secondly, approaches to describe natural language with formal languages often ignored or marginalized the ambiguity of language in their representations. Since ambiguity is a core component of natural language, these approaches thereby arguably compromised their own chances of success right from the start. In response to these realizations a more modest approach based on syntactic transfer was adopted in the rule-based Eurotra project (Arnold and Tombe, 1987). More recently, the application of syntax and more structurally rich translation rules has also proven its merit in hierarchical SMT (Zollmann and Venugopal, 2006; Chiang, 2010; Almaghout et al., 2011).

An alternative classification of translation systems based on 3-dimensions is proposed by (Wu, 2005), see Figure 2.2. The x-axis represents to what extent translation is performed by generalization or adaptation of stored examples during testing (*example-based*), as opposed to generalization during training (*schema-based*). The y-axis indicates to what extent rules are *compositional*, making use of recursion. The IBM models, which are fully lexical are found at the bottom of this axis. Fully compositional models are found at the top of this axis. For SMT hierarchical phrase-based models (Chiang, 2005), syntactic hierarchical SMT e.g. (Galley et al., 2004) and stochastic inversion transduction grammars (ITGs) (Wu, 1997) are well known examples of compositional models. Phrase-based (*collocational*) models are somewhere in between, because they compose lexical items into larger chunks but do not use categories or grammars to combine these chunks. The z-axis shows the last dimension of classification: *logical* systems versus systems based on statistics. For EBMT the early work (Nagao, 1984) was purely logical, while a much later approach by (Quirk and Menezes, 2006) combines the strengths of phrase-based SMT with dependency parsing and insights from EBMT. This produced a hybrid system that is strongly statistical. In a similar way, rule-based MT started out with a focus on compositional word-to-word translation without statistics (Locke, 1955). But while

later working systems such as the early Systran system were initially still purely logical (Toma, 1977), if continued they would eventually often be adapted to incorporate statistics in more and more places (Senellart et al., 2003).

2.1 Statistical Machine Translation

Statistical Machine Translation (SMT) aims to find the statistically most plausible translation **e** for an input source sentence **f** by means of a computer program. SMT employs statistical modelling in combination with machine learning and optimization methods to achieve this goal. SMT assigns to every target sentence **e** a probability for being the translation of the input sentence **f** based on a probability distribution $p(\mathbf{e}|\mathbf{f})$. Using this probability distribution, the translation with the highest conditional probability can be determined:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \tag{2.1}$$

SMT divides the translation problem into a few important core subproblems:

- **Translation model definition**: Defining a model that composes full translations from smaller atomic elements, and defines probabilities for full translations through parameters of atomic elements.
- **Training**: estimating the model parameters with the help of statistical estimation methods and machine learning techniques.
- **Decoding**: searching efficiently for the best translation within the space of possible translations that is exponential in the input length.

A simple working SMT system can be built by solving these subproblems. However, producing good translations requires additionally solving the following difficult problems, amongst others:

- **Reordering**: modeling word order differences as part of the translation model, and performing reordering as part of the decoding process.
- **Feature weights training**: automatically learning the relative importance of the different components (features) used by the translation model. Feature weights training is a part of training, but while essential for building strong systems, is not a required part for a minimal SMT system.
- **Language model learning and integration**: language models are an essential part of producing fluent translation output, and also play an important supporting role in finding a plausible word order for the output.

- **Morphology**: modeling the structure of language at a level below words i.e. at *morpheme* level. This is of particular importance for languages like Arabic and Hebrew, but also for example for most Slavic languages like Czech.
- **Semantics**: the study of meaning. Taking measures that encourage the preservation of meaning during translation are of crucial importance to take automatic translation to the next level.

We now give an overview of the core subproblems and their role in SMT, to provide a basic understanding.

Core subproblems SMT The process that is used to form translations is determined by the translation model. Complete translations may be formed by composing the translations of single words. Word-based translation models and their training are discussed in section 2.2. These models are also used to determine word-level translation equivalence relations, called word alignments. Alternatively translation may involve composition of bigger fragments called phrase pairs (Och and Weber, 1998; Wang and Waibel, 1998b; Zens et al., 2002; Koehn et al., 2003; Och and Ney, 2004), or even fragments with variables called hierarchical phrase pairs (Chiang, 2005). Based on word alignments, these larger translation rules can be composed and combined with features into translation models. This is discussed in section 2.3.

Apart from specifying the basic rules, a translation model must also specify how rules may be combined to form complete translations. And the model must assign a score or probability to translations, so that the best translation can be determined.

After a translation model has been formulated, and its parameters have been estimated, it is possible to compose translations for an input and compute their scores. But even with the restrictions of the model, there are still exponentially many translations that can be generated for an input sentence. We therefore need to search the space of possible translations efficiently, which is the problem of decoding (Berger et al., 1994; Tillmann et al., 1997a; Wang and Waibel, 1997, 1998a; Tillmann and Ney, 2000). Efficient solutions typically involve *dynamic programming*, the decomposition of a problem into smaller problems that can be solved independently, and then efficiently recombined into a solution to the original problem. But to permit dynamic programming, certain independence assumptions must be made in the translation model. This is where the decoding problem interacts with the modeling problem: the complexity and expressiveness of the translation model must be limited to guarantee a sufficiently efficient solution to the search problem.

Following this brief overview of the core subproblems of SMT, we will come back to the secondary problem of reordering later in this chapter in the context of phrase-based models. Improving reordering is the main topic of this thesis. Existing work on this topic will get more attention in the context of the next chapter, which discusses hierarchical SMT and synchronous grammars. Feature weights learning has an instrumental but essential role in our approach to improving word order. For the sake of conciseness, it will be briefly summarized at the end of this chapter in

Figure 2.3: The noisy channel model for translation

subsection 2.3.4, while a more complete overview is given Appendix A.2. Language modeling (Bahl et al., 1983; Brown et al., 1992; Charniak, 2001) and decoding will not be covered in detail because they are not directly essential to our contributions in this thesis, although they are essential for SMT. Similarly morphology and semantics, while important for SMT, fall outside the scope of this thesis and are therefore not discussed further.

2.1.1 The Noisy Channel Model

Shannon's noisy channel model (Shannon, 1948) was adopted early by SMT (Brown et al., 1988) and has been important ever since. In the noisy channel model, an input is corrupted while passing through a communication channel, producing the output. Applied to translation, instead of directly modeling the translation probability $p(\mathbf{e}|\mathbf{f})$, the source sentence ($\mathbf{f}$) is viewed as a corrupted version of the output translation ($\mathbf{e}$), produced by passing the translation through a noisy channel (see Figure 2.3). Mathematically, the noisy channel model then rewrites the direct maximization of the translation probability as the maximization of the product of a translation model component and a language model component. This is done by applying the Bayes rule and discarding the denominator, which has no influence on the maximization:

$$\begin{aligned}\hat{\mathbf{e}} &= \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})p(\mathbf{e})}{p(\mathbf{f})} \\ &= \arg\max_{\mathbf{e}} \overbrace{p(\mathbf{f}|\mathbf{e})}^{\text{translation model}} \overbrace{p(\mathbf{e})}^{\text{language model}} \end{aligned} \tag{2.2}$$

Splitting the probability function that is optimized into a translation model component and a language model component has important advantages. The language model is concerned only with the *fluency* of the output, and requires only monolingual data to be trained. This enables the use of huge amounts of training material, leading to very strong models. The translation model is focused on translation correspondence, also known as *adequacy*. Training the translation model requires bilingual data, which is less abundant. But because the translation model does not have to worry about producing fluent output, it can use the limited available data more effectively, focusing on the adequacy of bilingual mappings. Finally a separate strong language model plays an important role in selecting a proper word order.

In early SMT work, such as the IBM models, the noisy channel model was used in a literal way. Later work deviated from this approach, using the translation model and

language model as just two amongst a series of components used to asses the quality of translations in a log-linear framework. In this later work, the use of translation models in two directions $p(\mathbf{f}|\mathbf{e})$ and $p(\mathbf{e}|\mathbf{f})$ is also often practiced (Chiang, 2005). Both directions are typically heuristically estimated and imperfect, but can complement each other in providing partially non-overlapping information that helps to choose better translations. It is also common to use multiple, interpolated language models (Och and Ney, 2004; Chiang, 2010; Huck et al., 2013).

2.1.2 Generative and Discriminative Models

Generative translation models generate the source and target sentence synchronously as a structured statistical process. This allows the generation or sampling of synchronous productions, and furthermore allows the most likely translation for a source sentence to be determined from the joint probability of entire sentence pairs as given by the model:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} p(\mathbf{f}, \mathbf{e}) \tag{2.3}$$

Examples of generative models for translation are phrase-based models and the synchronous grammars used by hierarchical phrase-based models.[1] The original conditional word-based translation models were also formulated as a generative process, emitting $\mathbf{f}$ from $\mathbf{e}$ (Brown et al., 1993).

The formulation of successful generative models for translation is an engineering process that requires much effort. All steps and transformation of the translation process must be included. Furthermore, independence assumptions are essential to keep the level of model parameters at a manageable level and avoid *overfitting*. But the need for independence assumptions must be carefully balanced against loss in ability to adequately model translation phenomena which violate them.

Discriminative models in contrast directly model the conditional distribution $p(\mathbf{e}|\mathbf{f})$. This avoids the need to formulate a full generative model, mostly eliminating the need for predetermined independence assumption except for reasons of computational efficiency.[2] For SMT, discriminative models for translation are typically implemented by combining a set of feature functions $\phi_m(\boldsymbol{e}, \boldsymbol{f})$. Each feature function outputs a non-negative value that encodes information that is supposed to be relevant for the assessment of the quality of the translations. Without the need to formulate a generative story, the modeler only has to think about selecting features that are useful

[1] Modern phrase-based and hierarchical phrase-based models use many features to choose the most plausible translation. Many of these features are not part of the generative process and many also not even have a probabilistic interpretation. This makes these models discriminative rather than generative. However, the original core of these models is described as a generative process.

[2] Most features in (discriminative) phrase-based and hierarchical phrase-based translation are still local to the rules, with the exception of the language model. There is a clear reason for this, features that use information from the target side beyond rule boundaries increase dependencies, reducing the efficiency of dynamic programming as applied by the decoder and increasing computational cost.

to discriminate stronger from weaker translations. The log linear model combines the features as a weighted sum, using for every feature $\phi_m(\boldsymbol{e}, \boldsymbol{f})$ a specific weight λ_m. Dividing by the total weight obtained by summing over all possible translations turns the total feature weight into a probability:[3]

$$p(\boldsymbol{e}|\boldsymbol{f}) = \frac{exp\Big(\sum_{m=1}^{M} \lambda_m \phi_m(\boldsymbol{e}, \boldsymbol{f})\Big)}{\sum_{\boldsymbol{e'}} exp\Big(\sum_{m=1}^{M} \lambda_m \phi_m(\boldsymbol{e'}, \boldsymbol{f})\Big)} \tag{2.4}$$

Learning adequate feature weights is essential for discriminative methods. Basic approaches include maximum entropy (Berger et al., 1996b) and minimum error rate training (MERT) (Och, 2003). In section 2.3.4 we provide a brief review of MERT, a longer review of MERT and some of the more recent approaches for feature weights training in SMT is given in Appendix A.2. Lastly there is a big difference in the way that discriminative methods are applied in mainstream machine learning, such as in simple classification task, and the application of these methods in SMT. In basic classification tasks, the examples to be classified are given, and sometimes the space of possible examples is also finite. In SMT in contrast, because the space of possible translations is exponential in the input length, generating a set of plausible translations is an essential part of the problem, and cannot be solved by discriminative methods. Therefore phrase-based and hierarchical phrase-based systems essentially use a hybrid approach that combines a generative and a discriminative component. The generative component is the model that determines how translations are formed, in combination with the core features such as phrase weights and language model probabilities. Through the decoder, these components strongly steer the space of translations to be considered in the first place, while also providing a strong basic vote for the ranking of these considered candidates. The other features augment and refine this ranking, but change nothing to the initial translation selection process determined by the nature of the translation model in combination with the decoder. Even with the recent popularity of neural networks in SMT, this basic approach of keeping a phrase-based or hierarchical phrase-based translation model but extending it with additional features based on neural networks remains popular (Cho et al., 2014; Tran et al., 2014). The mostly generative translation selection process of the decoder can also be followed by a purely discriminative *reranking* process, working on an N-best list of selected translations provided by the decoder (Shen et al., 2004; Duh and Kirchhoff, 2008; Sokolov et al., 2012). This illustrates the need for discriminative models to work with a limited set of translations even more sharply. In parsing greedy approaches that sacrifice the ability to compactly represent an entire chart of parses for the benefit of using much more context have shown remarkable success (Zhang et al., 2014). But as an alternative, (Huang, 2008) has shown that (approximate) reranking of entire forests with non-local features is also possible. The dilemma between (nearly) optimal search

[3] In practice when applying the model this normalization step is often not even necessary, except in some cases, when learning the weights.

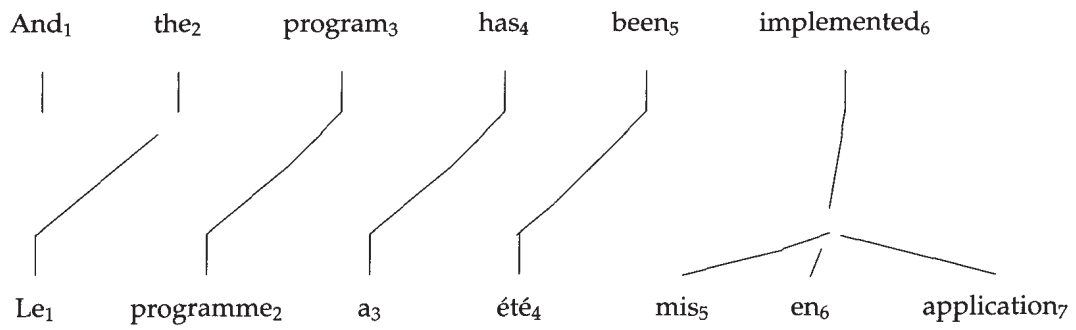


Figure 2.4: Example of many-to-one English–French word alignment taken from the original IBM model paper (Brown et al., 1993), highlighting the type of alignments IBM models permit (not general many-to-many word alignments).

in a simpler model or alternatively (strongly) greedy search in a much richer model is expected to remain relevant for machine translation and parsing alike in the future. But at the same time, improvements in search and optimization algorithms and finally computing power promises to offer increasingly good combinations of and tradeoffs between the two with ever greater success for increasingly more complex models.

2.2 Word Alignments

The best known theory for word alignment as used in statistical machine translation is the theory behind the IBM models, described in (Brown et al., 1993). In this theory the way the French string is produced from the English string by the generative probability model is conceptualized as follows. The English string is considered to be capturing a set of concepts which together produce the French sentence. However, in word alignment only the English words are used and not the (hidden) concepts. Therefore, the complete set of English words is divided into a collection of possibly overlapping subsets that substitute for the (actual) concepts. These subsets are called *cepts*. Word alignments then make explicit the connection between the English cepts and the French words they generate. But the IBM models allow only many-to-one alignments and not the more general case of many-to-many alignments. Therefore this general story is further restricted, by considering only the individual English words as cepts, and allowing multiple French words to be generated by each of these cepts.[4] Figure 2.4 shows an example of a many-to-one English–French word alignment, with independent English words generating a set of French words, which is the type of word alignments IBM models support.

The original IBM Model narrative is helpful to the extent that it provides some conceptual interpretation of what goes on in the models. It also gives a motivation for the particular decompositions used by the models of a very general joint probability

[4]Besides the actual English word a special additional cept called the *empty cept* is used to allow French words to be generated out of nothing, and not aligned to anything.

distribution for source sentence, target sentence and alignment together. On the other hand, from a mathematical point of view this story is not really necessary. Och and Ney (2003) leave the discussion of cepts completely out in their newer presentation of the IBM models, highlighting the point that from a mathematical point of view the different models are just alternative decompositions of a general joint probability distribution. We think this latter, more "modest" interpretation makes less assumptions, with the advantage of leaving more room to fit alternative models of word alignment such as discriminative methods and hierarchical alignment methods (Riesa and Marcu, 2010; Burkett et al., 2010) within the same conceptual framework.

The central question for an alignment model is: how can sentence alignments be expressed in terms of word alignments?And more specifically: how likely is each of the permitted word alignments? The form of the assumed alignment model puts restrictions on what type of word alignment (distributions) can be learned. These restrictions are crucial for enabling learning, and are sometimes referred to as *inductive bias* of the model (Mitchell, 1980). Through these restrictions put by the model on the permissible word alignment distributions, the model determines how to generalize from the unseen sentence alignments to predict the hidden word alignments.

2.2.1 The mathematical framework of statistical word alignment

In the following sections we will review the mathematical properties of statistical translation models. In doing so we follow (Och and Ney, 2003), and in particular their notation in describing the relevant formulas. The symbol $Pr(\cdot)$ is used to denote general probability distributions. In contrast, the generic symbol $p(\cdot)$ is used to denote model-based probability distributions.

Given a sentence pair $\{\mathbf{f}, \mathbf{e}\} = \{\mathbf{f}_1^J, \mathbf{e}_1^I\}$ with $\mathbf{f}_1^J = \mathbf{f}_1 \dots \mathbf{f}_J$ and $\mathbf{e}_1^I = \mathbf{e}_1 \dots \mathbf{e}_I$. In statistical alignment models the alignment $\mathbf{a}_1^J = \mathbf{a}_1 \dots \mathbf{a}_J$ is a hidden variable describing the hidden mapping from source to target positions. The translation model is obtained from the alignment model by marginalization over all alignments as expressed by the following relation between the two models:

$$Pr(\mathbf{f}_1^J|\mathbf{e}_1^J) = \sum_{\mathbf{a}_1^J} Pr(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^I) \tag{2.5}$$

This general formula represents the generation of the parallel sentence pair and word alignment in its most general form without any extra assumptions. But typically alignment models are parametric probability distributions that depend on a set of unknown parameters θ that are learned from training data. We use the following notation to express this:

$$Pr(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^J) = p_\theta(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^I) \tag{2.6}$$

Maximization of the likelihood of the parallel training corpus $\{\mathbf{f}_s, \mathbf{e}_s\}_{s=1}^{S}$ is used to determine the optimal values for the unknown parameters θ:

$$\hat{\theta} = \underset{\theta}{argmax} \prod_{s=1}^{S} \sum_{a} p_\theta(\mathbf{f}_s, a|\mathbf{e}_s) \tag{2.7}$$

Typically the expectation maximization (EM) algorithm (Dempster et al., 1977) is used to perform the maximization, but other optimization algorithms are also applicable.

It is important to be able to find the best alignment for a given sentence pair. In principle although there can be a large number of alignments, this best alignment can always be found:

$$\hat{a}_1^J = \underset{\mathbf{a}_1^J}{argmax}\ p_{\hat{\theta}}(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^I) \tag{2.8}$$

This alignment $\hat{a}_1^J$ is known as the **Viterbi alignment** of the sentence pair $(\mathbf{f}_1^J, \mathbf{e}_1^I)$.

Based on this general probabilistic framework, concrete alignment models are created by introduction of inductive bias, leading to specific parameterized probability distributions. We will next look at various popular alignment models, how they are derived from the original general formula, and how they relate to each other.

2.2.2 Statistical Alignment Models

Concrete alignment models are derived from the general formula by making additional assumptions with respect to the form of the distribution, typically in the form of independence assumptions, which allow to rewrite this formula as a certain product of independent factors. In what follows we will cover:

- hidden Markov model (HMM) alignment model
- IBM models 1 and 2
- Fertility-based alignment models

Och and Ney (2003) note that the alignment model $Pr(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^J)$ can be structured without loss of generality as follows[5]:

$$Pr(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^J) = Pr(J|\mathbf{e}_1^I) \cdot \prod_{j=1}^{J} Pr(\mathbf{f}_j, \mathbf{a}_j|\mathbf{f}_1^{j-1}, \mathbf{a}_1^{j-1}, \mathbf{e}_1^I) \tag{2.9}$$

$$= Pr(J|\mathbf{e}_1^I) \cdot \prod_{j=1}^{J} Pr(\mathbf{a}_j|\mathbf{f}_1^{j-1}, \mathbf{a}_1^{j-1}, \mathbf{e}_1^I) \cdot Pr(\mathbf{f}_j|\mathbf{f}_1^{j-1}, \mathbf{a}_1^{j}, \mathbf{e}_1^I) \tag{2.10}$$

[5]Using the **chain rule** of probability.

This decomposition highlights three components of the composed probability function: a length probability $Pr(J|\mathbf{e}_1^I)$, an alignment probability $Pr(\mathbf{a}_j|\mathbf{f}_1^{j-1}, \mathbf{a}_1^{j-1}, \mathbf{e}_1^I)$ and a lexicon probability $Pr(\mathbf{f}_j|\mathbf{f}_1^{j-1}, \mathbf{a}_1^j, \mathbf{e}_1^I)$.

The hidden Markov model (HMM) alignment model (Vogel et al., 1996) is derived by assuming:

- A first-order dependence for the word alignments $\mathbf{a}_j$
- A lexicon probability that only depends on the word at position $\mathbf{a}_j$

which gives the following simplified model components:

$$Pr(\mathbf{a}_j|\mathbf{f}_1^{j-1}, \mathbf{a}_1^{j-1}, \mathbf{e}_1^I) = p(\mathbf{a}_j|\mathbf{a}_{j-1}, I) \quad (2.11)$$

$$Pr(\mathbf{f}_j|\mathbf{f}_1^{j-1}, \mathbf{a}_1^j, \mathbf{e}_1^I) = p(\mathbf{f}_j|\mathbf{e}_{\mathbf{a}_j}) \quad (2.12)$$

Finally a simple length model $Pr(J|\mathbf{e}_1^I) = p(J|I)$ is assumed.[6] Putting everything together following basic HMM-based decomposition of $p(\mathbf{f}_1^J|\mathbf{e}_1^I)$ is obtained:

$$p(\mathbf{f}_1^J|\mathbf{e}_1^I) = p(J|I) \cdot \sum_{\mathbf{a}_1^J} \prod_{j=1}^{J} p(\mathbf{a}_j|\mathbf{a}_{j-1}, I) \cdot p(\mathbf{f}_j|\mathbf{e}_{\mathbf{a}_j}) \quad (2.13)$$

There are some more details in (Och and Ney, 2003) omitted here for brevity. These concern how the alignment probabilities are made independent of absolute word positions and how the network is extended with empty words to produce source words without aligned target words.

2.2.3 IBM Models 1 and 2

The relation between the hidden Markov alignment model (HMM) and IBM model 1 and 2 is clearly explained by (Och and Ney, 2003). Where the HMM is based on first-order dependencies $p(i = \mathbf{a}_j|\mathbf{a}_{j-1}, I)$ for the alignment distribution, Models 1 and 2 in contrast use zero-order dependencies $p(i = \mathbf{a}_j|j, I, J)$

- For Model 1 a uniform distribution $p(i|j, I, J)$ is used:

$$Pr(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^J) = \frac{p(J|I)}{(I+1)^J} \cdot \prod_{j=1}^{J} p(\mathbf{f}_j|\mathbf{e}_{\mathbf{a}_j}) \quad (2.14)$$

This means that the word order does not affect the alignment probability.

[6] In the original paper (Brown et al., 1993) describing the IBM models, an even stronger assumption is made for the length model, namely that the length J is independent of the English sentence $\mathbf{e}_1^I$ which leads to the usage of a simple uniform prior ϵ as the length model: $Pr(J|\mathbf{e}_1^I) = \epsilon$.

- For IBM Model 2

$$Pr(\mathbf{f}_1^J, \mathbf{a}_1^J|\mathbf{e}_1^J) = \frac{p(J|I)}{I+1} \cdot \prod_{j=1}^{J} p(\mathbf{a}_j|j, I, J) \cdot p(\mathbf{f}_j|\mathbf{e}_{\mathbf{a}_j}) \tag{2.15}$$

is obtained. In order to reduce the number of parameters, this is further simplified by ignoring the dependence J on the alignment model. Thus $p(\mathbf{a}_j|j, I)$ is used instead of $p(\mathbf{a}_j|j, I, J)$.

2.2.4 Fertility-Based Alignment Models

In the fertility-based IBM Models 3, 4 and 5 the generative processes are considerably changed by explicitly choosing the number of French words generated by each English word as the first step of the generation process. This new parameter is called the *fertility*. Thus, for a word $\mathbf{e}_i$ in position i, the fertility ϕ_i is defined as the number of aligned source words:

$$\phi_i = \sum_j \delta(\mathbf{a}_j, i) \tag{2.16}$$

In this, δ is the Kronecker delta function, which yields value 1 if its two arguments are the same and 0 otherwise. The models then have a probability $p(\phi|e)$ that the target word e produces ϕ source words, for different values of the fertility ϕ, including $\phi = 0$. These fertility parameters are combined with the earlier translation probabilities, and alignment probabilities now called *distortion* probabilities in these models. In the fertility-based IBM Models, for each English word, the selection of the fertility, the produced French words and their order (distortion) are separated as three consecutive steps in the generative process. To facilitate a more compact discussion (Och and Ney, 2003) combine the selection of the $fertility$ and the word order in a new function B, which gives for every target word the set of aligned source positions:

$$B : i \rightarrow B_i \subset \{1, \ldots, j, \ldots, J\} \tag{2.17}$$

Here B_0 contains the positions of all source words that are unaligned, i.e. aligned to the empty word. Using this new function, the fertility-based alignment models can now be formalized using the following decomposition and assumptions:

$$Pr(\mathbf{f}_j, \mathbf{a}_j|\mathbf{e}_1^I) = Pr(\mathbf{f}_1^J, B_0^I|\mathbf{e}_1^J) \tag{2.18}$$

$$= Pr(B_0|B_1^J) \cdot \prod_{i=1}^{I} Pr(B_i|B_1^{i-1}, \mathbf{e}_1^I) \cdot Pr(\mathbf{f}_1^J|B_0^I, \mathbf{e}_1^J) \tag{2.19}$$

$$= p(B_0|B_1^J) \cdot \prod_{i=1}^{I} p(B_i|B_{i-1}, \mathbf{e}_i) \prod_{i=0}^{I} \prod_{j \in B_i} p(\mathbf{f}_j|\mathbf{e}_i) \tag{2.20}$$

This equation assumes that the set B_0 of words aligned with the empty words is only generated after covering the nonempty positions. Note how in the derivation, in the first two steps we are merely rewriting the formula with no extra assumptions. Extra independence assumptions are introduced only in the last step to arrive at the final formula for this group of models (2.20). These independence assumptions are:

- The lexical selection is independent from all other steps
- The selection of the positions for the i-th English word depend only on that word, and the positions for the previous word.

Model 3, 4 and 5 differ in the details of the distortion component of the model, with very strong independence assumptions for model 3, that are weakened for model 4 and 5. We refer the interested reader to (Brown et al., 1993; Och and Ney, 2003) for more details. Model 3 and 4 are **deficient** (Brown et al., 1993), which means that probability mass is wasted on improper word alignments, so that the sum of probabilities for valid alignments does not sum to one. This is fixed in Model 5, at the price of even more parameters and a still considerably more complex model.

2.2.5 Parameter estimation

We will now very briefly touch upon the matter of parameter optimization of the IBM alignment models and HMM alignment model. The IBM Models 1 and 2 and the HMM alignment model are simple enough to have an EM algorithm derivation that uses a sum over alignments which can be calculated in an efficient closed form. This efficient closed form rearranges a sum over an exponential number of products as a product of sums, which for every sentence in the training set can be computed in time that is roughly quadratic in the sentence length. This allows efficient exact computation of the EM algorithm for these models. In contrast, such an efficient computation is not possible for IBM Model 3,4 and 5. Therefore in the expectation step of the EM computation for these models, the used approximation is to sum over only a small set of most likely word alignments and their close neighbors when computing the (new) counts used in the next maximization step. This set is found by starting from the best alignments for Model 2 (or the HMM alignment model), and using a greedy search method to improve upon those further. Once again details can be found in the original work (Brown et al., 1993). Another important aspect of the optimization regime, is that every incrementally more complex model has its parameters initialized by the previous simpler model, before the parameter optimization starts. This leads to a cascade of model optimizations whereby the simpler models are used to produce good starting parameters for the more complex ones, thereby increasing the change of ending up in a good rather than bad local optimum of the likelihood function for these more complex models. A detailed study of what cascade of what model optimizations leads to the best word alignments is given in (Och and Ney, 2003). One important outcome of their study is that both usage of the HMM alignment model, which is more sophisticated than IBM Model 1 and 2 but still efficiently optimizable by the EM algorithm, is important to achieve optimal result. This is determined by comparing

the *alignment error rate* for various schemes through comparison of the model Viterbi alignments against gold reference alignments. Finally a new model that combines the HMM model and Model 4 in a log-linear way helps to bring the alignment error rate even further down.

Symmetrization of Alignment Models

IBM models are asymmetric and model only one-to-many alignments, not general many-to-many alignments. To overcome this problem and improve the alignment quality, typically two alignment models are trained, one in source-to-target and one in target-to-source direction, and these two models are then combined into one final model. Intersection and union are the simplest schemes that can be used for the combination. More complex heuristic schemes that give higher alignment accuracy and yield better translation scores have been proposed (Och et al., 1999; Koehn et al., 2003). These schemes start from the intersection and grow this initial set of alignment links by adding more weakly supported links from the union provided they are in the neighborhood of existing links, and optionally adding links for remaining unaligned words as a final step.

Avoiding confusion about what symmetrization scheme is used and what it entails is key to the comparability and reproducibility of experiments. In Appendix A.1 we therefore describe the details of the different heuristic schemes and their naming in more detail.

2.2.6 Overview alternative alignment models

Symmetrization of two asymmetric alignment models in both directions is a simple and relatively effective way to get good quality many-to-many alignments. It is based on a combination of solid learning methods for the initial alignment model estimation and sensible heuristics for the symmetrization step. But Liang et al. (2006b) note that the disadvantage of this method is that the training of the asymmetric fertility-based models is highly complex, and computationally expensive. Also the complexity of these models makes them hard to reimplement. Finally modifications of these existing models, such as model 6 (Och and Ney, 2003) have yielded only modest improvements. This motivates Liang et al. (2006b) to propose a new alignment model that encourages agreement of two simple asymmetric alignment models already at training time. To this end the authors jointly train two HMM alignment models, one in source-to-target direction and the other in target-to-source direction. While training they maximize a combination of data likelihood and agreement between the models. This model cannot be efficiently optimized using the EM algorithm, so a heuristic approximation is required to make computation efficient. Using this approximation as the new model the authors achieve 32% reduction of alignment error rate (AER) in comparison to symmetrization of two alignment models by intersection of their alignments.

Word alignment models can also incorporate more (hierarchical) structure in the alignment process. This structure may or may not be syntactic. The structure can take a central place in the alignment model by requiring a particular hierarchical (possibly syntactic) model structure. But it can also be introduced as a soft constraint or supporting source of information for the alignment model. A popular way to strictly require hierarchical structure in the formation of word alignments is to assume them to be generated by a synchronous context free grammar or synchronous tree-substitution grammar. These grammars need not be syntactic. For example, the popular grammar formalism of inversion transduction grammars (ITGs) (Wu, 1997) require no syntactic structure, just a particular type of synchronous context free grammars.

The work by Xiao and Zhu (2013) in contrast requires syntactic structure on both sides, modeling word alignment as the task of finding a maximum likelihood (syntactic) tree-substitution grammar that generates the parallel training corpus. The grammar is estimated using the Expectation Maximization (EM) algorithm (Dempster et al., 1977) or a Bayesian method. In both cases a bias towards simple tree pair fragments is necessary to avoid extreme overfitting of the training data.[7]

This bias is introduced to the models in the form of hard constraints that limit the tree-fragment size as well as Bayesian priors that favor smaller fragments. A problem with such hard constraints on the tree-fragment size is that they have been shown in earlier work to lead to suboptimal models (Bod, 2001). In order to remain computationally feasible, very strong independence assumptions are made in the parameterization of the generative tree-substitution grammar model. When the proposed model is used to directly form rules, it loses over a baseline that uses heuristic rule extraction (Galley et al., 2004) based on the (sub-tree) alignment matrix that is estimated by their model. But performance is much improved when the model is applied to estimate better features for the translation of subtrees while using these features in combination with the baseline. Also in their experiments *soft-constraint decoding* (matching labels in a soft as opposed to strict way) is required to get the best results (Chiang, 2010).[8] These observations seem to support the conclusion that while syntax can be very helpful, enforcing it too strictly in translation is often a losing strategy. In rule extraction, strict syntactic constraints risk that good rules cannot be extracted. In decoding, such constraints risk that good translations are blocked. In both cases, formation of certain correct translations may become impossible, hence

[7]The problem that without proper measures, estimating a (synchronous) tree-substitution grammar with the EM algorithm leads to extreme overfitting is well known from both parsing and translation research (Prescher et al., 2004; DeNero et al., 2006).Solutions to this problem that use the empirical reusability of fragments as a criterion for acceptability of fragments have been proposed (Mylonakis and Sima'an, 2010; Mylonakis, 2012; Sangati and Zuidema, 2011). These approaches avoid the need of resorting to hard coded bias in the form of hard constraints on the fragment size (Bod, 1998) or priors as part of a Bayesian framework (Post and Gildea, 2009; Cohn et al., 2010), instead using only the data itself to arrive at strong models that generalize well and give good translation/parsing results.

[8]With this decoding method labels are interpreted as soft instead of hard constraints. This reduces the chance that valid translations get blocked because they contain rule substitutions with mismatching labels.

coverage is lost and performance drops. This is also another reason to look beyond syntax, at more language independent labels that can effectively model the mapping and reordering structure in the translation between two languages. We will come back to this point in chapter 4.

Syntax can be used to improve the alignment process without risking loss of coverage, by applying it as a soft constraint. In the work by (Burkett et al., 2010), the authors remark that recently there has been a lot of interest in improving alignments using syntactic information and improving tree transformations based on alignments. Yet these two things were not done simultaneously before in the literature. The authors fill this gap, by introducing a joint parsing and alignment scheme that builds upon monolingual parses for both language pairs, in combination with bilingual ITG tree pairs and a set of soft constraints in the form of synchronization features. Learning becomes intractable with exact methods, so the authors use Mean Field Inference to create computable approximations of the idealized functions they want to compute. The results show an increase in parsing and alignment performance separately, and a joint improvement in terms of better translation results.

The work by Riesa and Marcu (2010) gives yet another way to deeply yet softly involve syntax in the alignment model, without forcing the word alignments to satisfy syntactic constraints. Their work uses the structure of the best syntactic parse tree for the source sentence as the backbone for an efficient hierarchical search algorithm for the best alignment. As the nodes of the parse tree are visited in bottom up order, the algorithm builds up a hypergraph of alignments, storing alternatives at every node. Importantly, while the source parse tree is used to guide the search for word alignments, the explored word alignments are not required to satisfy syntactic constraints.

2.2.7 Word-based translation

Word alignment models model translation equivalence at the word level. As such they can be used directly to form a translation model, and in combination with a proper language model reasonable translation performance is achievable. This indeed was the initial proposal that prompted the renewed interest in statistical machine translation (Brown et al., 1988, 1993). Translation approaches that build translations directly based on the word level translation equivalence relations as induced by the alignment models are called *word-based* translation methods . Finding an effective search method (decoding algorithm) to deal with the large search space introduced by the IBM models can be challenging. Different decoding approaches to deal with this problem have been reported, we give a very short overview of historical approaches:

- Stack-based approach using priority queues(Berger et al., 1994, 1996a) and the original IBM models.
- Approach based on the $A*$ concept of multiple stacks (Wang and Waibel, 1997, 1998a).
- Approach based on dynamic programming (García-Varea et al., 1998; Nießen et al., 1998; Germann et al., 2001) with IBM Model 2 parameters.

- Approach based on monotone translation using HMM model alignments with dynamic programming based reordering as post-processing step (Tillmann et al., 1997a,b).

Germann et al. (2001) compares three decoding approaches for IBM Model 4: 1) Reimplementation of stack based decoder described in (Berger et al., 1996a), 2) greedy approach that improves upon initial solution 3) conversion of decoding problem into Integer Program (IP), solved by standard software. (Tillmann and Ney, 2000, 2003) introduce a Dynamic Programming based beam search decoder that works with the IBM Model 4 translation model.

2.3 Representing composed translation equivalence

While translation systems can in principle be built directly based on the alignment models described in the last section, there are good reasons to go beyond them and introduce composed TEUs as building blocks for translation. In the next subsection we will first motivate why this step is desirable. We will thereafter also give a formal description of translation equivalence in machine translation. This provides a formal basis for the work in this thesis in chapter 4 and 6. As part of this description we will give a definition of phrase pairs, which form the basis of phrase-based translation and are the non-hierarchical foundation of hierarchical phrase-based translation (HIERO) (Chiang, 2005). While phrase-based translation forms a rich and big subfield of machine translation by itself, here we will not go into detail on all its variants and improvements but will mainly describe it as a stepping stone to HIERO and its variants, as discussed in chapter 3.

2.3.1 Motivations for using composed translation equivalence units

In the last section we discussed alignment models, and we ended with a short discussion of how these models can be used to directly build word-based translation models that have already been applied in the past with reasonable amounts of success. But reconstructing frequent translation patterns entirely from word level translation equivalence relations is computationally expensive and error prone. When multi-word constructions (fragments[1]), are frequent enough to reoccur in data outside the training material, it is highly advantageous to extract and use these constructions directly when building translations. This insight was already discovered and tested early in the parsing community (Scha, 1990; Bod, 1992) but proved to be equally valuable for statistical machine translation (Och and Ney, 2004). Basic contiguous fragments known as phrase pairs or phrases are particularly effective for statistical translation, and are the reason why this type of translation is called *phrase-based* translation. The word alignment dictates which groups of words form *atomic* TEUs that may not be

broken up. But composing these atomic TEUs into larger fragments (phrase pairs) is allowed and has many advantages. Some of the main advantages are:

1. The translation of multiple words at once is much less ambiguous than the translation of the words individually. This is particularly clear for the translation of idiomatic constructions such as "*water bij de wijn doen | making a compromise*". Such idiomatic constructions could be nearly impossible to translate correctly without the use of larger fragments.

2. Phrase pairs can reduce translation (decoding) complexity, by reducing the search space. A large part of the complexity is caused by the reordering of fragments or words. When entire fragments are reordered as opposed to individual words, a much larger reordering can be achieved for a relatively much lower cost in terms of computational complexity.

3. The increased structure obtained from working with phrase pairs facilitates the usage of reordering features that can be important to improve the higher level reordering (Tillmann, 2004) and that would be hard to use when working at the level of individual words .

2.3.2 Translation equivalence in MT

In (Koehn et al., 2003), a TEU is a *phrase pair*: a pair of contiguous substrings of the source and target sentences such that the words on the one side align only with words on the other side (formal definitions next). The hierarchical phrase pairs (Chiang, 2005, 2007) are extracted by replacing one or more sub-phrase pairs, that are contained within a phrase pair, by pairs of linked variables. This defines a subsumption relation between hierarchical phrase pairs (Zhang et al., 2008a). Actual systems, e.g., (Koehn et al., 2003; Chiang, 2007) set an upper bound on length or the number of variables in the synchronous productions. For the purposes of our theoretical study, these practical limitations are irrelevant.

We give two definitions of translation equivalence for word alignments.[9] The first one makes no assumptions about the contiguity of TEUs, while the second does require them to be contiguous substrings on both sides (i.e., phrase pairs).

As usual, $\mathbf{f} = \mathbf{f}_1 ... \mathbf{f}_J$ and $\mathbf{e} = \mathbf{e}_I ... \mathbf{e}_I$ are source and target sentences respectively. Let $\mathbf{f}_j$ be the source word at position j in $\mathbf{f}$ and $\mathbf{e}_i$ be the target word at position i in $\mathbf{e}$. An alignment link $a \in \mathbf{a}$ in a word alignment $\mathbf{a}$ is a pair of positions $\langle j, i \rangle$ such that $1 \leq j \leq J$ and $1 \leq i \leq I$. For the sake of brevity, we will often talk about alignments without explicitly mentioning the associated source and target words, knowing that these can be readily obtained from the pair of positions and the sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$.

[9] Unaligned words tend to complicate the formalization unnecessarily. As usual we also require that unaligned words must first be grouped with aligned words adjacent to them before translation equivalence is defined for an alignment. This standard strategy allows us to informally discuss unaligned words in the following without loss of generality.

Given a subset $\mathbf{a}' \subseteq \mathbf{a}$ we define $words_{\mathbf{f}}(\mathbf{a}') = \{\mathbf{f}_j \mid \exists x : \langle j, x\rangle \in \mathbf{a}'\}$ and $words_{\mathbf{e}}(\mathbf{a}') = \{\mathbf{e}_i \mid \exists x : \langle x, i\rangle \in \mathbf{a}'\}$.

Now we consider triples $(\mathbf{f}', \mathbf{e}', \mathbf{a}')$ such that $\mathbf{a}' \subseteq \mathbf{a}$, $\mathbf{f}' = words_{\mathbf{f}}(\mathbf{a}')$ and $\mathbf{e}' = words_{\mathbf{e}}(\mathbf{a}')$. We define the *translation equivalence units (*TEUs*)* in the set $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ as follows:

Definition 2.3.1. $(\mathbf{f}', \mathbf{e}', \mathbf{a}') \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ *iff for all* $\langle j, i\rangle \in \mathbf{a}'$ *holds:* $\langle j, i\rangle \in \mathbf{a}' \Rightarrow$ *(for all x, if* $\langle j, x\rangle \in \mathbf{a}$ *then* $\langle j, x\rangle \in \mathbf{a}'$*)* $\wedge$ *(for all x, if* $\langle x, i\rangle \in \mathbf{a}$ *then* $\langle x, i\rangle \in \mathbf{a}'$*)*

In other words, if some alignment link involving source position j or target position i is included in $\mathbf{a}'$, then all alignments in $\mathbf{a}$ containing that position are in $\mathbf{a}'$ as well. This definition allows a variety of complex word alignments such as the so-called *Cross-serial Discontiguous Translation Units* and *Bonbons* (Søgaard and Wu, 2009).

We also define the subsumption relation (partial order) $<_{\mathbf{a}}$ as follows:

Definition 2.3.2. *A* TEU $u_2 = (\mathbf{f_2}, \mathbf{e_2}, \mathbf{a_2})$ *subsumes (*$<_{\mathbf{a}}$*) a* TEU $u_1 = (\mathbf{f_1}, \mathbf{e_1}, \mathbf{a_1})$ *iff* $\mathbf{a_1} \subset \mathbf{a_2}$*. The subsumption order will be represented by* $u_1 <_{\mathbf{a}} u_2$*.*

Based on the subsumption relation we can partition $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ into two disjoint sets: atomic $\mathbf{TE_{Atom}}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ and composed $\mathbf{TE_{Comp}}(\mathbf{f}, \mathbf{e}, \mathbf{a})$.

Definition 2.3.3. $u_1 \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ *is* atomic *iff* $\nexists\, u_2 \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a}) : (u_2 <_{\mathbf{a}} u_1)$.

Now the set $\mathbf{TE_{Atom}}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ is simply the set of all atomic TEUs, and the set of composed TEUs $\mathbf{TE_{Comp}}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ is the set of all TEUs minus the set of atomic TEUs: $\mathbf{TE_{Comp}}(\mathbf{f}, \mathbf{e}, \mathbf{a}) = (\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a}) \setminus \mathbf{TE_{Atom}}(\mathbf{f}, \mathbf{e}, \mathbf{a}))$.

Based on the general definition of translation equivalence, we can now give a more restricted definition that allows only contiguous TEUs (phrase pairs):

Definition 2.3.4. $(\mathbf{f}', \mathbf{e}', \mathbf{a}')$ *constitutes a* contiguous translation equivalence unit *iff:*

1. $(\mathbf{f}', \mathbf{e}', \mathbf{a}') \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ *and*
2. *Both* $\mathbf{f}'$ *and* $\mathbf{e}'$ *are contiguous substrings of* $\mathbf{f}$ *and* $\mathbf{e}$ *respectively.*

This set of TEUs is the unrestricted set of phrase pairs known from phrase-based machine translation (Koehn et al., 2003). The relation $<_{\mathbf{a}}$ as well as the division into atomic and composed TEUs can straightforwardly be adapted to contiguous translation equivalents.

In Figure 2.5 we see an example of an alignment with only contiguous TEUs, while in Figure 2.6 another alignment which yields both contiguous and discontiguous TEUs is shown. For actual translation, we restrict ourselves to phrase pairs (contiguous

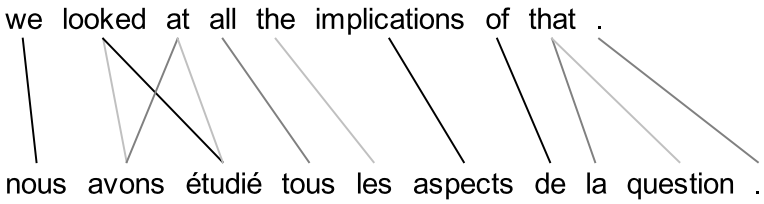


Figure 2.5: Alignment with only contiguous TEUs (example from Hansards English–French).

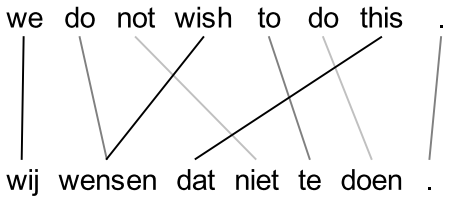


Figure 2.6: Alignment with both contiguous and discontiguous TEUs (example from Europarl English–Dutch).

TEUs) and its generalization *hierarchical phrase pairs*. The latter category constitutes phrase pairs with gaps, created by taking a composed phrase pair and replacing some smaller subsumed phrase pairs in it with variable pairs. While such *hierarchical phrase pairs* allow for hierarchical reordering, they are just contiguous TEUs with variable pairs, and therefore much weaker than actual discontiguous TEUs which allow for patterns with variables/gaps on the source/target side only without complement on the other side. The work by (Kaeshammer, 2013) describes *synchronous linear context-free rewriting systems*, as an extension of synchronous context free grammars (basically an extension of *linear context-free rewriting systems* (Vijay-Shanker et al., 1987) to the synchronous case), which can deal with discontiguous TEUs at the price of much higher computational complexity.

2.3.3 Phrase-Based Translation

Alignment Templates

The main foundations for modern phrase-based translation come together in the work on the alignment template approach to Statistical Machine Translation (Och and Ney, 2004). Some of the important components used in this approach were already published earlier (Wang and Waibel, 1998b; Och and Weber, 1998). But this longer work extends these and presents all main components of current phrase-based translation in a very structured, complete and clear way. The original alignment template approach uses bilingual word classes (Och, 1999) inside its rules, substituting those for the original words, to achieve better generalization and also to save memory

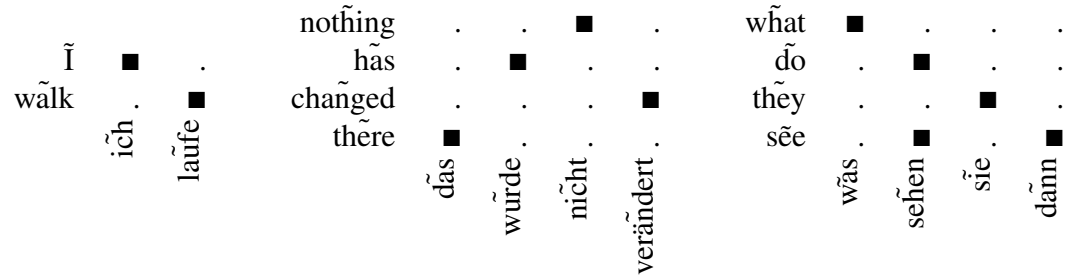


Figure 2.7: Examples of alignment templates. Following (Och and Ney, 2004) we use $\tilde{W}$ to indicate the word class assigned to word W.

by reducing the amount of rules. Another important property of this approach is to represent the internal word alignments as part of the rules. Rules in the alignment template model (alignment templates) are triples $\langle \tilde{\mathbf{f}}, \tilde{\mathbf{e}}, \tilde{\mathbf{a}} \rangle$ consisting of a source class sequence $\tilde{\mathbf{f}}$, a target class sequence $\tilde{\mathbf{e}}$ and the alignment between them $\tilde{\mathbf{a}}$. Figure 2.7 shows examples of alignment templates.

The alignment template model is of historical interest, and the choice to preserve word alignments inside rules is relevant in the context of this thesis which advocates applying word alignment information through bilingual reordering labels to improve word order, as discussed in Chapter 5. Nevertheless, the alignment template model soon after publication became dominated by the very similar *phrase-based translation model*. For the sake of brevity here we omit further details concerning the alignment template model, and instead next directly discuss the phrase-based model in some more depth.

Phrase-based SMT

The alignment template model emphasizes the use of word alignments as a central element of the alignment templates as well as the use of word classes instead of words inside the templates. The phrase-based model as proposed by (Koehn et al., 2003) and (Zens et al., 2002) simplifies the model by working with phrase pairs that use actual words instead of word classes and that don't keep internal word alignments explicitly associated with them.[10] This leads to a simplified generative process with three steps, without the preprocessing step of mapping to word classes and without the word selection step in the end, as shown in Figure 2.8. These three steps form a generative model, where each step involves decisions that require latent variables to be modeled probabilistically. In the first step the source sequence of bilingual word classes is split into K contiguous phrases based on the value of the *segmentation*

[10]The internal alignments are used for word selection in the last step of the alignment template model, but since the phrase based model works with actual words, this last step is not necessary. In the phrase-based model, alignments are used for computing *lexical weights* which are similar to the word selection probabilities in the alignment template model. However, lexical weights are computed already during phrase extraction and then stored with the phrase pairs, so that the alignments are no longer needed inside the extracted phrase pairs.

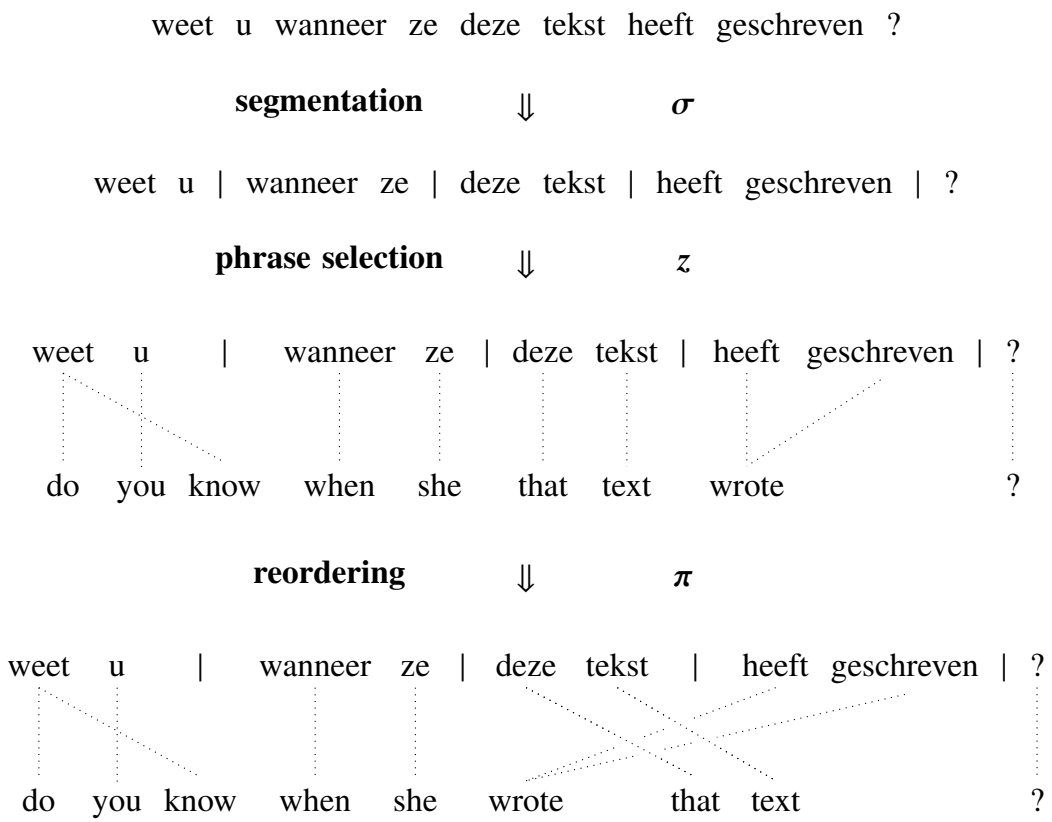


Figure 2.8: The phrase-based model, generation steps: 1) segmentation 2) phrase selection 3) reordering. Note that in contrast to the alignment template model, actual words rather than word classes are used throughout the model, which makes a final word selection step unnecessary. Also alignments (shown as dotted lines for clarity) are not retained within phrases.

variable $\boldsymbol{\sigma} \in \Sigma(f)$, with $\Sigma(f)$ being the set of all possible segmentations. In the second step for each segment a phrase pair $z_k = \langle f_k, e_k \rangle$ is selected, according to the *selection* variable $\boldsymbol{z} = z_1^K$. Finally, in the last step the selected phrase pairs are reordered as a permutation π_1^K of the phrase positions $1 \ldots K$, based on the *reordering* variable $\boldsymbol{\pi} = pi_1^K$. We can then describe the conditional translation probability of an aligned sentence pair with the following generative model:

$$p(\boldsymbol{e}|\boldsymbol{f}) = \sum_{\boldsymbol{\sigma} \in \Sigma(\boldsymbol{f})} \overbrace{p(\boldsymbol{\sigma}|\boldsymbol{f})}^{\text{segmentation}} \overbrace{p(\boldsymbol{z}|\boldsymbol{\sigma}, \boldsymbol{f})}^{\text{phrase selection}} \overbrace{p(\boldsymbol{\pi}|\boldsymbol{z}, \boldsymbol{\sigma}, \boldsymbol{f})}^{\text{reordering}} \tag{2.21}$$

Assuming all of the phrase pairs involved in the translation of $\boldsymbol{f}$ are independently applied gives:

$$p(\boldsymbol{z} = z_1^K|\boldsymbol{\sigma}, \boldsymbol{f}) = \prod_{k=1}^{K} p(z_k = \langle f_k, e_k \rangle | f_k) \tag{2.22}$$

Every combination of latent variables $\langle \boldsymbol{\sigma}, \boldsymbol{\pi}, \boldsymbol{f} \rangle$ yields a different derivation $\mathbf{d}$, whereby typically there are many different derivations yielding the same translation. Standardly, for complexity reasons, most models used make the assumption that the probability $P(\boldsymbol{e}|\boldsymbol{f})$ can be optimized through as single best derivation as follows:

$$\arg\max_{\boldsymbol{e}} P(\boldsymbol{e}|\boldsymbol{f}) = \arg\max_{\boldsymbol{e}} \sum_{\mathbf{d}\in G} P(\boldsymbol{e}, \mathbf{d} \mid \boldsymbol{f}) \quad (2.23)$$

$$\approx \arg\max_{\mathbf{d}\in G} P(\boldsymbol{e}, \mathbf{d} \mid \boldsymbol{f}) \quad (2.24)$$

Given a derivation $\mathbf{d}$, most existing phrase-based and hierarchical phrase-based models approximate the derivation probability through a linear interpolation of a finite set of feature functions ($\Phi(\mathbf{d})$) of the derivation $\mathbf{d}$. In this approximation, they mostly work with local feature functions ϕ_i of individual productions, the target side yield string t of $\mathbf{d}$ (target language model features) and other heuristic features:

$$\arg\max_{\mathbf{d}\in G} P(\mathbf{e}, \mathbf{d} \mid \mathbf{f}) \approx \arg\max_{\mathbf{d}\in G} \sum_{i=1}^{|\Phi(\mathbf{d})|} \lambda_i \times \phi_i \quad (2.25)$$

Features Phrase-Based Model The phrase-based model uses a range of features, of which many are highly similar or even identical to those used by the alignment template model. These features are typically used in a log-linear (discriminative) model as earlier discussed in this chapter, in subsection 2.1.2. We give an overview of the most common used features:

- **Phrase Translation probabilities**: these are estimated by relative frequency estimation. The probabilities are often computed in two directions: $p(\bar{e}|\bar{f})$ and $p(\bar{f}|\bar{e})$ [11] for source phrases $\bar{f}$ and target phrases $\bar{e}$ (2.26). The full scores are computed by multiplying these scores over all selected phrase pairs. In the actual implementation, this amounts to a summation of logs of probabilities, which allows for efficient computation. Finally the log of this product is taken to get the feature scores to be used in a log linear model (2.27):

$$p(\bar{e}|\bar{f}) = \frac{count(\langle \bar{f}, \bar{e} \rangle)}{\sum_{\bar{e}'} count(\langle \bar{f}, \bar{e}' \rangle)} \qquad p(\bar{f}|\bar{e}) = \frac{count(\langle \bar{f}, \bar{e} \rangle)}{\sum_{\bar{f}'} count(\langle \bar{f}', \bar{e} \rangle)} \quad (2.26)$$

$$\phi^{\mathbf{f}|\mathbf{e}}_{PHR} = \log \prod_{k=1}^{K} p(\bar{f}_k|\bar{e}_k) \qquad \phi^{\mathbf{e}|\mathbf{f}}_{PHR} = \log \prod_{k=1}^{K} p(\bar{e}_k|\bar{f}_k) \quad (2.27)$$

  In the above formulas *count* is the count of the rules in the parallel (training) corpus.

[11] In the original work by (Koehn et al., 2003), only the source given target probabilities $p(f|e)$ are used as, derived from the noisy channel model. But when working with a log-linear model both directions can be used easily. And since the features are based on heuristic estimation rather than proper statistical estimation as we discussed before, two imperfect features for both directions are often better than just one of them for one direction.

- **Reordering model**: the phrase-based model uses a phrase-alignment feature that is known as a **distance-based reordering model**. For each pair of consecutive (reordered) phrase pairs on the target side, the distance between the first source word of the current phrase pair and the last source word of the previous phrase pair is computed:

$$d(\text{start}_i - end_{i-1} - 1) \tag{2.28}$$

The phrase-based model then uses this distortion probability distribution $d(\cdot)$ to assign a probability to each reordered phrase pair separately. These probabilities are then either multiplied as factors of the total translation probability (Koehn et al., 2003) or alternatively used as features in a log-linear model. The distribution for $d(\cdot)$ can be computed as part of a phrase-based joint probability model (Marcu and Wong, 2002) or alternatively it can be modeled as a simpler parametric distribution $d(x) = \alpha^{|x|}$ with an appropriate $\alpha \in [0.1]$.

A more context sensitive **phrase reordering model** is proposed in (Tillmann, 2004) and also used in (Koehn et al., 2005). This model assumes the target sequence is generated from left to right, using blocks (phrase pairs) whose sources need not be consecutive but can be reordered. This means the i^{th} block has an $(i-1)^{th}$ predecessor block, with respect to which the orientation of the current block can be determined. This orientation can be one of either *monotone*, *swap* or *discontinuous*. This allows translation to be modeled as generation of a sequence of blocks with orientations. For this, the model uses phrase pair specific orientation probabilities for blocks with monotone or swap orientation. These orientation probabilities are estimated with relative frequency estimation from the aligned training corpus, and are used to extend the basic product of individual phrase weights and target side language model probabilities. Because the orientation probabilities are phrase pair specific and therefore depend on the lexical context of phrase pairs, the proposed model is also known as a **lexicalized** reordering model. For blocks with discontinuous orientation, no special reordering scoring is applied. This method shows significant improvement over 1) allowing no reordering or 2) using only the language model to score reordering decisions.

A **hierarchical phrase reordering model** extending to the previous phrase reordering model is proposed by Galley and Manning (2008). This model determines the orientation for each translated block by seeing if an adjacent *hierarchical block* can be found that precedes (*monotone*) or follows (*swap*) the block on the source side. A hierarchical block is formed by merging phrase pairs that translate a consecutive span of directly preceding target words into a bigger phrase pair. The fall-back orientation *discontinuous* only needs to be used in the rare case that no such block can be found. During decoding, an instance of the shift-reduce algorithm is used to efficiently look for possible hierarchical phrases that yield the reordering obtained by the translation, allowing quadratic running time. Based on this algorithm, the new reordering method can be readily

implemented in a left-to-right phrase-based translation system such as Moses. In the authors experiments it gave significant improvements in translation accuracy on the Chinese–English and Arabic–English translation tasks. The authors compare their algorithm with (Tillmann, 2004) and also with the simpler **word-based orientation model**, that is standard in Moses (Koehn et al., 2007), which determines block orientations by looking at the word alignment positions of the source word that directly precedes the block and the one that directly follows it.

- **Lexical Weighting**: for a phrase pair $\langle f, e\rangle$, lexical probabilities are computed to smooth the phrase probabilities. First lexical weights $w(f|e)$ for word translations are computed with simple relative frequency estimation based on the aligned word pairs taken from the word-aligned sentence pairs (2.29). With the help of the word alignments a for a phrase pair, these lexical weights are used to compute for every target word the average lexical weight over its aligned source words, and these averages are multiplied to get the total lexical weight for the aligned phrase pair $p_w(\bar{f}|\bar{e}, a)$ (2.30). When there are multiple alternative alignments for a phrase pair, the score for the alignment that gives the highest total lexical weight $\widehat{p_w}(\bar{f}|\bar{e})$ is taken for it (2.31). Finally, to get the total lexical weight $\phi^{\mathbf{f}|\mathbf{e}}_{LEX}$ of a translation, all the lexical weights for the phrases used in the translation are multiplied. When used as a feature inside a log-linear model the log of this total weight is taken (2.32). As is the case for phrase weights, lexical weights are often computed in two directions. This means an analogously computed feature $\phi^{\mathbf{e}|\mathbf{f}}_{LEX}$ is typically also added.

$$w(f|e) = \frac{count(f, e)}{\sum_{f'} count(f', e)} \tag{2.29}$$

$$p_w(\bar{f}|\bar{e}, a) = \prod_{i=1}^{n} \frac{1}{|\{j|(i, j) \in a\}|} \sum_{\langle i,j\rangle \in a} w(f_i|e_j) \tag{2.30}$$

$$\widehat{p_w}(\bar{f}|\bar{e}) = \max_a p_w(\bar{f}|\bar{e}, a) \tag{2.31}$$

$$\phi^{\mathbf{f}|\mathbf{e}}_{LEX} = \log \prod_{k=1}^{K} \widehat{p_w}(\bar{f}|\bar{e}) \tag{2.32}$$

- **Language Model**: strong language models play a major role in producing high quality translation output. The Markovian n-gram language models score the produced target output across phrase boundaries, which is crucial for producing fluent output. Because of the conditioning on words across state boundaries, the integration of the language model in the decoding process adds a lot of complexity, because it necessitates the search hypotheses to keep the state of the language model. While originally a trigram language model was used (Koehn

et al., 2003), nowadays it is best practice to use at least a 4-gram if not 5-gram language model and smooth with modified Knesser-Ney discounting. A description of this common setup and an overview of language models frequently used in MT is found in (Chen and Goodman, 1998).

- **Word and phrase penalties**: In addition to the word penalty that was introduced already for the alignment template model, a *phrase penalty* can be used to count the number of used phrases, and thereby allow the tuner to indirectly learn a preference for the relative length (in source words) of phrase pairs used in constructing translations.

Search Finding the best translation is implemented as *beam search* which means breadth-first search with pruning. This search can be conceptualized as a graph search in which three actions are possible : 0) creating an initial (empty) hypothesis 1) extending a hypothesis, by applying a phrase pair to translate some untranslated words, creating a new extended hypothesis 2) finishing the translation of a sentence.

For each of these actions the relevant features are applied as far as possible. Furthermore, a function that computes an admissible heuristic of the *future cost* of finishing incomplete translations is used. This function is crucial in reducing the amount of serious search errors due to discarding the wrong hypotheses during pruning. The chance of search errors is decreased by keeping different search stacks, one stack for every number of source words n_s with $n_s = 1 \ldots J$ that has been translated. Efficient search is then furthermore facilitated by different types of pruning:

1. **Observation pruning**: only the most likely words are selected for template instantiation (Tillmann and Ney, 2000).
2. **Histogram pruning**: a maximum number of n hypotheses is kept on every stack (Steinbiss et al., 1994).
3. **Threshold pruning**: a fixed threshold α is kept for every stack, and any hypothesis which has a score that is α times worse than the best one in the stack is pruned out.

A last key component of efficient search is *hypothesis recombination*. Multiple hypotheses can be recombined, keeping only the best scoring one, when they are identical or indistinguishable by the language and translation models.[12] This condition guarantees that no extension of one of the inferior hypotheses can ever beat the same extension of the best hypothesis in the future, even though the hypotheses might not have the exact same target output.

[12]Hypothesis recombination assumes a search for the single best derivation. In a Minimum Bayes-Risk (MBR) decoding setting, where some form of summation over derivations that yield the same output is done, these sufficient conditions for hypothesis recombination no longer hold.

2.3.4 Feature Weights Training

Learning effective weights for the features used in Statistical Machine Translation is a crucial component in the creation of effective translation systems. Early phrase-based translation systems used pure probabilistic models with only a tiny amount of core features. The move to more general log-linear models with more features, trained in a discriminative way, was mainly the introduction of effective feature weights training methods such as Minimum Error Rate Training (MERT) (Och, 2003) and maximum entropy models (Berger et al., 1996b; Och and Ney, 2002) that enabled the effective tuning of a more than tiny, but still very modest number of features. The ability to effectively use more substantial numbers of features has only been obtained recently with the adoption of large scale discriminative learning algorithms such as the margin infused relaxed algorithm (MIRA) (Crammer and Singer, 2003) and in particular its general availability for machine translation, implemented as a batch-tuning variant (Cherry and Foster, 2012).

Although feature weights learning has a crucial supporting role for the experiments in this thesis, these methods are used as a finished component and the thesis makes no further contribution to their development.[13] With these considerations in mind, and only a limited amount of space available, we will next very briefly discuss only the most popular tuning methods MERT and MIRA. The reader is then referred to Appendix A.2 for a deeper and much more complete overview of these methods.

MERT MERT (Och, 2003) is a discriminative feature weights training method that directly minimizes the translation error. This is done by repeatedly performing a line search, along a line in the N-dimensional search space of feature weight values. The weight optimization is done using N-best lists of best translations produced on a development set, and the new N-best list is incrementally merged with older N-best lists for earlier iterations during optimizations. This incremental merging is done to guarantee stable conversion towards a local optimum over all translations that are seen so far, and avoid optimization for only translations seen in the current iteration. MERT is a greedy optimization algorithm, essentially a form of *hill-climbing* (Russell and Norvig, 2003). The fact that MERT in its core only optimizes the weights along a line (i.e. one dimension at a time) makes it inherently unable to scale up to a large number of features (beyond 15 features problems start, and tuning more than 30 features is practically impossible).

MIRA The Margin Infused Relaxed Algorithm (MIRA) was first used by (Watanabe et al., 2007) and later refined by (Chiang et al., 2008, 2009; Chiang, 2012). MIRA builds upon on the two fundamental concepts of *cost* (or loss) and *margin*. The cost

[13] The experiments in this thesis use an approach called *soft constraints*, which will be discussed in the next chapter. The form of this approach adopted in the experiments is only feasible thanks to the availability of reliable, scalable feature weights training methods. This dependence makes it relevant to be familiar with the principles behind these methods, and have some understanding of their strengths and limitations.

of choosing a translation given some oracle translation and a reference is defined as the difference between their scores on some evaluation metric, given the reference. The margin, or distance in model scores, between a candidate translation and oracle translation is defined as the summed differences of their feature values multiplied by their feature weights. Conceptually, the aim of MIRA is to separate all pairs of reachable hypotheses $\langle h_1, h_2 \rangle$ in such a way that the margin between them is at least as big as the cost of choosing h_1 in place of h_2. Such a separation guarantees that not just one best hypothesis get the highest weight, as in MERT, but also globally the hypothesis space is properly structured such that lower quality hypotheses will have correspondingly lower model scores. This leads to a much higher stability of the optimization, and allows this method to scale to millions of features.

Summary and outlook

In this chapter we covered the basics of statistical machine translation, concentrating on word alignment, translation equivalence and phrase-based models. In the next chapter we continue by looking at hierarchical SMT and synchronous grammars. We will first discuss the basic methods and formalisms for hierarchical SMT. We will then zoom in on work that attempts to improve composition and reordering in hierarchical SMT, emphasizing on methods that use a form of rule labeling to do so.

3

Background Hierarchical SMT and Synchronous Grammars

The human language interpretation process has a strong preference for recognizing sentences, phrases and patterns that have occurred before. Structures and interpretations which have occurred frequently are preferred above alternatives which have not or rarely been experienced before. All lexical elements, syntactic structures and "constructions" which the language user has ever encountered, and their frequency of occurrence, can have an influence on the processing of new input. The amount of information that is necessary for a realistic performance-model is therefore much larger than the grammars that we are used to. The language-experience of an adult language user consists of a large number of utterances. And every utterance contains a multitude of constructions: not only the whole sentence, and all its constituents, but also all patterns that we can abstract from these by substituting "free variables" for lexical elements or complex constituents.

– Remko Scha, *Language theory and language technology; competence and performance.*

At the end of the previous chapter we discussed phrase-based translation, which has yielded big improvements over word-based models by allowing the reuse of consecutive bilingual patterns longer than word pairs. While successful for many language pairs, this model is still suboptimal when modeling the translation of language pairs with big differences in word order. In this chapter we will look at hierarchical translation methods that strive to overcome some of the major shortcomings of the phrase-based method in this respect.

In the translation of new sentences from previously seen phrases, phrase-based translation builds upon the general framework of machine learning. A crucial assumption of this framework is that the training data must be representative for future test data. Here, the bilingual training set must be representative for the bilingual patterns that are observed in the future. In particular, new data should contain a

sufficient amount of earlier observed phrase pairs bigger than word pairs. When both languages of a language pair come from the same language family, such as for French-English, this is a valid assumption. But for language pairs involving languages with a lot of morphology, such as Arabic or Hebrew, this assumption becomes problematic. Such language pairs may benefit from approaches that stay closer to the original word-based translation formalism in combination with more flexible constraints, as illustrated by (Ittycheriah and Roukos, 2007). But rich morphology is not the only challenge for *phrase-based* translation. Language pairs that involve a substantial amount of non-local reordering, such as Chinese–English and German–English[1] can also be hard to grasp with phrase-based translation. Language pairs involving a language with a more free word order on the source side, such as German or Dutch also reduce the success of phrase-based translation. The main reasons for these word order and reordering related problems are:

1. Modeling long-distance reordering with phrase-based models has a high computational complexity.

2. Phrase-based models in their basic form fail to embed non-local reordering decisions in a wider context. While some remedies to this problem are available (Tillmann, 2004), in general phrase-based structure is not designed to model the global, hierarchical reordering that certain language pairs require.

3. A more free word order of the language on the source side means that there is more variation in translation input patterns, decreasing the chance that larger phrase pairs are reusable. This means that the model has to rely more on smaller phrase pairs, which compromises translation quality.

These problems, which highlight some of the main shortcomings of phrase-based translation motivated hierarchical phrase-based translation (Chiang, 2005, 2007) called HIERO as well as syntactic translation methods such as string-to-tree translation (Galley et al., 2004), tree-to-string translation (Huang et al., 2006) and forest-to-string translation (Mi et al., 2008a). In this chapter we discuss hierarchical translation methods, focusing on HIERO and labeled variants of it. We start by discussing synchronous context-free grammars (SCFGs) and then continue with hierarchical SMT (HIERO) and syntax-augmented machine translation (SAMT) (Zollmann and Venugopal, 2006). Following the discussion of SAMT, we review a range of other labeling methods for HIERO. We end the chapter with some other important techniques that have been used in combination with HIERO, obtaining large gains in the quality of the word order of the produced translations.

[1] In fact, German–English and English–German have both a rich morphology, as well as non-local reordering, and furthermore these two elements interact. This makes these language pairs particularly hard to work with, as they essentially demand an approach that considers morphology and reordering together. Our approach discussed in chapter 5, considers word order but ignores morphology, which while better than the baseline, remains suboptimal for this language pair.

3.1 Synchronous Context-free Grammars

Statistical parsing builds on (statistical) CFGs and stronger grammar formalisms such as Tree-Substitution Grammars (Bod, 1992; Sima'an et al., 1994; Bod, 2003; Bod et al., 2003) and Tree-Adjoining Grammars (Joshi et al., 1975; Joshi, 1985; Joshi and Schabes, 1997). These monolingual syntactic approaches can be extended to a bilingual setting. This allows bilingual strings to be generated or recognized and enables translation and tree-transduction based on grammars. For context free grammars, different proposals for extensions to the bilingual setting were made early in the literature. These include syntax directed translation (Aho and Ullman, 1969) and syntax directed transduction (Lewis and Stearns, 1968). Another influential extension is inversion transduction grammars (ITGs) (Wu, 1997). Other more recently proposed extensions are Multitext Grammars (Melamed, 2003) and the earlier mentioned synchronous linear context-free rewriting systems (Kaeshammer, 2013). But the stronger monolingual grammar formalisms can also be extended to the bilingual setting. Synchronous Tree Adjoining Grammars (Shieber and Schabes, 1990) extend Tree Adjoining Grammars to allow bilingual parsing and Synchronous Tree-Substitution Grammars (Poutsma, 2000; Eisner, 2003) produce string pairs based on syntactic subtrees which gives the complete source and target trees more freedom to diverge in terms of structure.

SCFGs (Aho and Ullman, 1969) are an attractive formalism for the modeling of translation. With their recursive translation rules and thanks to nonterminals, SCFGs naturally capture compositional translation equivalence relations and long distance reordering. Formally a SCFG $\mathbf{G}$ is defined as the tuple $\langle N, E, F, R, S \rangle$, with:

- N: A finite set of non-terminals.
- $S \in N$: the start symbol.
- E/F: Finite sets of words for the source/target language.
- R: A finite set of rewrite rules.

The rules $\mathbf{r_i} \in R$ are of the form:

$$\mathbf{r_i} : L_s || L_t \rightarrow < \gamma_i, \alpha_i, \theta_i >$$

Here $L_s || L_t$ is a pair of linked source and target nonterminals (possibly the same) [2] forming the left-hand-side of the rule. Next γ_i and α_i are sequences of both nonterminal

[2]The fact that general SCFGs allow different non-terminals on the source and target side is often glossed over in formal descriptions in the SMT literature. While concatenation of the source and target nonterminal into one unit (since the two are anyway linked) may be used during implementation, at least from the point of view of probability estimation it is important to note that it may make a lot of difference which labels are added on what side, and copying the same label to both sides may not always be a good default. Another argument to keep the two apart is that this may help in making more effective rule-bins when pruning partial derivations during decoding.

$S \rightarrow das\ X^{\boxed{1}} \,||\, X^{\boxed{1}}\ there$ NP-NN $\rightarrow nicht \,||\, nothing$

$X \rightarrow VBZ^{\boxed{1}}$ NP-NN$^{\boxed{2}}\ VBN^{\boxed{3}} \,||$ NP-NN$^{\boxed{2}}\ VBZ^{\boxed{1}}\ VBN^{\boxed{3}}$ $VBZ \rightarrow wurde \,||\, has$

$VBN \rightarrow verändert \,||\, changed$

Figure 3.1: A SCFG rule set for the inversion of *VBZ* and NP-NN in German to English translation of the sentence "das wurde nicht verändert || nothing has changed there".

and terminal symbols, while θ_i is a bijective function[3] from nonterminals in γ_i to nonterminals in α_i. The type of SCFGs that is used in translation contains nonterminal tokens chosen from a finite set of nonterminal types, and word tokens taken from a finite set of word types.

Weighted SCFGs Hierarchical Statistical Machine Translation works with *weighted* SCFGs[4]. Weighted Synchronous Context-Free Grammars add for every rule $\mathbf{r_i}$ a vector of features $\vec{\phi_i}$, so that the i^{th} rule of such a grammar is denoted:

$$\mathbf{r_i} : L_s || L_t \rightarrow < \gamma_i, \alpha_i, \theta_i, \vec{\phi_i} >$$

$\vec{\phi_i}$ quantifies in particular the phrase- and lexical-probabilities of translating between γ_i and α_i but also counts, binary features and any other informative features that can somehow be expressed as a function of the rule or the translation context. The feature vectors make the grammar into a log-linear model with weights that can be discriminatively tuned. Unless otherwise stated, when we use the term SCFGs in this thesis we will refer to the *weighted* version.

Binary SCFGs

The *rank* of a CFG/SCFG rule is defined as the number of nonterminals / nonterminal pairs it contains on its right-hand side. The rank of a CFG/SCFG as a whole is defined as the maximum rank amongst the rules it contains. For a CFG it is always possible to convert it into a (weakly) equivalent CFG with rank two or less (Chomsky Normal form). Also any SCFG of rank 3 can always be converted into a SCFG of rank 2 by binarization. For example, in Figure 3.1 we show such a SCFG for German to English translation. When we binarize this, we get the SCFG displayed in Figure 3.2.

[3] A bijective function is a function giving an exact pairing of the elements of two sets. Every element of one set is paired with exactly one element of the other set, and every element of the other set is paired with exactly one element of the first set. There are no unpaired elements. In formal mathematical terms, a bijective function $f : X \rightarrow Y$ is a one to one and onto mapping of a set X to a set Y.

[4] The term *weighted* is used to emphasize the fact that these grammars work with rules with a set of features with associated weights, corresponding to a log linear model. Using this more general form allows these models to be trained and applied in a *discriminative* way, using a very heterogeneous set of features, while automatically learning which features are useful for improving the translation quality.

$S \rightarrow das\ X^{\boxed{1}} \,||\, X^{\boxed{1}}\ there$ NP-NN $\rightarrow nicht \,||\, nothing$

$X \rightarrow$ X'$^{\boxed{1}}\ VBN^{\boxed{2}} \,||$ X'$^{\boxed{1}}\ VBN^{\boxed{2}}$ $VBZ \rightarrow wurde \,||\, has$

$X' \rightarrow VBZ^{\boxed{1}}$ NP-NN$^{\boxed{2}} \,||$ NP-NN$^{\boxed{2}}\ VBZ^{\boxed{1}}$ $VBN \rightarrow verändert \,||\, changed$

Figure 3.2: Binarized version of the rule set shown in Figure 3.1.

However, SCFG rules with rank ≥ 4 are not always binarizable. For example the rule

$$A \rightarrow \ A^{\boxed{1}}\, B^{\boxed{2}}\, C^{\boxed{3}}\, D^{\boxed{4}} \,||\, C^{\boxed{3}}\, A^{\boxed{1}}\, D^{\boxed{4}}\, B^{\boxed{2}} \tag{3.1}$$

cannot be binarized. And while this construction is just given as an example, in fact real data contains many non-binarizable constructions as we will see in chapter 6 despite earlier suggestions that such patterns should be very rare and that most rules should be binarizable (Wu, 1997; Huang et al., 2009). While not all valid SCFG rules are binarizable, restriction to well chosen binary SCFGs offers an adequate balance between flexibility and complexity constraints. A particularly successful grammar formalism that has been applied in its original form and variations upon it for many applications and language pairs is introduced next.

Inversion transduction grammars

The usability of stochastic binarizable synchronous grammars for many tasks including segmentation, word alignment and bracket annotation was established by (Wu, 1995, 1997). This work introduced a particular type of binarizable synchronous grammars called *inversion transduction grammars* (ITGs) . Later it was shown that ITG also forms a strong basis for adequate and efficient translation of language pairs with considerable word order differences, in particular when lexicalization of rules is applied (Chiang, 2005). In their most general form ITGs can contain any rules of the form

$$X \rightarrow \alpha_1 E_1^{\boxed{1}} \alpha_2 E_2^{\boxed{2}} \ldots \alpha_n E_n^{\boxed{m}} \alpha_{n+1} \,||\, \beta_1 F_1^{\boxed{1}} \beta_2 F_2^{\boxed{2}} \ldots \beta_n F_n^{\boxed{m}} \beta_{n+1}$$
$$X \rightarrow \alpha_1 E_1^{\boxed{1}} \alpha_2 E_2^{\boxed{2}} \ldots \alpha_n E_n^{\boxed{m}} \alpha_{n+1} \,||\, \beta_1 F_1^{\boxed{m}} \beta_2 F_2^{\boxed{n\text{-}1}} \ldots \beta_n F_n^{\boxed{1}} \beta_{n+1}$$

that rewrite some linked pair of nonterminals $X = E_q F_r$ as a list of source nonterminals $E_1 \ldots E_n$ that is linked pairwise with the list of target nonterminals $F_1 \ldots F_n$, such that the order of the former is either completely monotone or completely inverted relative to the latter. The symbols $\alpha_1 \ldots \alpha_{n+1}$ and $\beta_1 \ldots \beta_{n+1}$ indicate optional source/target strings of words. These words provide optional lexicalization of the rules. Importantly, (Wu, 1997) also shows that any ITG can be transformed into an equivalent normalized ITG that involves only binary rules and terminal rules producing pairs of (possibly empty) linked strings. Using a simplified notation where [ ] indicates straight reordering rules

and $\langle\ \rangle$ indicates inverted rules, such *normal form* ITG (NF-ITG) grammars have the form[5]:

$$A \rightarrow [BC] \qquad A \rightarrow \langle BC \rangle \qquad A \rightarrow e\ /\ f$$

SCFG Algorithms

While SCFGs are closely related to CFGs, the complexity of SCFG algorithms in general increases exponentially with the rank of the grammars. (Satta and Peserico, 2005) show that both parsing and decoding with arbitrary SCFGs is NP-hard. But in the case of binary SCFGs both tasks can still be done in polynomial time, which make this subset of general SCFGs particularly attractive for use in practical applications.

Parsing Parsing of SCFGs can be done based on an adapted version of the CYK-algorithm (Cocke, 1969; Younger, 1967; Kasami, 1965) in $O(n^6)$ time for binary SCFGs, with n the length of the source/target strings. The running time also scales linearly with the number of grammar rules $|\mathbf{G}|$.[6] Dyer (2010) proposes an alternative way to implement synchronous parsing, based on two consecutive rounds of monolingual parsing. First the source side is parsed with the translation grammar, producing an intermediate hypergraph of translations unconstrained by the target. Then this hypergraph is converted into a (highly specific) SCFG, which is used next to parse the target side, thereby creating a final hypergraph that is constrained by both source and target. This approach, while not improving the theoretical complexity of the algorithm, in practice substantially outperforms other synchronous parsing algorithms including the synchronous CYK-algorithm (Wu, 1997) and a more recent approach by Blunsom and Osborne (2008) derived from cube pruning (Huang and Chiang, 2007).

Decoding Decoding is the core task of Machine Translation, in which for a given source input sentence the most likely[7] target output is to be found. While bilingual parsing is already complex, the interaction with the language model during decoding makes the complexity of decoding even much higher.[8]

The basic complexity of general SCFG decoding with an integrated m-gram language model is $O(|w|^{3+2n(m-1)})$ (Huang et al., 2009). With $|w|$ the number of source words in the input, and n the rank of the SCFG. But Huang et al. (2005) show that using an optimization called the "hook trick" the complexity of SCFG parsing for SCFGs can

[5] Skipping here productions involving the empty token in one of the two stings, for brevity.

[6] But since this is a constant it is normally not part of the complexity formulas.

[7] Or highest scoring derivation, in the typical log-linear framework setup.

[8] While most decoders integrate the language model re-scoring with decoding process, it is also possible to first build an un-scored translation hypergraph and next intersect it with the language model, which is the approach advocated by (Dyer et al., 2010). Nevertheless, this neither fundamentally changes theoretical decoding complexity nor actually observed running time of this method as compared to other decoders that use integrated language models (Heafield, 2013).

be lowered to:

$$O(|w|^{3(m-1)+2(n+1)}) \tag{3.2}$$

Once again, in both cases the running time also scales linearly with the number of grammar rules $|\mathbf{G}|$.[6]. The hook trick re-factors the computation done during decoding, reducing the maximum number of interacting variables in any computation step. The authors remark however that there may be problems due to the interaction of the hook technique with pruning methods: "Building the chart items with hooks may take more time than it saves if many of the hooks are never combined with complete constituents due to aggressive pruning." This suggests that this theoretical complexity may be somewhat optimistic in practice. As (Huang et al., 2009) mention, when binarization of SCFG rules is possible, it is an important step to reduce the computational complexity of translation. Yet as mentioned before, binarization is not always possible and an complexity grows exponentially with the SCFG rank, even in the most optimistic complexity formula given in equation 3.2. This means that it may be better to explicitly restrict ourselves to only binary branching grammars beforehand, even if it may mean losing the ability to correctly represent certain reordering phenomena. This approach taken by the earlier discussed inversion transduction grammars (ITGs) is also followed by the derived Hiero grammar, which we will discuss in the next subsection.

Expectation Maximization The parameters of a synchronous grammar can be estimated given a corpus of sentence pairs using the EM algorithm. The expected counts of all rules must be computed for all rules of the SCFG given the current estimate. For this, a modified version of the Inside-Outside algorithm can be used (Baker, 1979; Lari and Young, 1990), which has the same complexity as SCFG parsing ($O(n^6)$ for binary SCFGs). Overfitting can be a problem for estimating SCFGs with the EM algorithm, as it is for (synchronous) tree-substitution grammars, while for the latter the problems are typically more severe. Using a variant of the EM algorithm called Cross-Validated EM (Mylonakis, 2012) can be one way to overcome these problems.[7]

3.2 Hierarchical Statistical Machine Translation

Hiero SCFGs (Chiang, 2005, 2007) allow only up to two (pairs of) nonterminals on the right-hand-side (RHS) of synchronous rules. The types of permissible Hiero rules are:

$$X \rightarrow \langle \alpha,\ \delta \rangle \tag{3.3}$$

$$X \rightarrow \langle \alpha\ X_{\boxed{1}}\ \gamma,\ \delta\ X_{\boxed{1}}\ \eta \rangle \tag{3.4}$$

$$X \rightarrow \langle \alpha\ X_{\boxed{1}}\ \beta\ X_{\boxed{2}}\ \gamma\ ,\ \delta\ X_{\boxed{1}}\ \zeta\ X_{\boxed{2}}\ \eta\ \rangle \tag{3.5}$$

$$X \rightarrow \langle \alpha\ X_{\boxed{1}}\ \beta\ X_{\boxed{2}}\ \gamma\ ,\ \delta\ X_{\boxed{2}}\ \zeta\ X_{\boxed{1}}\ \eta\ \rangle \tag{3.6}$$

Here $\alpha, \beta, \gamma, \delta, \zeta, \eta$ are terminal sequences. These sequences can be empty, except for β, since Hiero prohibits rules with nonterminals that are adjacent on the source side. Hiero also requires all rules to have at least one pair of aligned words. These extra constraints are intended to reduce the amount of spurious ambiguity. Equation 3.3 corresponds to a normal phrase pair, 3.4 to a rule with one gap and 3.5 and 3.6 to the monotone and inverting rules respectively.

Given an Hiero SCFG G, a source sentence $\mathbf{f}$ is translated into a target sentence $\mathbf{e}$ by one or more synchronous derivations $\mathbf{d}$, each of which is a finite sequence of well-formed substitutions of synchronous productions from G, see (Chiang, 2006, 2007). The goal of finding the most likely translation is then replaced by the somewhat simpler problem of finding the most likely derivation $\mathbf{d}$:

$$\arg\max_{\mathbf{d} \in G} P(\mathbf{e}, \mathbf{d} \mid \mathbf{f}) \tag{3.7}$$

We parse $\mathbf{f}$ with G so we limit the space of derivations to those that are licensed by G for $\mathbf{f}$, and so we have $P(\mathbf{e}, \mathbf{d} \mid \mathbf{f}) = P(\mathbf{d})$ ($\mathbf{e}$ is the sequence of target terminals generated by $\mathbf{d}$). Following Och and Ney (2002), a log-linear model over derivation $\mathbf{d}$ computes the probability of a derivation as a product of weighted features ϕ_i for that derivation. Apart from the language model feature ϕ_{LM}, every other feature ϕ_i is defined as a product over a function applied at the individual rule level. The total derivation probability is then computed by multiplying the weighted language model probability $P_{LM}(e)^{\lambda_{LM}}$ with the product over the other features, weighted by their feature weight λ_i:

$$\begin{aligned} P(\mathbf{d}) \;&\propto\; P_{LM}(\mathbf{e})^{\lambda_{LM}} \cdot \prod_{i \neq LM} \prod_{(X \to \langle \alpha, \delta \rangle) \in \mathbf{d}} \phi_i(X \to \langle \alpha, \delta \rangle)^{\lambda_i} \\ &= P_{LM}(\mathbf{e})^{\lambda_{LM}} \cdot \prod_{(X \to \langle \alpha, \delta \rangle) \in \mathbf{d}} \prod_{i \neq LM} \phi_i(X \to \langle \alpha, \delta \rangle)^{\lambda_i} \end{aligned} \tag{3.8}$$

By rearranging the two products, we obtain a product ranging over individual rule features. Apart from the language model feature, all other weighted features can be multiplied together for every rule separately, giving individual rule weights which are computed efficiently. Unfortunately, the computation of $P(\mathbf{d})$ demands multiplication with the language model probability $P_{LM}(e)$, which is not defined in terms of individual rules. This adds considerable complexity to the decoding process, and for this reason approximation is necessary in the form of beam-search with pruning, e.g., *cube-pruning* (Huang and Chiang, 2007; Chiang, 2007).

3.3 Extensions to Hiero

In this section we will look at relevant work that improves the word order produced by hierarchical statistical machine translation. We will focus on a selection of work that is either in some way close in terms of techniques or ideas to the work in this thesis, such as (Chiang, 2010) and (Huck et al., 2013) or has in one or more ways been influential

to our work, such as (Zollmann and Venugopal, 2006) and (Mylonakis and Sima'an, 2011). As one guideline, we mainly focused on relevant extensions to HIERO, yet not so much on *strict syntactic models*, but rather on approaches that label HIERO rules. We will now describe our criterion to distinguish strict syntactic models from models that refine HIERO rule labels with or without the use of syntax. A strict syntactic model is a model that makes syntax so central to its approach as to reject rules from the grammar that are accepted by HIERO, because of syntactic constraints. Some models such as SAMT and HIERO extended with soft syntactic constraints (Chiang, 2010) have a clear syntactic flavor. Yet they are still very close to HIERO because they use the same set of translation rules. The only difference between these models and HIERO is that the rules are enriched with syntactic labels that are used as hard or soft constraints during decoding.

Because of constraints of space we are unable to do justice to all work that may be relevant or related to our work in this thesis. In particular, we realize that there is a lot of work on:

- Inversion transduction grammars.
- Tree-to-string models.
- String-to-tree models.
- Tree-to-tree models.

which we want to mention here, without having space to go into detail for separate publications.

Inversion transduction grammars (ITGs) (Wu, 1997, 1995) were applied to translation in (Wu and Wong, 1998). Zhang and Gildea (2005) proposed a lexicalized variant of ITGs, used for word alignment. Tree-to-string models (Huang et al., 2006; Liu and Gildea, 2008; Langlais and Gotti, 2006; Nguyen et al., 2008; Zhang et al., 2007a; Hopkins and Kuhn, 2007), use a rich representation of the source side to produce unannotated target translations. String-to-tree models (Galley et al., 2004; M.Galley et al., 2006; Marcu et al., 2006; Huang and Knight, 2006; DeNeefe et al., 2007; Williams and Koehn, 2011), use syntactic trees on the target side to facilitate grammatically coherent output and use syntactic properties to ground reordering. Tree-to-tree models (Imamura et al., 2005; Nesson et al., 2006; Zhang et al., 2007b; Shieber, 2007; Mi et al., 2008b; Liu et al., 2009; Ambati et al., 2009), use syntax on both sides, trying to make the most of available syntactic information. This is just a limited selection of the work not further discussed in this thesis.

3.3.1 Syntax-augmented Machine Translation

Syntax-augmented machine translation (SAMT) (Zollmann and Venugopal, 2006) was the first approach to syntactically enriched translation that managed to outperform the

$$
\begin{array}{rcl}
VP & \rightarrow & NP^{\boxed{1}} \text{ zur kenntnis} \quad \| \quad \text{taken note of } NP^{\boxed{1}} \\
NP & \rightarrow & NP^{\boxed{1}} \text{ man } VB^{\boxed{2}} \text{ kann} \quad \| \quad NP^{\boxed{1}} \text{ we can } VB^{\boxed{2}} \\
X & \rightarrow & AUX^{\boxed{1}} \text{ nicht einzusehen , } X^{\boxed{2}} \quad \| \quad AUX^{\boxed{1}} \text{ hard to see } X^{\boxed{2}} \\
VP & \rightarrow & \text{zur erreichung } NP^{\boxed{1}} \quad \| \quad \text{to work towards } NP^{\boxed{1}} \\
X & \rightarrow & JJ^{\boxed{1}} \text{ feststellung } MD^{\boxed{2}} \text{ wir} \quad \| \quad JJ^{\boxed{1}} \text{ thesis that we } MD^{\boxed{2}} \\
NP & \rightarrow & NP^{\boxed{1}} \text{ einer frau } VBG^{\boxed{2}} \quad \| \quad \text{a woman } VBG^{\boxed{2}}\ NP^{\boxed{1}} \\
X & \rightarrow & \text{in } NP^{\boxed{1}} \text{ behandelt } AUX^{\boxed{2}} \quad \| \quad AUX^{\boxed{2}} \text{ given consideration in } NP^{\boxed{1}} \\
NP & \rightarrow & \text{ein mensch} \quad \| \quad \text{a man}
\end{array}
$$

Figure 3.3: SAMT (SCFG) rules for German–English, with single constituent labels and the default X label.

$$
\begin{array}{rcl}
NP & \rightarrow & NP/NN^{\boxed{1}} \text{ geschichte} \quad \| \quad NP/NN^{\boxed{1}} \text{ story} \\
X & \rightarrow & X^{\boxed{1}} \text{ einen } JJ + NN^{\boxed{2}} \text{ gemacht haben} \quad \| \\
 & & \text{have made a } JJ + NN^{\boxed{2}}\ X^{\boxed{1}} \\
NN + PP & \rightarrow & \text{entwicklung } JJ + NNS^{\boxed{1}} \text{ der } NNP^{\boxed{2}} \text{ union} \quad \| \\
 & & \text{evolution of } NNP^{\boxed{2}} \text{ union } JJ + NNS^{\boxed{1}} \\
X & \rightarrow & DT + NNS^{\boxed{1}} \text{ nicht } NP/PP^{\boxed{2}} \text{ auf} \quad \| \\
 & & \text{not } NP/PP^{\boxed{2}} \text{ about } DT + NNS^{\boxed{1}} \\
NP + NN & \rightarrow & \text{haushalt der europäischen union} \quad \| \quad \text{the european union budget}
\end{array}
$$

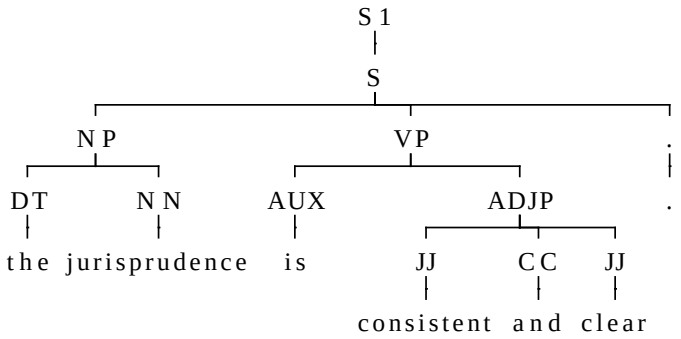
Figure 3.4: SAMT rules for German–English, with single constituent labels, compound nonterminal labels and the default X label.

strong HIERO pipeline. While initially the improvements were not large, the main point is that there were improvements at all. This is in stark contrast with other syntactic approaches that often do not even report comparisons against HIERO (Galley et al., 2004; Wang et al., 2010; Williams et al., 2014) including recent work. One important reason why syntax can harm performance is suggested by (Koehn et al., 2003). Their work shows that enforcing syntax as a constraint when selecting phrases decreases performance of phrase-based systems. Syntactic systems typically only allow rules that are consistent with the syntax, which means many rules available in HIERO are not allowed, and hence coverage is lost and with it also performance. The fact that re-structuring, re-labeling and binarization increase performance (Wang et al., 2010) seems to support the hypothesis that loss of coverage is the main reason that syntactic SMT often does not manage to catch up with HIERO. SAMT diverges from mainstream syntactic SMT by starting from HIERO as the basis, but replacing the HIERO *X* labels with syntactic labels. SAMT uses a heuristic labeling scheme reminiscent of *Categorial Grammar* (Hillel, 1953; Steedman, 1987, 2000). This scheme allows it to assign a, possibly compound, syntactic label to most phrases. When even no compound label can be formed using the heuristic scheme, the HIERO *X* default label is simply used. This assures that all rules in a HIERO grammar are also present in the corresponding SAMT grammar, only typically with different labels. Figure 3.3 shows examples of SAMT rules with single constituents as labels. Examples of rules with compound labels are shown in Figure 3.4.

SAMT Labeling Algorithm

We will now look at the algorithm that SAMT uses to form labels for target (or source) spans given a target (or source) constituency parse. As an example, consider the source-target sentence pair ⟨ *"die Rechtsprechung ist konsistent und klar."*,*"the jurisprudence is consistent and clear."*⟩ which has a completely monotonic word alignment. The parse for the target side is shown in Figure 3.5a, with Figure 3.5b showing the SAMT labels produced for different spans using this parse. The algorithm that finds a SAMT label given a parse tree and a span is a heuristic procedure. It first extracts a table of constituents from the parse tree, and then uses this table to form heuristic labels. To find a label, the algorithm uses the first applicable strategy amongst a list of alternatives ordered by decreasing preference:

1. $NT1$: Get a proper constituent for the span.
2. $NT1$ + $NT2$: Find a partition into two subspans that both correspond to constituents.
3. $NT1/NT2$ or $NT2\backslash NT1$: Find the smallest extended span (to the left or right) that allows a Categorial Grammar style label
   - $NT1/NT2$: $NT1$ missing a $NT2$ on the right



(a) Target parse.

Begin\End	0	1	2	3	4	5	6
0	DT	NP	NP+AUX	NP+AUX+JJ	FAIL	NP+VP	S
1	—	NN	NN+AUX	NN+AUX+JJ	FAIL	NN+VP	DT\S
2	—	—	AUX	AUX+JJ	VP/JJ	VP	VP+.
3	—	—	—	JJ	JJ+CC	ADJP	ADJP+.
4	—	—	—	—	CC	CC+JJ	CC+JJ+.
5	—	—	—	—	—	JJ	JJ+.
6	—	—	—	—	—	—	.

(b) SAMT labels.

Figure 3.5: Constituency parse for the target side *"the jurisprudence is consistent and clear .*", and SAMT label chart produced from this target parse for the different target spans. The start of a span [*begin*,*end*] is indicated in the row headers, and the end in the column headers. Spans are only valid if *begin* ≤ *end*.

- $NT2\backslash NT1$: $NT1$ missing a $NT2$ on the left.[9]

4. $NT1 + NT2 + NT3$: find a partition into three subspans that all correspond to constituents. (If "double-plus style" labels are allowed.)

5. X: No reasonably simple syntactic label could be found, return the default label.

The pseudocode for the SAMT labeling procedure is shown in Algorithm 11, in Appendix A.3. As an example, consider the span $[2, 5]$ which is a VP as can be verified from the parse tree, and read from the table. Similarly for span $[5, 5]$ there is a JJ. As expected the table shows that we get a label VP/JJ for span $[2, 4]$ formed by subtracting the missing JJ on the right from VP. Analogously, the parse shows that the span $[1, 6]$ is an S missing a DT on the left, and indeed the table shows that correspondingly the label $DT\backslash S$ is assigned to it.

[9]CCG (Steedman, 2000) uses $NT1\backslash NT2$ in place of $NT2\backslash NT1$, to indicate that $NT1$ misses $NT2$ on the left. The different notation used by SAMT, which places the argument itself to the left in this case can be confusing to people that are used to CCG notation.

SAMT Features

In this section we describe the features we use in our experiments. To be unambiguous we first need to introduce some terminology. Let r be a translation rule. We use $\hat{p}$ to denote probabilities estimated using simple relative frequency estimation from the word-aligned sentence pairs of the training corpus. The function *un* removes labels from nonterminal symbols. Then $src_{exc}(r)$ is the source side of the rule (excluding the source side of the left-hand-side label). Similarly $tgt_{exc}(r)$) is the target side of the rule (excluding the target side of the left-hand-side label).[10] SAMT uses the following basic rule probability features:

- $\hat{p}(r|lhs(r))$: Probability entire rule given left-hand side label[11]
- $\hat{p}(r|src_{exc}(r))$ Probability entire rule given only source side
- $\hat{p}(r|tgt_{exc}(r))$ Probability entire rule given only target side

As well as the following smoothing features:

- $\hat{p}(r|un(src_{exc}(r)))$, $\hat{p}(r|un(tgt_{exc}(r)))$: Probability entire rule given unlabeled source/target side
- $\hat{p}(un(tgt_{exc}(r))|un(src_{exc}(r)))$, $\hat{p}(un(src_{exc}(r))|un(tgt_{exc}(r)))$: Phrase probabilities using unlabeled phrases, as used by Hiero
- $\hat{p}_w(tgt(r)|src(r))$, $\hat{p}_w(src(r)|tgt(r))$: Lexical weights based on terminal symbols as for phrase-based and hierarchical phrase-based MT

In addition a set of standard features such as word-, rule- and rarity-penalties are added as well as some additional binary features. These additional features are also used by our own system and are described in more detail in section 5.4.1. The labeled rules used by SAMT can become very sparse, especially for rarer rules, causing the unsmoothed features to be unreliable. The smoothing probabilities play an important role in compensating for this. The effective usage of these smoothing features, tuned with MERT, is the second way in which SAMT succeeds to add syntax to Hiero without risking to lose much performance because of sparsity. Chapter 4 of (Zollmann, 2011) shows that these smoothing features result on average in 0.5 BLEU points improvement on Chinese–English translation.

SAMT: Extensions and Limitations

SAMT works with strict matching of labels during decoding, so that in principle the failure to match labels can still lead to a decrease of coverage. But by using many

[10] Later in chapter 5 we will use $src(r)$ and $tgt(r)$ similarly to denote the source and target side, *including* the source/target side of the left-hand-side of the rule.

[11] This feature is particularly important since some labels are inherently much more rare then others, so that rules with this label will on average also have more peaked conditional probabilities as a result of sparsity. Without compensating for this by means of these generative probabilities, our already not properly probabilistic model becomes even more skewed, as we are then effectively strongly rewarding rare rules, without giving any appropriate penalty for the rareness their left hand side.

alternatively labeled variants of rules in parallel, as observed during training, the risk of coverage loss is significantly reduced. There is a price to pay for this however. Using many different label variants for every HIERO rule type leads to huge grammars, increases spurious ambiguity and slows down decoding. Grammars may also become so big as to not even fit into memory. As a technical solution to this last problem (Zollmann, 2011) propose a method in which they use Hadoop to efficiently extract a separate grammar for each sentence that is translated.[12] A more effective and definite solution to the problems of lost coverage when adding syntax is to switch to using the syntax in the form of soft-constraints (Venugopal et al., 2009; Chiang, 2010) as discussed later in this chapter.

Various extensions to the original SAMT approach are proposed in Zollman's PhD thesis (Zollmann, 2011). One small extension, already mentioned before, is the usage of labels that consist of three combined constituents ("double-plus" labels). This extension may improve performance, but can also dramatically increase the number of rules. A bigger extension is the usage of source syntax, instead of or in combination with target syntax. Using only source syntax works, though not as good as only target syntax. Combining both source and target syntax in labels is detrimental to performance. Zollmann (2011) suggests that this is caused by a combination of badly estimated rare rules in combination with an increase in the number of blocked syntactic derivations due to mismatching labels. One other successful method to improve SAMT (as well as HIERO) also proposed in the same work is the usage of N-best alignments, instead of single alignments, when extracting grammars and computing rule counts. N needs to be kept small in order not to obtain a too large increase in decoding times. Usage of N-best parses during rule extraction was also attempted but did not help.

In the last chapter of his thesis, Zollmann (2011) furthermore explores alternative labeling schemes. He labels the words inside phrases with various methods and then combines the two labels of the phrase boundary words to form the final phrase labels. He proposes various methods to label the words, including simple POS-tagging, usage of bilingual word classes (Och, 1999) and K-means clustering. The former two methods give comparable performance to original SAMT, the last method performs worse. These approaches are interesting because they require less linguistic resources (no parses, only POS-tags) or even no linguistic resources at all, in the case of bilingual word classes. Still they are reported to retain the improvements of original SAMT.

3.3.2 Problems with (strict) syntactic models

Syntactic methods are elegant and attractive from a linguistic point of view. But in practice they often suffer a loss of performance in comparison to hierarchical phrase-based translation because of the strict syntactic constraints these models enforce. Coverage of these models can be increased by different methods. Binarization of

[12] To make this approach work well in practice, the decoder needs to be able to load grammars (very) fast. This is not possible with all open-source hierarchical SMT decoders.

translation rules (Wang et al., 2007) can directly increase their coverage, while re-labeling the trees can increase the chance that rules can be combined and re-aligning trees to the source/target strings may reduce the conflicts between syntactic constraints and alignment structure (Wang et al., 2010). Similarly in case of tree-to-string translation, the generalization to forest-to-string translation increases coverage and improves performance (Mi et al., 2008a).

All these methods have in common that they try to relax syntactic translation models. And they all try to solve the same fundamental problem: monolingual syntactic structure is not necessarily compatible with the hierarchical translation equivalence structure induced from word alignments. In contrast to syntactic models, Hiero is strictly an extension to phrase-based models. Because it uses all normal phrases as well a hierarchical generalizations of these phrases formed by introducing variables, it can only increase coverage over phrase-based methods, not reduce it. This is in strong contrast with strict syntactic models that use the syntactic structure as a hard constraint, which prohibits both many rules that are usable by Hiero and also limits the permissible combination of rules. When insisting on the usage of syntax it may therefore be wise to stretch syntax in a way to be more compatible with phrase pairs. SAMT (Zollmann and Venugopal, 2006) was discussed in the previous subsection as one successful approach to do this. Even better may be to use syntax as a soft rather than hard constraint (Chiang, 2010). But syntax is not the only way to add structure to the translation process, nor necessarily the best. In fact, word alignments themselves induce a rich hierarchical translation equivalence structure. Chapter 4 explains how this structure can be represented explicitly and chapter 5 shows how it can be used to improve translation of language pairs with considerable word order differences.

3.3.3 Other Labeling Approaches

We will now review various other successful labeling schemes. Most of these approaches primarily use syntactic information to form labels. Just as SAMT, they take Hiero as a basis and then refine the X label in its rules. This contrasts with strict syntactic models for hierarchical SMT, which eliminate many of the rules Hiero can extract by enforcing syntax as a hard constraint, even in the rule extraction process.

Head-Driven Hierarchical Phrase-based Translation

Head-driven hierarchical phrase-based translation (HD-HPB) (Li et al., 2012a,b) proposes an approach to label refinement that is similar to the boundary POS-tag labeling approach introduced in chapter 6 of (Zollmann, 2011) and discussed in our overview of SAMT extensions. But rather than taking the POS-tags of phrase boundary words, the authors propose to use information from the source dependency parse to select the POS-tags used to form labels. For a phrase f_i^j spanning source words i to j, any word $f_k (i \leq k \leq j)$ is regarded a **head** if a word outside the phrase dominates it. There can be multiple heads per phrase, these are combined by left-to-

right concatenation based on positions in the phrase to form the label. Rules containing nonterminals with more than four heads are discarded to limit sparsity.

Apart from the POS-tag selection method, another important difference with the POS-tag SAMT variant is the usage of source rather than target syntax.[13] Furthermore grammars are filtered at test time to comply with the source syntax of the test sentences. Filtering is done by checking first that the left-hand-side (LHS) nonterminal matches to the label of some span for some source sentence in the test set. If present, right-hand-side nonterminals are required to match the labels for subspans of the LHS matching span (in the right order). Filtering is only applied to fully lexicalized rules and Hiero rules with gaps, abstract rules are all included without filtering.[14] Furthermore, the unfiltered abstract rules can be applied to produce nonterminals on the left-hand-side that were initially not permitted based on the label information extracted from the syntax of the test sentences. These dynamically added nonterminals next open the possibility for both abstract rules and Hiero rules to substitute to them, adding possibly even more nonterminals that were initially not allowed.[15]

The filtering based on the source syntax of the test sentences reduces the amount of rules significantly, by a factor 30 approximately. And this leads to a considerable speedup as well, where a variant of the system that does not use abstract rules runs almost 2 times faster than Hiero. More importantly, with the help of the abstract rules the HD-HPB approach achieves on average a significant improvement of 1.91 BLEU over Hiero for a larger training set (1.5 M sentence pairs). For a smaller training set (240 K sentence pairs), without the help of abstract rules, the performance of the HD-HPB is below that of Hiero. Thanks to abstract rules, the system still beats Hiero with on average 1.32 BLEU improvement. On the bigger dataset the relative contribution of the abstract rules decreases, but still accounts for nearly half of the total improvement on average. This suggests that for Chinese–English, long distance reordering may be an important component of the total reordering problem. Completely removing Hiero's reordering limit for abstract rules is then apparently an efficient way to achieve big improvements in translation quality. And computational cost can be kept low provided that the used labels are derived from the source input. The latter assures that rules can

[13] Some SAMT variants also explore source syntax, but the most effective versions of SAMT use target syntax.

[14] Filtering is done for the entire combined test set, which is less precise, but easier to implement than filtering for every test sentence separate. The latter involves making private grammar for all test sentences and loading them separately for every sentence that needs to be decoded. In fact, even filtering per sentence is not exact since accepted rules can still be applied to places where they do not match, for example when source phrases or words reoccur in the same sentence but with different labels. It is thus nearly impossible to make source label matching by rule filtering exact, and even making it just nearly exact with per-sentence filtering is quite costly. These are strong arguments to prefer an alternative, exact solution. This solution is to adapt the decoder to match rules against a source label chart, it was implemented for example by (Mylonakis and Sima'an, 2011).

[15] It would be interesting to test whether the matching relaxation added by this approach is desirable or not. This might be done by comparing with a stricter matching policy, filtering also abstract rules or implementing the same labeling method but with matching inside the decoder as in (Mylonakis and Sima'an, 2011).

be filtered out of the used grammar if they do not match the source labels that occur in the test set.[14]

CCG Augmented Hierarchical Phase-Based Machine Translation

Yet another approach to label HIERO rules with much similarity to SAMT (Zollmann and Venugopal, 2006) is proposed by Almaghout et al. (2010). In their work, instead of using the SAMT labels, *combinatory categorial grammar* (CCG) (Steedman, 1987, 2000) labels are used to label the target side of nonterminals in HIERO rules. Their work is also similar in spirit to (Hassan et al., 2007) which applied *supertags* (Bangalore and Joshi, 1999; Clark and Curran, 2004) to phrase-based translations, using them in the target side of the language model and the target side of the translation model.

CCG labels use a very restricted set of atomic categories in combination with the two binary CCG operators to form complex categories. The atomic categories are typically restricted to a very limited set. Typically: S (sentence), N (noun), NP (noun phrase). In addition sometimes: PP (prepositional phrase) and VP (verb phrase). These categories are then combined by the functors:

- $X\backslash Y$: a functor which takes as argument the category Y to the left and produces the category X. (Intuitively meaning "X missing Y on the left.)
- X/Y: a functor which takes as argument the category Y to the right and produces the category X. (Intuitively meaning "X missing Y on the right.)

Both X and Y in these definitions can be complex or primitive categories. CCG supertags can give a more precise description of constituents by forming more complex labels. CCG labels also allow to label a larger fraction of the available phrases: (Almaghout et al., 2010) report that with CCG labels only 30% of the phrases remain unlabeled as opposed to 50% for SAMT. While this is clearly a large gain in coverage, a large fraction of phrases still remains unlabeled.

In the experiments (Almaghout et al., 2010, 2011) work with strict matching of the labels during decoding, and consequently like SAMT suffer from issues related to loss of coverage by blocked substitutions.

Coarsening the labels The problems with sparsity of labels is a motivation to coarsen the labels, which is done in (Almaghout et al., 2011). First, full CCG labels are replaced by simpler labels that combine just the left and right contextual CCG categories and leave out the resulting categories. Let $C = (R\backslash L1)/L2$ be a CCG category (label), with R being the resulting category, $L1$ the left argument category and $L2$ the right argument category. The CCG *contextual label* becomes $L1\_\ L2$, indicating $L1$ is required on the left and $L2$ on the right. Second, features carried by some atomic labels such as declarative $S[dcl]$ or wh-questions $S[wq]$ are removed, coarsening both refined S categories to the featureless S category. Removing such features throughout in atomic and complex categories further simplifies the labels. This

is tried out both for the CCG contextual labels and the original (full) CCG labels. The result is four different variants of the labeling scheme, including the original CCG labels of (Almaghout et al., 2010).

Results Some of the labeled systems show improvements over Hiero, but none of these improvements are statistically significant. The proposed ways to coarsen the labels help, but not in a stable way across language pairs and domains. The CCG contextual labels perform the best in two out of four experiments. Yet whether or not the second simplification of removing the CCG category features is beneficial differs even over these two experiments. This shows once again how hard it can be to improve over Hiero with a strict labeling approach.[16]

In a later work (Almaghout et al., 2012) apply *soft constraints* as proposed by (Venugopal et al., 2009) instead of performing strict matching. This finally leads to substantial and significant improvements over a Hiero baseline for Arabic–English and Chinese–English translation tasks. This shows the importance of relaxing strict matching and using instead a form of soft constraints, an approach that is discussed in more detail later in this chapter.

SATM Label Coarsening

A method to coarsen the labels of SAMT, by performing clustering is proposed in (Hanneman and Lavie, 2013). We will refer to this method as *SAMT-coarsening*. Source and target side are parsed, and from these parses bilingual SAMT labels are produced. The similarity between conditional distributions of label distributions in two directions is used to merge labels. Labels are merged greedily and incrementally, every time merging the two labels that have the most similar conditional distributions over labels on the other side. This is essentially a form of hierarchical/agglomerative clustering (Press et al., 2007) whereby instead of elements of incrementally merged sets, the probability distributions of incrementally merged clusters are compared. Given a source label s and a target label t, label distributions $P(s|t)$ and $P(t|s)$ are simply computed by relative frequency estimation:

$$P(s_i|t_j) = \frac{Count(s_i :: t_j)}{\sum_{s \in S} Count(s :: t_j)} \qquad P(t_j|s_i) = \frac{Count(s_i :: t_j)}{\sum_{t \in T} Count(s_i :: t)} \tag{3.9}$$

And based on these distributions L1 distances are defined for all pairs of monolingual labels:

$$d(s_1, s_2) = \sum_{t \in T} |P(t|s_1) - P(t|s_2)| \qquad d(t_1, t_2) = \sum_{s \in S} |P(s|t_1) - P(s|t_2)| \tag{3.10}$$

[16] There is also some reason to believe that the relatively small sizes of the training corpora used (at most 546K sentence pairs for Chinese English) in the experiments could also be a reason why the labeled systems suffered in the experiments. It is shown in (Zollmann, 2011), chapter 3 and also again shown in (Li et al., 2012b) that a large enough training set is important to reduce sparsity enough when working with (finely) labeled hierarchical grammars.

At each iteration of the clustering algorithms the two source or target labels with the most similar distributions are merged, formally defined as:

$$\arg\min_{\langle s_i,s_j\rangle\in S^2,\langle t_k,t_l\rangle\in T^2} \{d(s_i, s_j), d(t_k, t_l)\} \tag{3.11}$$

After clustering stops, the coarsened SAMT grammar is formed. For every SAMT rule, each of its labels is mapped to its associated cluster, and replaced with the target side of the label for that cluster. The resulting coarsened grammars have much less rules than original SAMT, and give significant improvements over SAMT and also over Hiero for Chinese–English translation on some test sets. A problem is how to determine when to stop label collapsing. Hanneman and Lavie (2013) just explored the results for several manually selected stopping points.

Interestingly the authors first applied the method for coarsening of bilingual labels, in a bilingually labeled decoding setting (Hanneman and Lavie, 2011) (i.e. not discarding the source side as with SAMT). While no direct comparisons were given with Hiero, combining the two papers a comparison appears to be possible, assuming the same test settings were used across experiments in both papers. Under these assumption the comparison reveals a substantial loss of the bilingually labeled system on BLEU in comparison to the Hiero baseline. This shows the difficulty of making bilingual labels work well in a strict matching setting, even when label coarsening is applied.

Improved, fast SAMT Label Clustering Mino et al. (2014) propose an approach to SAMT label Coarsening/Clustering similar to (Hanneman and Lavie, 2013), that yields comparable improvements over SAMT but uses a much faster clustering algorithm. Their approach uses an exchange algorithm (Uszkoreit and Brants, 2008) that is an order of magnitude faster than the agglomerative clustering used by (Hanneman and Lavie, 2013).

Summary of strategies to reduce the negative effects of strict label matching

In this subsection we have looked at various other labeling approaches beyond SAMT. Like SAMT and in contrast to strict syntactic methods, these approaches do not discard phrases incompatible with syntax but rather keep them and leave them unlabeled. Nevertheless, these methods still face problems due to the strict enforcement of label matching during decoding. In particular, they suffer from problems that rise from blocking valid translations because labels do not match. Partly to reduce these problems, the discussed approaches use several strategies, which we summarize here:

- Heuristically coarsening labels so that more phrases can be labeled: (Almaghout et al., 2011).
- Automatically coarsening labels using clustering methods: (Hanneman and Lavie, 2013; Mino et al., 2014).

- Using alternative information (e.g. POS-tags) in a way that assures all phrases can be labeled: (Li et al., 2012a,b) and (Zollmann, 2011), chapter 6.

While these strategies all help to reduce the severity of the problems caused by strict matching, essentially none of them succeeds in solving them completely. The next section discusses approaches that attempt to solve the original problem at the root, by using labels as soft constraints as opposed to strictly enforcing them to match.

3.3.4 Soft Constraints

Making contextual refinements to HIERO rules by labeling nonterminals is an important technique for improving target word order, and generally producing better, more coherent translations. But labeling also carries big risks, as it can easily lead to sparsity an blockage of valid rules, which reduces coverage and decreases translation accuracy. While these problems become critical when syntax is applied to both sides (Zollmann, 2011), they are also the main reason why strict labeling approaches frequently fail to consistently improve over a HIERO baseline (Almaghout et al., 2011; Hanneman and Lavie, 2011). Automatic coarsening of labels, using clustering can help a lot (Hanneman and Lavie, 2011, 2013; Mino et al., 2014) while automated learning of label distributions (Mylonakis and Sima'an, 2010, 2011) or latent variables that play a role similar to labels (Saluja et al., 2014) may be even better. But these clustering and learning methods are only half of the solution. Because even when labels are learned, typically many derivations of the same translations compete with each other. The exception to this are systems that allow only minimal rules such as (Saluja et al., 2014; Vaswani et al., 2011), so that only one derivation per translation exists. But that approach itself has the big disadvantage of eliminating many larger rules, which are essential for the performance of HIERO as they introduce a much stronger context to translation decisions. Whether or not labels are learned, it is much preferable to use these labels as soft rather than hard constraints. Here we briefly summarize the main properties of two of the most popular approaches towards soft constraints, referring the interested reader to a longer and more detailed discussion in Appendix A.4.

Preference Grammars It was shown that when labels are used as latent distributions, in a framework called *preference grammars* (Venugopal et al., 2009), they can yield significant improvements over HIERO. The latter approach effectively approximates the summation of probabilities over different labeled derivations for the same translation. By performing approximate summation over derivations that only differ in their labels, preference grammars find the derivation class with the highest probability.[17]

Soft Syntactic Constraints Chiang (2010) notes that the preference grammars approach still only includes derivations that satisfy the matching constraint, and

[17] Just as in HIERO, the same translations can still be formed by alternative derivations that segment the input in different ways. Therefore just as in HIERO the translation with highest probability is not found, only the most likely derivation class.

proposes instead to soften the matching constraint itself. This is called in the literature decoding with *soft matching constraints*, or often just *soft constraints* or *fuzzy matching*. It was shown by Chiang (2010) that when labels are used as soft constraints, source and target labels can be effectively combined in a system that gives significant improvements over a Hiero baseline.

Recently, the effectiveness of soft constraints was again established for English–German Translation by (Huck et al., 2014) using GHKM-style (Galley et al., 2004) string-to-tree translation, with soft syntactic constraints on the source side. In their work rules are extracted under the constraint that they are compatible with the target syntax, which means the total rule set is a small subset of what is available to Hiero. In that setting it turned out to be better to use the target labels as hard constraints than as preference grammars, at least on English–German translation. But this may well be just a consequence of using syntax as a hard constraint even for rule extraction.[18]

Whether preference grammars or soft-constraints as introduced by Chiang (2010) are preferable is mostly an empirical question. In a sense these methods are similar: while preference grammars approximate summation over different labeled derivations, learning soft constraints for the matching of labels achieves a similar goal by finding to what degree certain labels are interchangeable. But this also reveals a strength of the soft constraints approach of (Chiang, 2010) which is not shared by preference grammars. This strength is that by using soft constraints, the method can learn to give high importance to the correct matching of relevant labels, while simultaneously learning to ignore labels that are uninformative. In the limit this approach can even implicitly cluster interchangeable labels, by learning matching weights that achieve such a clustering. Preference grammars in contrast, have no mechanism to achieve such learning and therefore have no way to compensate for labels that are possibly not learned and therefore suboptimal when applied in their raw form. The dependence on scalable discriminative learning approaches of (Chiang, 2010) can also be a weakness, but with the general availability of popular and successful methods such as batch-mira (Cherry and Foster, 2012) this is no longer really a problem. What remains a challenge with both methods is that they are difficult to implement in a way that scales well to large label sets.

3.3.5 Learning labels

In the previous subsection we discussed soft constraints as a fundamental way to overcome the problems with strict label matching during decoding. Here we look at learning of labels as another way deal with these problems. While there is a considerable amount of work in this area, here we restrict ourselves to the discussion of two methods with the highest impact on the work in this thesis.

The first method uses a variant of the EM algorithm to learn the labels as well as

[18] As the latter implies a smaller choice of suitable lexically grounded translation rules, and hence a somewhat artificially higher demand on syntax to guide word order amongst other things.

$$\begin{array}{lcl} A & \rightarrow & [B\ C] \\ A^L & \rightarrow & [B\ C] \\ A^R & \rightarrow & [B\ C] \end{array} \qquad \begin{array}{lcl} A & \rightarrow & \langle B^L\ C^R \rangle \\ A^L & \rightarrow & \langle B^L\ C^R \rangle \\ A^R & \rightarrow & \langle B^L\ C^R \rangle \end{array}$$

(a) Reordering rules

$$\begin{array}{lcl} A & \rightarrow & A_P \\ A^L & \rightarrow & A_P^L \\ A^R & \rightarrow & A_P^R \end{array}$$

(b) Pre-terminal rules

$$\begin{array}{lcl} A_P & \rightarrow & \alpha\ /\ \beta \\ A_P^L & \rightarrow & \alpha\ /\ \beta \\ A_P^R & \rightarrow & \alpha\ /\ \beta \end{array}$$

(c) Terminal rules

Figure 3.6: Rule types of a hierarchical reordering SCFG (HR-SCFG) (Mylonakis and Sima'an, 2011).

the rule segmentations of the aligned sentence pairs (Mylonakis and Sima'an, 2011). Although this work uses only phrase pairs and abstract binary rules, it is highly relevant to our work. The reason is that it uses labels to keep track of the relative orientation (straight, inverted) of abstract rules with respect to their parents.

The second method (Saluja et al., 2014) first simplifies things by working with minimal rules so that there is just one segmentation. For this simplification it initially pays a large price in terms of performance. It then however manages to combine learning of the labels with using them as a form of soft constraints in a reranking step, with much similarity to preference grammars (Venugopal et al., 2009). Despite its modest improvement over Hiero, this approach is significant as a proof of concept that soft constraints and learning can in principle be combined.

Hierarchical Reordering SCFG

Mylonakis and Sima'an (2011) present an approach to hierarchical translation that focuses on learning a binary reordering SCFG (NF-ITG) with linguistically motivated source labels. The approach uses the same labels as SAMT, but rather than selecting for each span the simplest possible label, all applicable labels are kept and used to form alternative rules. A variant of the EM algorithm (Dempster et al., 1977) called Cross-Validated EM (CV-EM) (Mylonakis and Sima'an, 2008, 2010) is used to estimate the parameters of this grammar. Another innovation of the approach is the usage of labels that mark the reordering context of the ITG reordering rules. Left-swapping (L) and right-swapping (R) labels (see figure 3.6a) indicate that the left-hand-side of rules is embedded as the left or right part of an inverting rule. Such rules facilitate

a form of Markovization that propagates the information of swapping by a rule to its two reordered children. The used grammar formalism combines a labeled NF-ITG reordering grammar with labeled phrase pairs. The phrase pairs (see Figure 3.6c) are used as leaf nodes, while pre-terminal rules (see Figure 3.6b) substitute onto them and facilitate the transition to the separated reordering grammar.

Consistency with labels for the source input is enforced by requiring a match between rule labels and labels in a source label chart. This source label chart is provided to the decoder together with the source sentence. As an example, if the rule $A \rightarrow [B\ C]$ is applied to derive label A for span $[i, j]$ by combining label B for subspan $[i, k-1]$ and label C for subspan $[k, j]$, then label A must be present in the source label chart for span $[i, j]$. Furthermore, at the moment of adding labels B and C earlier during decoding, corresponding labels must have been present in the source label chart for their spans $[i, k-1]$ and $[k, j]$. But because multiple alternative labels are present for every source span, including the default label X, the grammar always has an ability to back-off to rules with less specific labels. Thus a grammar can be learned which gives preference to rules with more specific labels. Yet the risk of completely blocking certain translations during decoding is avoided.

Experiments show significant improvements over a strong Hiero baseline on four language pairs, with English as source. The highest improvement, +1.92 BLEU, is achieved for English-Chinese translation.

Latent-Variable SCFGs for Hierarchical Machine Translation

Saluja et al. (2014) propose a method to learn latent variables for synchronous CFGs (SCFGs). The latent variables are used to effectively reduce independence assumptions and improve coherence in the composition of rules into full translations. In order to avoid the complex problem of simultaneously learning the latent variables and the segmentations of word alignments, the authors work with minimal rules (M.Galley et al., 2006; Zhang et al., 2008a; Vaswani et al., 2011) instead of Hiero rules. This guarantees the existence of just one derivation per translation, greatly simplifying the learning problem. The latent states of rule nonterminals are represented as *tensors*.[19] These tensors are multiplied in a hypergraph (tensor) inside-outside algorithm, that re-estimates the marginal probabilities of translation rules, as well as the conditional probabilities $\hat{P}(e|f)$ and $\hat{P}(f|e)$ for these rules. In this computation, 3^{rd}-order tensors as opposed to matrices (2^{nd}-order tensors) are required in the case of binary rules, to capture the relation whereby the rule tensor takes the vectors of its two nonterminals as inputs to produce an output vector for the left-hand-side of the rule.

[19]Tensors are multidimensional arrays that generalize scalars and vectors. A 3^{rd}-order tensor T can be imagined as a stack of matrices. When T is combined in a tensor-vector product with two input vectors $\vec{v}_1$ and $\vec{v}_2$ to produce an output tensor, this corresponds to the following computation: First $\vec{v}_1$ is multiplied (on the right) with each of the stacked matrices, producing a single intermediate result matrix M_{int}: $T \cdot \vec{v}_1 = M_{int}$. Second, $\vec{v}_2$ is multiplied (on the right) with M_{int} to produce the final result vector $\vec{v}_{result}$: $M_{int} \cdot \vec{v}_2 = \vec{v}_{result}$.

These features are then used as extra features on top of a standard MT pipeline. Essentially a standard decoder with standard features first produces a hypergraph of translations. As a second step, the latent-variable features are computed from this hypergraph, by multiplying the contained rule tensors as part of the hypergraph tensors inside-outside algorithm. The result is essentially a reranking of the original translation scores, with similarity to other forest reranking methods (Huang, 2008) as well as variational methods (Li et al., 2009) and Minimum Bayes Risk decoding methods (Kumar and Byrne, 2004; Tromble et al., 2008). In contrast to the mentioned related methods, this approach works with one derivation per translation and focuses on forming better translations by loosening the independence assumptions in the selection of rules.

For the sake of brevity, here we omit the discussion of learning of latent variable tensors. The interested reader is referred to Appendix A.5 for a discussion of the learning method and other details about the method.

3.3.6 Lexicalized Orientation Models for Hierarchical SMT

We end this chapter by discussing two approaches that adapt the lexicalized hierarchical phrase reordering model proposed by (Galley and Manning, 2008) for usage with hierarchical phrase based translation (HIERO). This model has proven its effectiveness for phrase-based translation but was not used before in hierarchical translation. These approaches are of particular interest in the context of this thesis, as they partially share motivations and mechanisms with the reordering labels that are proposed in chapter 5.

A Phrase Orientation Model for Hierarchical Machine Translation

The first instantiation of the idea of adopting a lexicalized orientation model in HIERO was proposed by Huck et al. (2013). The approach has two main parts. The first part entails the computation of lexicalized orientation probabilities for phrase pairs and HIERO rules with nonterminals. The second part consists of the usage of these probabilities to add reordering features to rule applications in HIERO derivations, based on the dynamically determined hierarchical phrase orientation of the phrase that is formed by applying a rule. The first part is computed statically before decoding. The second part involves adapting the decoder, to dynamically add or update features once the orientation of a phrase pair can be determined. The orientation can often but not always be determined once another HIERO rule substitutes to the phrase pair on the right-hand-side in bottom up CYK+ decoding. We will now discuss these two parts in some more detail.

Computing Lexicalized Orientation Probabilities The computation of lexicalized orientation probabilities is done based on the aligned parallel corpus used for training HIERO. From this corpus phrases are extracted without length constraints, which are then used to determine the (hierarchical) phrase orientation of phrase pairs as described

in (Galley and Manning, 2008). Orientations can be combined in two directions: left-to-right or right-to-left. Both directions may provide different information that could be complementary for the determination of the best word order during translation.

The prior probability of orientations $p(O)$ independent of phrase pairs, is computed by relative frequency estimation using the training set:

$$p(O) = \frac{N(O)}{\sum_{O' \in M,S,D} N(O')} \tag{3.12}$$

Here $N(O)$ is the total number of occurrences of orientation O over the entire training set, summed over all different phrase pairs. M, S and D stand for monotone, swap and discontinuous orientation respectively.

The lexicalized orientation probability given a phrase pair $\langle \tilde{f}, \tilde{e} \rangle$ is then computed as:

$$p(O|\langle \tilde{f}, \tilde{e} \rangle) = \frac{\sigma \cdot p(O) + N(O|\langle \tilde{f}, \tilde{e} \rangle)}{\sigma + \sum_{O \in M,S,D} N(O'|\langle \tilde{f}, \tilde{e} \rangle)} \tag{3.13}$$

Here $N(O|\langle \tilde{f}, \tilde{e} \rangle)$ is the number of times the phrase pair $\langle \tilde{f}, \tilde{e} \rangle$ occurs with orientation O in the training set. Furthermore σ is a constant that determines how much weight should be given to the prior orientation probability, which is used for smoothing. In the case of a hierarchical rule r_h with a nonterminal N_α, orientation counts are accumulated over all phrases from which r_h is formed by replacing a sub-phrase by N_α. These accumulated counts are then used to estimate the lexicalized orientation probability. The computation for Hiero rules with two nonterminals is done analogously.

Reordering feature computation

The computation of lexicalized reordering features during decoding requires determining the left-to-right and right-to-left orientation of phrases. To be able to do so, every Hiero rule is extended to keep the alignment matrix that has been most frequently observed for it during training. The orientation of the nonterminals can then be determined in most cases, by combining the word alignment information kept in the rules with the information about the relative source and target spans of the nonterminals. The exception to this is when a nonterminal is on a *boundary position* on the target side. This happens *iff* there are no nonterminals or words aligned between the phrase-internernal target index of the nonterminal and the (left or right) phrase boundary. In this case the scoring needs to be delayed until the orientation can be established in an upper hypernode later during decoding. When this delayed scoring is necessary, a temporary orientation cost is used, computed from fractional costs for the orientations that are still possible. This avoids giving unjustified advantage to other derivations. We refer to Huck et al. (2013) for a full description of the procedure for determining orientations, and computing and updating feature values, as well as examples of cases in which the orientation is directly known or can only be established

later.

Results

The lexicalized reordering features are tested for Chinese–English translation where they give an improvement of 1.2 BLEU over the Hiero baseline. Most of this improvement is achieved by left-to-right orientation features, the addition of right-to-left orientation features only increase the score by 0.1 BLEU. For French-German translation an improvement of 0.3 BLEU over Hiero is achieved. The authors also compare against their own earlier work on discriminative reordering extensions to Hiero (Huck et al., 2012), which we will abbreviate here as *DRE*. Furthermore they build a very strong system by adding discriminative word lexicon (*DWL*) models (Bangalore et al., 2007; Mauser et al., 2009) and *triplet* lexicon models (Hasan et al., 2008) in source-to-target and target-to-source direction. In the setting where all these additional components are used, which by itself gives a 2.5 BLEU improvement over Hiero, the lexicalized reordering features still add an additional 0.7 BLEU additional improvement. This in contrast to addition of DRE, which barely increases performance further in the setting of this very strong initial system.

Integrating Phrase-based Reordering Features into a Chart-based Decoder for Machine Translation

The work by Nguyen and Vogel (2013a) was published at the same time as (Huck et al., 2013) and proposes a similar extension to Hiero. The main difference is that the lexicalized reordering model used in this work is the *phrase-based* orientation model taken from (Galley and Manning, 2008), as opposed to the *hierarchical* orientation model adopted by (Huck et al., 2013), taken from the same paper. Another smaller difference is that this work also adds a basic distance-based reordering model, as commonly used in phrase-based SMT, in addition to the lexicalized reordering model. One restriction of the work is that rules with non-aligned lexical items inside it were not allowed. While mainly a technical issue that did not seem to compromise performance much, although it did yield a significant decrease in the amount of available rules, this restriction was not present in (Huck et al., 2013). Overall (Huck et al., 2013) and (Nguyen and Vogel, 2013a) mainly reinforce each other by presenting similar improvements for a similar approach. (Nguyen and Vogel, 2013a) presents experiments showing significant improvements for three language pairs, including Arabic–English, which was not covered by (Huck et al., 2013). Since the improvements are similar and the experimental setups not entirely comparable, it is hard to conclude definitely which approach is more successful. Nevertheless, it seems plausible to believe that the improvements made for Chinese–English by (Huck et al., 2013) are stronger. The reason is that it uses a much larger training set (3.0M) as compared to the relatively small (384 K sentences) training corpus used by (Nguyen and Vogel, 2013a). Consequently the baseline for (Huck et al., 2013) has a score that is 2.6 BLEU higher, and it is plausible to believe that having more training material makes

the basic Hiero reordering patterns more reliable and harder to improve upon, making the either way slightly higher improvement by (Huck et al., 2013) just relatively more significant. More importantly the original work by (Galley and Manning, 2008) shows that the *hierarchical* orientation model is far superior for phrase-based translation, suggesting that it should also be better for hierarchical translation.

Summary and outlook

In this chapter we have looked at hierarchical SMT, synchronous grammars and related work. We have looked at inversion transduction grammars and Hiero, and in the discussion of related work concentrated on labeling approaches that strive to improve composition and reordering in Hiero using labels.

But while these labels are typically based on syntax, in the next chapter we cover the representation of the hierarchical translation equivalence and reordering structure inherent in word alignments themselves. This implicit information induced by word alignments is then made explicit in structures called hierarchical alignment trees (HATs). Based on these structures, Hiero grammars can be labeled with specific reordering labels, thereby significantly improving the quality of translations in hierarchical SMT, which is discussed in chapter 5. And also, with their help the coverage of word alignments by grammars can be studied in a structured and exact way, which is discussed in chapter 6.

Lastly, HATs have also been successfully applied for automatically learning and performing adequate preordering of translation input before decoding, to improve target word order (Stanojević and Sima'an, 2015). This application, and the application of HATs for evaluation (Stanojević and Sima'an, 2014c,b) are discussed in some more detail in the discussion of related work, in chapter 5, subsection 5.8.6.

4

Representing Hierarchical Translation Equivalence: Hierarchical Alignment Trees

"Take some more tea," the March Hare said to Alice, very earnestly.
"I've had nothing yet," Alice replied in an offended tone, "so I can't take more."
"You mean, you can't take less," said the Hatter : "it's very easy to take more than no-thing."
"Nobody asked your opinion," said Alice.
"Who's making personal remarks now ?" the Hatter asked triumphantly.

– Lewis Carroll, *Alice in Wonderland*

In this chapter we will introduce a framework for the compact and exact representation of hierarchical translation equivalence called HATs. This chapter builds upon existing work on the decomposition (also called factorization) of translation equivalence, algorithms and datastructures for the maximal decomposition of permutations called *permutation trees* (PETs) (Gildea et al., 2006; Zhang and Gildea, 2007) and the maximal decomposition of general word alignments *normalized decomposition trees* (NDTs) (Zhang et al., 2008a) in particular. These works in turn build upon an even earlier more theoretically, purely mathematically oriented work concerning the decomposition (also called factorization) of permutations (Albert et al., 2005).

In the next two sections we will explain the motivation and novelty of our work, and give an intuitive presentation of HATs. In section 4.3 we describe the existing work on the decomposition of translation equivalence. After that, in section 4.4 we describe hierarchical translation equivalence, which builds further on general notions of translation equivalence described in section 2.3.2 of the background chapter. Section 4.5 then describes set permutations, which are two foundations for the formulation of HATs. HATs are described in detail in section 4.6 and finally the efficient computation of HATs is described in section 4.7.

Translation Model Type	adequate local disambiguation	adequate generalization and handling discontiguity	supporting global compositional coherence
word based	✗	✗	✗
phase based	✓	✗	✗
hierarchical phrase based	✓	✓	✗
hierarchical phrase based with adequate labels	✓	✓	✓

Table 4.1: Schematic view of translation model capabilities. Note that these capabilities are not meant as absolutes, but rather are intended to summarize the main relative conceptual advantages gained by using these incrementally more complex translation models.

4.1 The challenge of representing hierarchical translation equivalence

Phrase pairs are the basic units of most contemporary statistical translation systems. The simple intuition to use many non-minimal overlapping fragments goes back to earlier work on *data-oriented parsing* (Scha, 1990; Bod, 1992; Sima'an et al., 1994). The usefulness of composed fragments is not limited to parsing. Since those fragments provide strong disambiguation power, they appear to be even more important for translation, which is highly ambiguous at the word level. Such large, specific fragments implement the strategy of building reliable translations by exploiting as much as possible what has already been seen and not extrapolating unnecessarily. This conservative strategy works well, but has the drawback of offering poor generalization power.

The constraint that phrase pairs are by definition contiguous on both source and target side further restricts their usability for the purpose of translating language pairs with big word order differences, such as Chinese–English or German–English. Hierarchical phrase pairs used by Hiero introduced in (Chiang, 2005) overcome this limitation by introducing gaps to phrase pairs, lifting them to form (lexicalized) synchronous context free grammar rules. This both increases the generalization ability particularly for reordering phenomena and also the ability to effectively deal with discontiguous translation phenomena. Still, just like their non-hierarchical counterparts, hierarchical phrase pairs use no labels and thus have no context available to encourage the formation of coherent translations. Lexicalization of the rules helps somewhat to reduce ambiguity and provides some lexical evidence for the individual rules, however it can do nothing to promote global coherence of the composed rule applications.

Our starting point is thus hierarchical statistical machine translation, and our goal is to make translations more coherent, particularly with respect to word order. In Table 4.1 we schematically summarize the situation. Phrase based models have managed to achieve adequate local disambiguation and hierarchical phrase based models have

brought in adequate generalization and handling of discontiguity. But in order to also achieve global compositional coherence more information is needed to sensibly combine rules in a compositionally informed way, and we hypothesize adequate labels can achieve that. Syntactic information based labeling approaches are a way to increase global coherence, and they have been applied with success, for example see (Zollmann and Venugopal, 2006; Hassan et al., 2009; Mylonakis and Sima'an, 2011; Chiang, 2010; Hanneman and Lavie, 2013; Li et al., 2012b). But we believe there is an inherent mismatch between monolingual syntactic information and, bilingual translation information, a problem which holds to a lesser extent also for dependency information. Furthermore, we do not want to rely on these additional resources which are not always available, but we rather want to use the rich compositional information already implicitly given by just the alignment structure itself.

To get the required compositional information from the alignments demands a clear and complete representation of hierarchical translation equivalence. For such a representation it is not enough to compactly represent all phrase pairs. Rather the representation must allow to reconstruct the original compositional translation equivalence relations from its components, which requires equipping the components with enough contextual information to remember how they were originally composed. This leads to our main hypothesis of this chapter, which motivates the creation of our Hierarchical Alignment Tree representation:

Hypothesis. *A representation that adequately captures the hierarchical bilingual structure of word-aligned bilingual training data provides the necessary and sufficient means to create fragments that can reconstruct bilingual structure and word order for seen data. Further, through generalization such a representation can predict likely structure and word order for unseen data. In the form of translation rules, fragments created from such a representation are capable of producing translations that coherently reproduce the type of reordering observed in the training data.*

Next we will specify exactly what we believe constitutes an *adequate* representation of hierarchical bilingual structure, as a set of criteria that such a representation must meet. Naively we could equip every TEU with a unique label, and if we then use these labels to label nonterminals when extracting synchronous rules, we are sure to reconstruct exactly and only our original translation equivalence patterns. This is not very useful however, as it offers no generalization at all. Another requirement is thus that extractable labels can support different levels of generalization as required by the specific application.

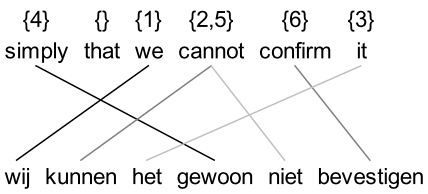
We therefore look for a representation that satisfies the following **criteria for adequate hierarchical bilingual structure representations of word alignments**:

1. Represents not only the TEUs but also their bilingual mapping in a compositional, hierarchical way.

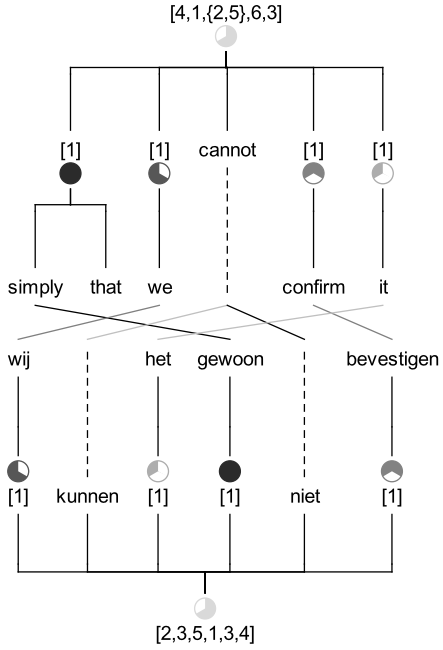
2. Retains exactly all information present in word alignments.

3. Enables the construction of labels that give an adequate description of the context, by representing explicitly all required information for this.

This chapter first gives an intuitive, example based presentation of HATs. As part of this presentation, it explains how HATs satisfy the above criteria and how HATs relate to other decompositions of word alignments. It then reviews the existing work on the decomposition of translation equivalence, giving a summary of the existing work on permutation trees and normalized decomposition trees described in (Gildea et al., 2006; Zhang and Gildea, 2007) and (Zhang et al., 2008a). After these foundational works have been reviewed, we continue with our contribution of *hierarchical alignment trees* (HATs) which is based on the work described in (Sima'an and Maillette de Buy Wenniger, 2013; Maillette de Buy Wenniger and Sima'an, 2014b). This presentation of the existing work and our extension together in one chapter was chosen because it facilitates a clearer and more natural exposition. We will end the chapter with a short description of applications, in particular improving reordering for hierarchical statistical machine translation and facilitating tools for effective analysis and visualization of hierarchical translation equivalence as induced by word alignments.



(a) Example of a word alignment, with discontiguous word mappings



(b) Hierarchical Alignment Tree for (a).

$$([1,6],[1,6])$$

$$([1,1],[4,4]) \quad ([3,3],[1,1]) \quad ([5,5],[6,6]) \quad ([6,6],[3,3])$$

(c) Normalized decomposition tree (Zhang et al., 2008a) for (a).

Figure 4.1: A word alignment (a), with discontiguous word mappings, and its corresponding Hierarchical Alignment Tree (b) and normalized decomposition tree (c). In (b) set-permutation labels, such as [4,1,{2,5},6,3] denote the local relative reordering mapping at every node. Circles with different fillings and shades are used to indicate matching TEUs on the source and target side of the HAT. In (c) note that the NDT representation does not explicitly represent the mapping relations, in contrast to the HAT representation. Also information about the mapping of the discontiguously aligned words, present in the HAT representation, is completely lost in the NDT.

4.2 An intuitive, example-based presentation of HATs

In Figure 4.1a we show a word-aligned sentence pair taken from European Parliament training data for English–Dutch, with in Figure 4.1b the corresponding hierarchical alignment tree (HAT). As discussed in chapter 2, in subsection 2.3.2, word alignments induce TEUs, in particular a set of contiguous TEUs known as phrase pairs. These phrase pairs are structured by means of subsumption relations, whereby smaller phrase pairs are composed into bigger ones. HATs structure the recursive composition relations in such a way that maximal decompositions are obtained. The construction of HATs can be thought of as a bottom-up process, whereby first non-decomposable atomic phrase pairs are induced from the word alignment, and next these are recursively composed into bigger phrase pairs up to the sentence level.

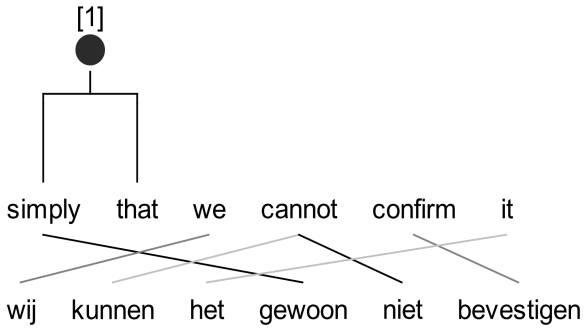
The process is illustrated for the HAT of Figure 4.1b. In Figure 4.2 this HAT is composed from the phrase pairs induced by the word alignment in steps 1–5. In step 1 the phrase pair ⟨“simply that”, “gewoon”⟩ is added.[1] Next, in steps 2, the phrase pair ⟨“we”, “wij”⟩ is added. Similarly, in step 3 and 4 the phrase pairs ⟨“confirm”, bevestigen⟩ and ⟨“it”, “het”⟩ are added. For brevity, steps 3 and 4 are omitted in the Figure. Finally, in step 5 the HAT is completed by composing the discontiguous TEU ⟨“cannot”, “kunnen . . . niet”⟩ with the phrase pairs added in steps 1–4 into a phrase pair spanning the entire sentence pair. During the process of constructing a HAT, when a phrase pair is added, it is enriched with a set-permutation label, which describes exactly its recursive decomposition into smaller phrases and single words. Notice how the atomic phrase pairs (steps 1–4) in Figure 4.2 are decorated with a label “[1]”, since they all map to one element on the target side. In contrast, the phrase pair spanning the entire sentence pair added in step 5 has a complex label $[4, 1\{2, 5\}, 6, 3]$, indicating its composition from the four reordered smaller phase pairs in relative target positions 4, 1, 6 and 3 as well as the discontiguous TEU ⟨“cannot”, “kunnen . . . niet”⟩ that connects position 3 in the source to positions 2 and 5 in the target.[2]

4.2.1 How HATs satisfy the criteria for effective and complete hierarchical alignment representations

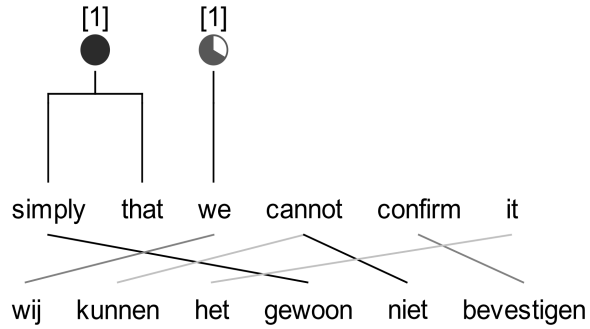
HATs satisfy the first criterion of adequately representing the TEUs as well as their compositional mapping relations, by means of the (hierarchical) structure of the HATs in combination with the set-permutation labels. The labels give a precise description of the reordering context of TEUs, and can therefore be used to provide adequate

[1] In this step, the unaligned word “that” is grouped with “simply”. But, in general, unaligned words can be considered free to bind left or right adjacent atomic phrases. Provided these words and their positions are remembered, it is therefore generally not necessary to explicitly represent their grouping within HATs.

[2] Note that the unaligned source word “that” is considered to be grouped with an adjacent aligned word, here “simply”, and therefore has no own position in the source.

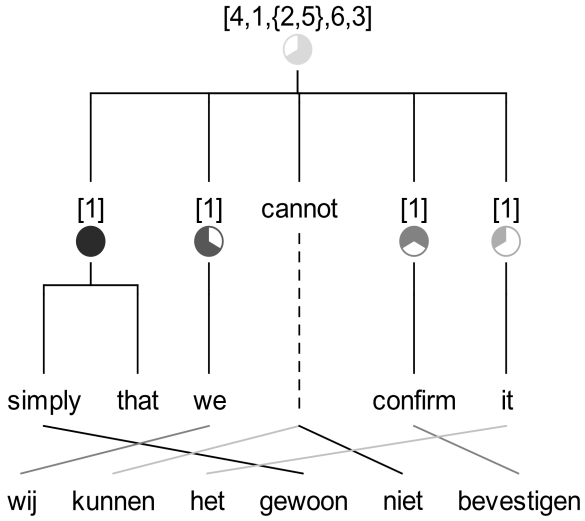


Step 1



Step 2

...



Step 5

Figure 4.2: The stepwise composition of a HAT. Steps 3 and 4 are ommitted, for brevity.

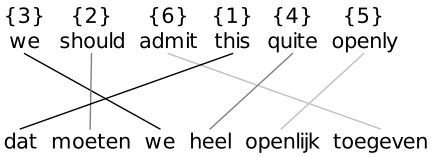
	Alignment Coverage			Representational Completeness
Tree type	covers binarizable alignments	covers bijective alignments	covers general alignments	represents mapping relations
Binary trees (NF-ITG based)	✓	✗	✗	✓
Permutation trees	✓	✓	✗	✓
Normalized decomposition trees	✓	✓	✓	✗
Hierarchical alignment trees	✓	✓	✓	✓

Table 4.2: Schematic view of capabilities of alternative (hierarchical) tree models for the representation of translation equivalence as indicated by (word) alignments .

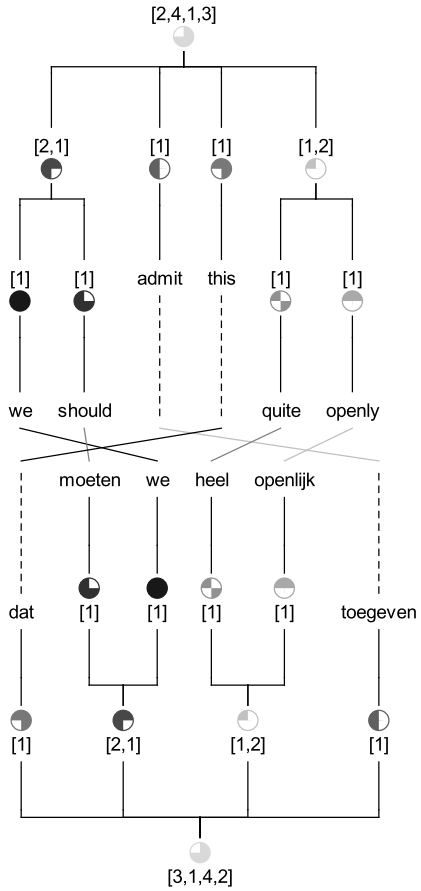
context-describing labels for rules. In chapter 5 this is explored extensively. Thus the second criterion is also met. The same chapter also shows how the third criterion is fulfilled: set-permutation labels are bucketed into coarser categories denoting broad reordering complexity classes. Different granularities are explored for the labels, albeit following a heuristic rather than learning approach in the formation of the buckets. Nevertheless the effectiveness of supporting labels at different granularities becomes clear, and the heuristically defined labels already significantly improve hierarchical machine translation, showing about 1 BLEU point improvement on a Chinese–English translation task[3]. Lastly this chapter also shows how without any extra information outside the word alignment the HATs provide labels that are outperforming both unlabeled hierarchical translation (Chiang, 2005) but also syntax-based syntax-augmented machine translation (Zollmann and Venugopal, 2006), hence demonstrating how also the fourth and last criterion is met.

We now zoom out again to our main goal: making translation more coherent particularly with respect to word order. In chapter 5 we will be offer empirical evidence that this goal is achieved. Here we refer back to **Hypothesis 4.1**, our main hypothesis in this chapter, introduced in the last section. HATs are exactly the type of representation this thesis demands, and are shown to satisfy all the criteria a complete representation of hierarchical translation equivalence must fulfill. Thus building on the foundation of HATs as our representation of the hierarchical bilingual structure of word alignments there is a high change to achieve main goal.

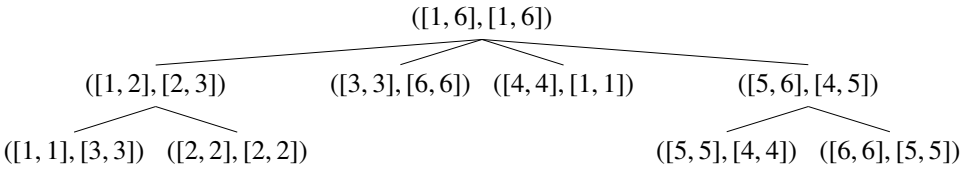
[3]This could be further improved upon by using a learning approach that starts from the set-permutation labels, and learns labels of appropriate granularity by appropriate incremental split and merge steps or by application of (cross-validating) Expectation Maximization (Dempster et al., 1977) somewhat similar to the work by Petrov and Klein (2007) for parsing and Mylonakis and Sima'an (2011) for translation. Current follow-up work in our group goes in this direction.



(a) Non-binarizable bijective word alignment



(b) Permutation tree (PET) for (a).



(c) Normalized decomposition tree (NDT) Zhang et al. (2008a) for (a).

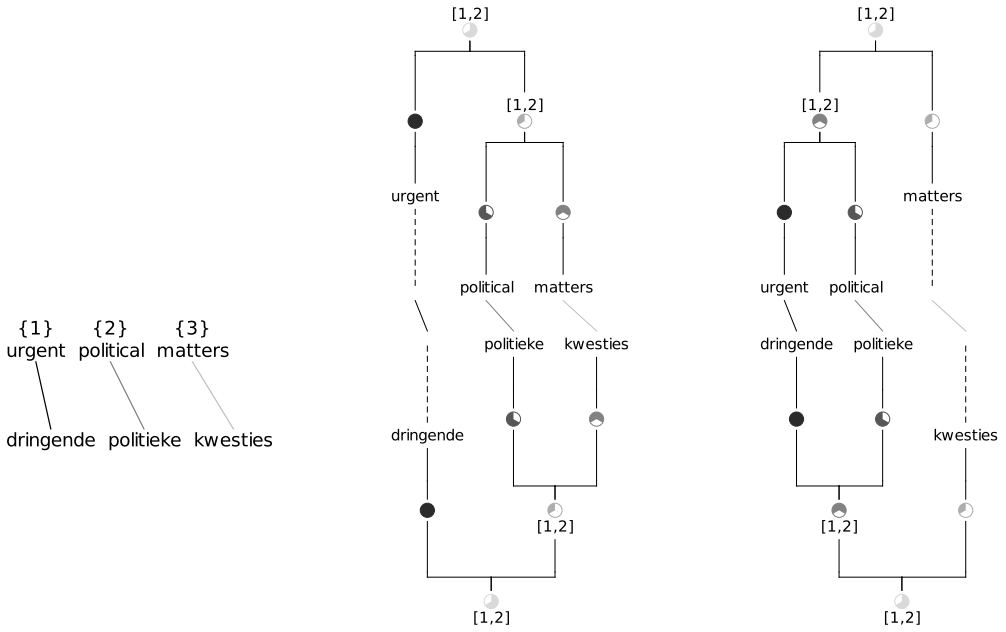
Figure 4.3: A word alignment (a), with non-binarizable bijective word mappings (permutation) and its corresponding permutation tree (PET) (b) and normalized decomposition tree (NDT) (c). In (b) permutation labels, such as [2,4,1,3] denote the local relative reordering mapping at every node. Note that the NDT representation (c) does not explicitly state the mapping relations, in contrast to the PET representation. It instead specifies pairs of source/target span ranges, such as ([1,6] ,[1,6]), that are translation equivalent. While for PETs the mapping relations are in principle still retrievable from the NDT by reasoning, in the case of NDTs for general non-bijective (discontiguous) word alignments, even this reconstruction is no longer possible and information about the mapping is lost.

4.2.2 How HATs relate to other decompositions of word alignments

HATs can be seen as an extension of normalized decomposition trees (NDTs) (Zhang et al., 2008a) and PETs (Gildea et al., 2006). NDTs are an effective formalism for compactly representing the phrase pairs induced by a word alignment and enabling their very efficient (linear time) extraction. PETs are an equally effective formalism for representing hierarchical translation equivalence over permutations (a subclass of alignments), but are limited to bijective mapping relations. Figure 4.3a shows an example of a word alignment with a bijective non-binarizable mapping. This type of alignment is thus representable by both NDTs and PETs, and the corresponding PET is shown in Figure 4.3b and the corresponding NDT is shown in Figure 4.3c.

PETs have an important property not shared by NDTs : they completely retain all information of the original permutation, so that by incremental evaluation of the node operators that original permutation is easily reconstructed. We illustrate this reconstruction using the example in Figure 4.3b. Starting from the permutation label [**2**, 4, 1, 3] at the top, and expanding the first child **2** with the child permutation [2, 1], shifting the numbers as necessary, we arrive at the (intermediate) permutation [3, 2, 5, 1, **4**]. Finally expanding the last child **4** in the intermediate permutation (originally 3 before expansion) with the child permutation [1, 2] and shifting the numbers again we get back the original permutation [3, 2, 6, 1, 4, 5]. In contrast, NDTs represent only the set of phrase pairs and their subsumption relations, but not the information about the reordering. Figure 4.3c illustrates this. Thus, NDTs do not contain the reordering information and therefore are not equivalent to the original word alignments. This shows that PETs as a representation are richer than NDTs in that they keep (permutation) operators that describe the local mapping relations at the tree nodes, which are absent in NDTs. Therefore NDTs are essentially not fully an extension of PETs. Both PETs and NDTs focus on canonical maximal decompositions, as opposed to explicitly representing all induced TEUs.

HATs, like PETs, differ from NDTs in an important way in that they preserve all information of the original word alignments underlying them. In Figure 4.1a we see a discontiguous word alignment, with in Figure 4.1b the corresponding HAT and in Figure 4.1c the corresponding NDT. Note how unlike a PET the HAT is capable of representing this word alignment, while unlike an NDT (and like a PET) it also manages to completely represent all involved mapping relations. The *set-permutation* labels, which generalize permutations to arbitrary m-n mappings, make this possible. These labels keep information about the mapping relation that is not necessarily present from only the synchronous tree pairs. HATs are also powerful because, equal to NDTs they decompose phrase pairs in a maximal way, which is an essential but not sufficient condition to capture all contiguous TEUs in a maximally compact, hierarchically structured way. The situation is summarized in Table 4.1: HATs, unlike NDTs and like PETs and binary trees are capable of completely representing mapping relations. But HATs, unlike PETs and binary trees are also capable of covering general word



(a) Monontone word alignment

(b) HAT 1

(c) HAT 2

Figure 4.4: Example of a simple monotone word alignment from Europarl English–Dutch, that induces two alternative HATs.

alignments. Thus HATs merge the strong properties of both PETs and NDTs into a representation that unlike all other representations has both full *alignment coverage* as well as *representational completeness*.

Different from NDTs, HATs explicitly capture all contiguous TEUs, rather than restricting to one canonical form decomposition of the word alignment and leaving the representation of alternative maximal decompositions implicit. For example, in Figure 4.4 a simple monotone word alignment induces two alternative HATs, which are compactly stored as a forest by the HAT parser. This small detail will turn out to be important when HATs are used to extract reordering labeled translation rules, as discussed in chapter 4. It is also of theoretical importance when HATs are used as an equivalent representation of alignment induced contiguous translation equivalence. This is necessary as part of a bigger effort to exactly measure alignment coverage by the intersection of TEUs induced by the word alignments and those permitted by a synchronous grammar given the (aligned) sentence pair.

But HATs can help to quantify meaningful metrics of reordering complexity that go beyond any specific grammar formalism, as a result of focusing on the more general properties of the hierarchical translation equivalence itself, rather than concentrating on

any one particular grammar formalism. These applications of HATs are discussed in chapter 5. Finally HATs can also help to give a much better qualitative understanding of hierarchical translation equivalence and its empirical underlying structure. In chapter 6 we discuss how the HATs representation forms the basis for a rich and illuminating visualization of hierarchical translation equivalence, permitting much deeper understanding of hierarchical translation equivalence than non-hierarchical visualization tools have allowed so far.

4.3 Existing algorithms for decomposition of translation equivalence

In what follows we describe two foundational existing approaches to the decomposition and recursive representation of translation equivalence. We start with permutations and *permutation trees* (PETs) (Gildea et al., 2006) , which offer compact representations of bijective translation equivalence. These representations are not strong enough to capture general translation equivalence. Yet the success of phrase-based statistical machine translation for many language pairs proves that just bijective reordering, with discontinuity limited to embedding within phrases, can be still highly effective. This makes permutation trees a valid starting point for studying representations of translation equivalence, which is further reinforced by recent findings concerning their effectiveness for translation evaluation (Stanojević and Sima'an, 2014b,c). After first introducing permutations and permutation trees, we next introduce *normalized decomposition trees* (NDTs) (Zhang et al., 2008a) , which are an extension of PETs for the more general case of phrase pairs (general continuous TEUs). While very useful for efficient phrase extraction, these representations lose one attractive feature of PETs, namely the operator labels. These labels are, as we will see in the next chapter, crucial for effective learning of desirable and coherent reordering in hierarchical statistical machine translation. This motivates our extension of PETs and NDTs to Hierarchical Alignment Trees, basically NDTs enriched with set-permutation labels, that precisely describe the hierarchical mapping operations defining the phrase pairs. But before we make that last step, we now first introduce PETs and NDTs, essential foundations for our work made by researchers before us.

4.3.1 Permutation Trees

Permutations

Permutations are mathematical constructs that describe how a sequence of elements gets *permuted* into a new reordered sequence of the same elements in a different order. The reordering is such that every element maps to exactly one new position in the reordered sequence. This is mathematically known as a *bijective mapping* from

a sequence X to the permuted sequence Y, a mapping that is *one-to-one* and *onto*.[4] A permutation of length n denotes a reordering function for a domain of n elements that can be described by a reordering of the integers $[1, 2, ..., n]$[5]. Every integer in the original range occurs exactly once in the reordered range as well. Furthermore every element (integer) y at position x in the reordered sequence Y is to be interpreted as specifying the position y to which x maps in the reordered sequence Y. As a simple example the permutation $[3, 1, 4, 2]$ states that the first element maps to the third position in the reordered sequence, the second element to the first position, the third element to the fourth position and the fourth element to the second position.

Decomposing permutations

Some permutations can be decomposed (Albert et al., 2005). Testing whether some decomposition is possible amounts to testing whether for a permutation with elements at positions $[1, ..., n]$ there exists any sub-range $[k, m]$ such that the subsequence s of elements at positions $[i, i + 1, ..., j - 1, j]$ map to a proper range of consecutive positions $[l, ..., u]$. As Albert et al. (2005) remark, this can be tested by taking the minimum $m_\pi(i, j)$ and maximum $M_\pi(i, j)$ position mapped to by s and checking whether it holds that $M_\pi(i, j) - m_\pi(i, j) = j - i$, which is equivalent to saying the maximum minus the minimum is equal to the length of the subsequence minus one. In their terminology a non-decomposable permutation is known as a *simple permutation*. The work by (Gildea et al., 2006) builds on the work of Albert et al. (2005) and introduces an efficient algorithm to find a *maximal decomposition* of a permutation in $O(n \log n)$ time. A maximal decomposition, sometimes called *maximal factorization*, is a decomposition that *maximally* decomposes the permutation, so that there is no sub-permutation for which further decomposition is possible. A maximal decomposition of a permutation can be represented as a tree, a so-called *permutation tree (*PET*)* in which every node of the tree has an operator which describes how the (possibly complex) children are to be reordered to get the order of the parent. Such a PET encodes the reordering in a recursive manner, meaning that the original permutation can be reconstructed by recursively applying the permutation at the parent node to the partial sequence at the child nodes, starting from the leaf nodes and going bottom-up to the top of the PET. Examples of PETs are shown in Figure 4.5.

[4] A *one-to-one* mapping from domain X to codomain Y is a mapping that never maps multiple elements from X to Y. A *onto* mapping from domain X to codomain Y is a mapping such that every element in Y has a corresponding element in X.

[5] It is a mathematical convention in formal descriptions of permutations to work with indices $[1, 2, ..., n]$, however in implementations it is more common to work with indices $[0, 1, ..., n - 1]$, staying closer to the way arrays are indexed. It is important to state that both notations are essentially equivalent, in the same way indexing an array starting from 1 as in certain programming languages is equivalent to the more common way of starting from 0.

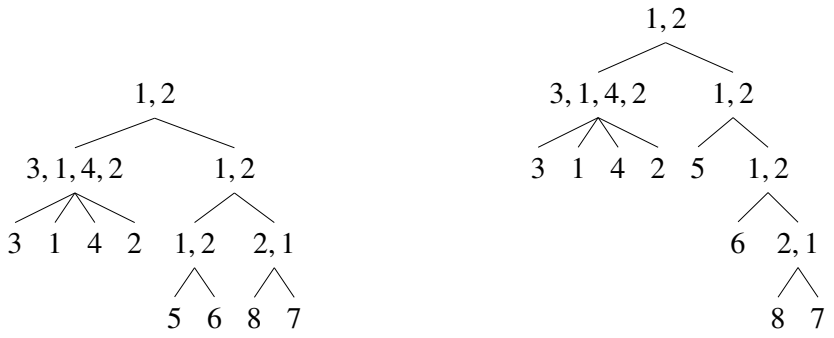


Figure 4.5: Two alternative permutation trees (PETs) corresponding to alternative maximal decompositions of the permutation $[3, 1, 4, 2, 5, 6, 8, 7]$. In total there are five alternative maximal decompositions for this permutation with corresponding PETs.

Two components characterizing permutation trees

Albert et al. (2005) explain how permutations are decomposed in what they call *blocks* and what we refer to as *sub-permutations*. Permutations specify a bijective mapping between the elements of a sequence and a reordered form of that same sequence. PETs visualize a specific maximal decomposition of this bijective mapping between sequences, by representing the permutation as a tree whose nodes correspond to the sub-permutations of the decomposition and whose node labels specify the correspondence (mapping) relations between the children of the nodes. A PET can be understood as specifying a pair of linked trees, of which only the first tree that specifies the mapping from the original to the reordered sequence is represented, and the second tree representing the reordered sequence and its mapping back to the original is left implicit. A PET dissects the original permutation into two components that make the hidden structure of the permutation explicit, and which are only in their combination sufficient to fully characterize the original permutation and allow its reconstruction. These components are:

1. **Subsumption relation**: the (tree) structure of the permutation tree describes exactly how the contained sub-permutations recursively decompose into parts. Importantly, he structure by itself describes just the subsumption (parent-child) relationships, it does not specify mapping relations between children.

2. **Mappping relation**: the mapping relation between nodes in the permutation tree, and the thereby implied recursive reordering characterizing the permutation, is described by the permutation labels on the nodes.

Multiple maximal decompositions per permutation

Figure 4.5 in fact shows two alternative PETs that correspond to different maximal decompositions of the same permutation. In general there are typically many maximal decompositions of any given permutation. These arise from different alternative

maximal decompositions of monotone and inverted sequences. A fully monotone or fully inverted sequence of length n can be split at any of $n - 1$ positions during the first step of decomposition, starting from the root. Then each of the two resulting subsequences of length m_1 and m_2 at the child nodes can again be recursively split in $m_1 - 1$ and $m_2 - 1$ ways respectively each. This leads to an exponential number of decompositions. Computing all maximal decompositions can still be effectively done with a chart-parsing algorithm, but takes in the worst case $O(n^3)$ time. However, for certain purposes, such as factoring grammars and extracting minimal grammars (Gildea et al., 2006; Zhang and Gildea, 2007; Huang et al., 2009), a single *canonical* decomposition is sufficient, and finding an optimal way to compute it is very useful. The next section reviews existing efficient algorithms for decomposing permutations, focusing on algorithms that find only a single maximal decomposition as opposed to the entire forest of maximal decompositions.

4.3.2 Efficient algorithms for decomposing permutations

4.3.3 Sorting-based algorithm

Gildea et al. (2006) propose a first efficient algorithm to find a maximal decomposition for any permutation. The algorithm has a computational complexity of $O(n \log n)$. As the authors remark, their algorithm is reminiscent of Merge Sort (Cormen et al., 2009). In the outer loop of the algorithm, spans of increasing size are visited. The idea is to look for valid reductions within these spans of increasing size. This looking for possibly multiple nested reductions within a certain span is called the "*merge*" step of the algorithm. Each added reduction must be extending any properly contained reductions made for smaller and contained spans visited at earlier iterations. At the first iteration the span size is 1, and at every iteration the span size is doubled. The visited spans are non-overlapping, which means that at every iteration the number of spans is halved as their span size is doubled. This means that in the ideal case where the permutation length is a power of 2, exactly $2 \times n$ spans will be visited in the outer loop, in general the number of spans may differ slightly but will always be $O(n)$.

The algorithm works with two arrays during the *merge* steps. One array h maps from vertical to horizontal positions within the current subsequence. A second array v maps from horizontal to vertical positions within the subsequence. The arrays are formed by sorting the indices in the original permutation for the processed span to follow the order of the input permutation as much as possible, under the constraint that only the indices within the considered span can be moved. The array v is essentially the inverse of h, meaning that $\forall i : v[h[i]] = i$, which is exploited by only sorting h and then setting the values for v using this relation. It is important to understand the motivation of working with the indices within the processed subsequence stored in h

and v, which need to be (re-) sorted when processed by merge.[6]

Apart from (re-)sorting, central to the working of the algorithm is that it keeps track of two pairs of boundaries, one in the horizontal and one in the vertical direction. The boundaries start at the middle of the subsequence being considered for reduction and are iteratively moved towards the beginning and end of this subsequence. When the positions read in the vertical array are outside the range of the horizontal boundaries, these boundaries are extended to include them. Similarly, when positions read in the horizontal array are outside the vertical boundaries, these are extended. Alternating between vertical and horizontal scanning, the different boundaries are extended until none of them changes. Meanwhile, during each iteration of this scanning procedure it is checked that the distance between the current vertical boundary values in the subsequence permutation is not smaller than the distance between the input permutation values for those boundary values. If the distance is smaller, the subsequence misses values and no (further) reduction can be possible for the subsequence under consideration. If this reduction check does not yield a negative result before the boundaries stop changing, this means a valid reduction has been found by the algorithm. If a valid reduction is made for a subsequence, more reductions are attempted for that subsequence, by further extending the boundaries starting from the boundaries of the previous reduction. Globally the algorithm also keeps track of minimum and maximum positions of reductions that have already been made. These minimum positions of reductions at smaller spans are then used at containing larger spans to assure that reductions being considered directly skip on to reductions that extend smaller reductions that have already be formed before. As a result of these smart tricks and extensive bookkeeping it can be proven that the algorithm runs in $O(n \log n)$ time.

Looking once more globally at the working of the algorithm it is natural to wonder why working with the within-subsequence sorted indices is necessary, and why it could not be possible to work with the original permutation values directly. There are two reasons for this. The main motivation to work with the within-subsequence indices rather than with the original permutation values, is that this permits the direct mapping from within-subsequence permutation values to increased boundaries that can be mapped back and forth in horizontal and vertical direction, enabling the efficient boundary extension within the scan algorithm which would not be possible otherwise. To understand why sorting is also necessary note that the spans that are reduced do not necessarily need to span the entire subsequence, and the fact that h and v are sorted permits to gradually expand the boundaries from the center of the subsequence, while guaranteeing that whenever a reduction is found, smaller valid reductions either do not exist or else have already been added.

As we will see next though, this algorithm is not the best possible yet, besides

[6]Since sorting is redone at every iteration this also could influence computational complexity. However, resorting h for a larger subsequence amounts to merging the partially sorted lists of the subsumed subsequences, which can be done in linear time. Hence the sorting does not, increase the total complexity of the algorithm beyond $O(n \log n)$.

not being the most intuitive. In the next paragraph we will shortly review a linear time algorithm which was developed in a follow up by two of the same authors. This algorithm keeps track of the lowest and highest value occurring within every span, functions which can be efficiently computed incrementally by a dynamic programming algorithm. Using these functions in combination with a stack-based shift-reduce style algorithm, subsequence-relative permutations and constant resorting are no longer necessary, as the reduction test can be evaluated directly and efficiently for every considered subsequence.

4.3.4 Stack based algorithms

Zhang and Gildea (2007) introduce a linear time algorithm for the decomposition of permutations through modification of the algorithm of (Uno and Yagiura, 2000) for the closely related problem of finding all common intervals of two permutations. Their algorithm can be seen as a refinement of a simpler stack-based shift-reduce algorithm. This simpler algorithm scans the input from left-to right, and after addition of every input element scanned and added to the stack, tests subsequences that start at the end of the stack and are incrementally extended to the beginning as long as no reduction is found. Whenever a reduction is possible, which is the case when the total sum of elements of all to be merged subsequences equals the maximum minus the minimum element over all those subsequences (basically again the standard reduction test introduced by (Albert et al., 2005)), this reduction is printed and the subsequences involved in the reduction are taken from the stack, and replaced by one new subsequence formed by gluing them together.[7]

The authors next introduce a function that describes the reducibility of a span $[x, y]$ with $(1 \leq x \leq y \leq n)$.

$$f(x, y) = u(x, y) - l(x, y) - (y - x) \tag{4.1}$$

with

$$l(x, y) = min_{i \epsilon [x,y]} \pi(i)$$

$$u(x, y) = max_{i \epsilon [x,y]} \pi(i)$$

$l(x, y)$ gives the minimum amongst the permuted numbers in the range $[x, y]$ and $u(x, y)$ gives the maximum within that range. Note that to give some more intuitive meaning to u and l they may be thought of as abbreviations for *low* and *up*.

[7] Since reduced sequences are sorted, only the first and last element need to be consulted to get their minimum and maximum. Hence when shifting down the stack, for both simple subsequences of one element and for compound subsequences of multiple elements, the minimum and maximum position and the total length for the currently shifted range, can be updated in constant time.

This function $f(x, y)$, encodes the reduction test, and is non-negative but not monotonic in general. A reduction is possible only for spans where the function is zero.

Armed with this function the authors remark that if tried out ranges could be ordered in such a way that the function is monotonically decreasing, this would prevent the need for (repeatedly) testing invalid reductions, as at every iteration all valid reductions would be added first and only once a value of $f(x, y) > 0$ would need to be seen to stop all reduction attempts for that iteration.

That this is possible is shown by (Uno and Yagiura, 2000). The authors implicitly use three lemmas which are made explicit in the next follow-up paper, which generalizes the linear-time algorithm for permutations to work with full-fledged word alignments (set-permutations). These lemmas are as follows:

Lemma 4.1. *If $x_1 < x_2 < y$ and $f(x_1, y) < f(x_2, y)$, then for all $y' \geq y$, $f(x_2, y) > 0$ (i.e. $[x_2, y]$ is not a permutation).*

Lemma 4.2. *If $x_1 < x_2 < y$ and $u(x_1, y-1) > u(x_2, y-1)$, but $u(x_1, y) = u(x_2, y)$ then for all $y' \geq y$, $f(x_2, y') > 0$ (i.e. $[x_2, y']$ is not a permutation).*

and very similar for l:

Lemma 4.3. *If $x_1 < x_2 < y$ and $l(x_1, y-1) < l(x_2, y-1)$, but $l(x_1, y) = l(x_2, y)$ then for all $y' \geq y$, $f(x_2, y') > 0$ (i.e. $[x_2, y']$ is not a permutation).*[8]

Next we will give an intuitive understanding of these lemmas without proving their correctness, see (Uno and Yagiura, 2000) for a more formal analysis. Lemma 4.1 states that if there are two spans, both ending at the same upper boundary y, but one having a smaller lower boundary x_1 and a smaller f-value, then it follows that any span starting from the bigger lower boundary x_2 and extending to any upper boundary has an f-value bigger than 0, i.e. non-reducible. Why is this the case? Informally, the fact that the span extending to x_1 has a lower f-value means that x_1 contains an element that must be included to form a permutation that can be reduced. Hence, any span that starts from x_2 and does not include the element from x_1 will always be incomplete, no matter how far y is stretched on the right side.

Lemma 4.2 can be understood as follows. We have $span_1 = [x_1, y-1]$ and $span_2 = [x_2, y-1]$, both spans sharing the same right boundary but $span_1$ extending further to the left than $span_2$. Now it holds that $span_1$ has a higher maximum ($u(x_1, y-1)$) which we will call max_1 then $span_2$ ($u(x_2, y-1)$) , which we will call max_2. But if we extend both spans one to the right, their maximum, which we will call max_{shared}, becomes equal. Simple reasoning shows that this implies that $span_1$ must contain an element max_1 which is also required but not contained in any span equal to $span_2$ or extending $span_2$ further on the right. As Lemma 4.3 is completely analogous to Lemma 4.2, its correctness also be understood analogously, and therefore is not further explained here.

[8] This second analogous lemma for l is in fact not spelled out completely in Zhang and Gildea (2007), but we do state it here explicitly for reasons of clarity.

These lemmas play a central role in the timely elimination of all spans that will never yield valid reductions, preserving the invariant that the f-value of spans encoded on the stack is monotonically decreasing, and hence enabling linear time performance of the algorithm.

The lemmas are exploited in the algorithm by finding at every extension of the stack by a new element y *pivot values* for x, which are the leftmost values for x in which either $u(x, y-1) < u(x, y)$ or $l(x, y-1) > l(x, y)$ so that the value of l or u gets *updated* for these spans by extending them to include y. These pivot values are then used by applying the lemmas described above and efficiently eliminating all x values, that will yield spans $[x, y]$ that can never be reduced. The fact that such x values must be at or right of the left-most pivot value, makes searching for them efficient. This avoids the need to repeatedly test different right extensions of spans that are simply missing something on the left and hence can never be completed. When taking care of certain performance affecting details in the implementation this gives a correct $O(n)$ maximal decomposition algorithm. For a proof of the algorithm complexity as well as more details see Zhang and Gildea (2007).

4.3.5 Normalized Decomposition Trees

Normalized decomposition trees (NDTs) (Zhang et al., 2008a) provide a compact representation for maximal decompositions (maximal decompositions) of word alignments into a set of nested phrase pairs. *Maximal decomposition* means that the decomposition is constructed such that it is maximally compositional, which implies that the word alignment is recursively decomposed into a minimal number of parts at every level, yielding a maximally binary decomposition tree. *normalized* implies that NDTs work with a canonical-form, left-branching[9] decomposition in order to deal with the ambiguity caused by monotonic and inverted parts of the word alignment, which can induce (exponentially many) alternative possible maximal decompositions. One important observation is that non-canonical form maximal decompositions can still be efficiently constructed from NDTs, but they are not explicitly captured by this representation. Representing all induced phrase pairs explicitly requires a (packed) forest rather than a tree representation, but since the original goal of NDTs was mainly fast (linear time) extraction of synchronous rules, such an explicit representation was unnecessary and even undesirable for efficiency reasons.

In Figure 4.6 we show an example of a word alignment, taken over from (Zhang et al., 2008a). The position sets above the English words at the top of the figure correspond to the positions of French words on the bottom to which the English words are aligned. The example is interesting since the word alignment is rather complex, and it yields two alternative maximal decompositions, shown in Figure 4.7 and Figure 4.8.

[9]NDTs give preference to *left strong intervals*, this corresponds to a preference to left-branching trees.

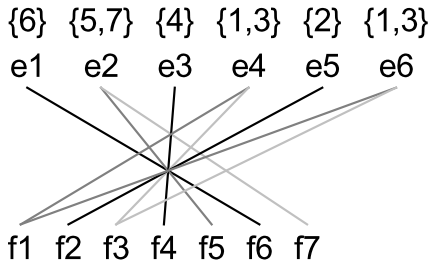


Figure 4.6: Example of word alignment taken from (Zhang et al., 2008a)

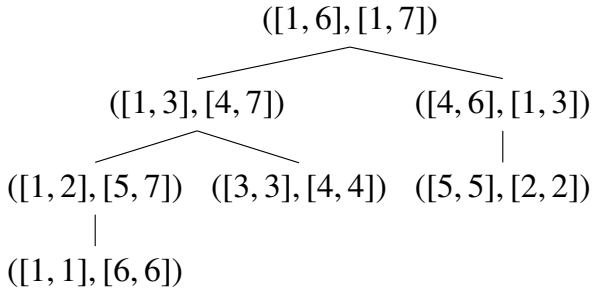


Figure 4.7: Normalized decomposition tree (Zhang et al., 2008a) for the example from Figure 4.6.

Figure 4.7 shows the NDT for the alignment of Figure 4.6. The tree contains nodes with labels showing pairs of ranges on the source and target side, corresponding to phrase pairs. For example, the root node has label ([1,6],[1,7]) which means the root node is the phrase that spans the source positions 1 to 6 and the target positions 1 to 7. The NDT tells how this can be decomposed minimally as two subsumed phrases ([1,3],[4,7]) and ([4,6],[1,3]). NDTs are formed using a linear time algorithm that processes the word-alignment from left to right. In case of alternative maximal decompositions it chooses these decompositions that yield phrase pairs corresponding to *left strong intervals*, those intervals that do not overlap with any others on the left. Figure 4.8 shows the alternative maximal decomposition for the alignment of Figure 4.6. In this maximal decomposition the phrase pair ([3,3],[4,4]) is composed

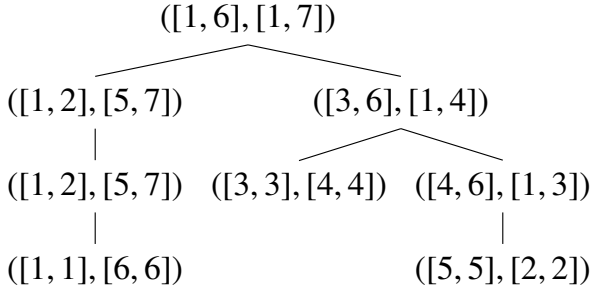


Figure 4.8: Alternative maximal decomposition for the example from Figure 4.6.

with the phrase pair to the right instead of left as in the original NDT. While this is also a valid maximal decomposition it is not an NDT, as it is an alternative, non-canonical decomposition of the word alignment. Finally it is important to notice that one limitation of NDTs is that they do not make explicit the mapping relations between phrase pairs, only their relative embedding. For example in the NDT of Figure 4.6, the top phrase pair is decomposed into two parts that are inverted, and so is its left child. In the alternative maximal decomposition of Figure 4.8, the top phrase pair and its right child are now the inverted nodes. This information is not present in the NDT, and in this case looking back at the original word alignment plus reasoning is necessary to derive it. But having this information explicit is important when we want to reason about the reordering context of synchronous rules. This is our main reason to extend NDTs and permutation trees (PETs), into a generalized formalism called hierarchical alignment trees (HATs), which subsumes both and makes the (reordering) mapping relation at the nodes for all phrase pairs induced by the word alignment explicit.

Efficient algorithms for computing Normalized Decomposition Trees

With some adaptations and generalizations the stack-based algorithms for decomposing PETs are extended in (Zhang et al., 2008a) to work effectively for general word alignments as well, giving algorithms that effectively find maximal decompositions of word alignments as sets of minimal phrase pairs. In the case of permutations the difference between the maximum and minimum element of a sequence in comparison to the sequence lengths yields an adequate reduction test. But Zhang et al. (2008a) note that for phrase pairs where there can be many to many alignments, assuring all involved alignment links are also included becomes necessary. This suggests a new reduction test that is based on the counting of links. By introducing a function that counts the total number of links for a certain span starting from the beginning, and noting that by subtraction of two counts the total link count for any arbitrary span can be efficiently computed for source or target side, the authors lay the foundation for an effective generalization of the earlier algorithms for decomposition of PETs to phrase pairs.

The authors introduce two functions:

$$F_c(j) = |\{(i', j') \epsilon A \mid j' \leq j\}|$$

$$E_c(i) = |\{(i', j') \epsilon A \mid i' \leq i\}|$$

which count the number of links contained in the subsequences E and F starting on the left and running up to the i-th or j-th word, for the source and target side respectively. The difference $F_c(u) - F_c(l-1)$ counts the total number of links in the source range $[l, u]$ while $E_c(y) - E_c(x-1)$ counts the total number of links in the target range $[x, y]$.

Based on these new functions, the new reduction test function becomes:

$$F(x,y) = F_c(u(x,y)) - F_c(l(x,y)-1) - (E_c(y) - E_c(x-1)) \tag{4.2}$$

The first part of this formula. $F_c(u(x,y)) - F_c(l(x,y)-1)$ computes the number of links on the target side between the minimum and maximum target position mapped to in the source range $[x,y]$. The second part of the formula, $(E_c(y) - E_c(x-1))$, counts the number of links on the source side, in the source range $[x,y]$. Now the definition of phrase pairs states that we have only a phrase pair exactly when all links started from a source range, are exactly included by those links produced by the mapped-to target range, and nothing more or less. This is the case if and only if the first part of the formula is exactly equal to the second part, so that the value of F becomes zero. This informally shows that the adapted formula, exactly tests for the existence of phrase pairs, and thus succeeds to generalize the central formula from the permutation decomposition algorithm to the decomposition of general word alignments.

There are still some minor differences between the original permutation decomposition algorithms and their generalized forms for word alignments based on the above generalized formula for F. One difference is that the generalized form algorithm described in (Zhang et al., 2008a) explicitly described "*steps*" of l and u, which are sets of ranges with consecutive starting points and same end points sharing the same l or u value. These "*steps*" can in fact be explicitly represented, and in a way help to make the implementation more structured in the representation of u and l and its efficient updating. Another minor difference is that (Zhang et al., 2008a) explicitly works with two separate pivot values which are also important for the proofs of correctness of the algorithm, while in the original description for decomposition of permutations only the left-most pivot value is explicitly used in the description of the algorithm. This is not an essential difference in the working of the algorithm but rather a minor difference in the description of the algorithms it seems though. Overall the two papers (Zhang et al., 2008a) and Zhang and Gildea (2007) are complementary for a better understanding of the overlapping ideas and methods presented in both papers, since both papers are quite dense in their descriptions of the actual algorithms. Possibly the most effective way to get a detailed understanding of the algorithms is to reimplement them.[10] We end our overview of existing work on permutation and alignment decompositions here, and encourage interested readers to look at the original publications for more details about applications, performance, correctness and implementation.

4.4 Hierarchical Translation Equivalence

An essential element of a complete notion of translation equivalence is the compositional structure of the TEUs. The important insight is that the contiguous TEUs

[10] Our reimplementation is available as part of the project at https://bitbucket.org/teamwildtreechase/hatparsing

(phrase pairs) induced by word alignments by definition take part in *subsumption* relations, formally defined in section 2.3.2, definition 2.3.2, as soon as any phrase pair smaller than the full sentence pair exists. This is almost always the case, and typically there exist many induced smaller phrase pairs and these phrase pairs exhibit a rich hierarchical subsumption structure.

When the complete set of TEUs and their subsumption relations is given, much is known about the hierarchical alignment relation, but not necessarily enough to completely retrieve the original word alignment. This is illustrated by the normalized decomposition tree (NDT) in Figure 4.1c, which represents all contiguous TEUs and their subsumption relations, but fails to represent the existence of the remaining discontiguously linked words ⟨"cannot", "kunnen ... niet"⟩ and their role in completing the word alignment. NDTs also fail to represent the internal word alignments inside phrase pairs, which is a second reason why they fail to preserve enough information to allow reconstruction of the original word alignment.

In section 4.3.1 we saw that in the case of PETs besides the *subsumption relation* a second component being the *mapping relation* is required to fully characterize the original permutation and allow its reconstruction. While the node operators of PETs capture the thus defined mapping relation adequately, PETs can only capture bijective translation equivalence relations. When we generalize to general word alignments, the requirement of having both these components present in order to get a complete representation of the original word alignment and as such of hierarchical translation equivalence, remains. NDTs (Zhang et al., 2008a) represent only the subsumption relation, and not the mapping relation, and are therefore not adequate as complete representations of hierarchical translation equivalence induced by word alignments.[11]

Using again the components *subsumption relation* and *mapping relation* that characterize the ability of PETs to completely characterize hierarchical translation equivalence for bijective word alignments, through generalization we can now define hierarchical translation equivalence for general word alignments:

Definition 4.4.1. *Hierarchical Translation Equivalence*
Hierarchical Translation Equivalence *for a word alignment is defined as the union of the full sets of:*

1. **Translation equivalents** *under the notion of contiguous translation equivalence.*[12]

[11]Note that in normalized decomposition trees, as might be expected the subsumption or decomposition relation is implicitly available, with the limitation that only the canonical form decomposition is given. The mapping relation however, is not available, not even implicitly, and as a consequence normalized decomposition trees cannot distinguish between equivalent sets of phrase pairs that exhibit completely different internal word alignments.

[12]As discussed in section 2.3.2 there are two notions of translation equivalence: *contiguous translation equivalence* (phrase pairs) and *general translation equivalence* (all induced TEUs including discontiguous ones).

2. **Subsumption relations** *between pairs of parent phrase pairs and their subsumed children.*

3. **Mapping relations** *and implied reordering relations, between the children of parent nodes on the source and target side and their linked children on these source and target sides.*

over all possible maximal decompositions of that word alignment.

An important property of a representation that satisfies this notion of hierarchical translation equivalence is that it always permits complete reconstruction of the original word alignments for which the representation was induced.

Note that in practice we limit ourselves, as is common in the field, to *contiguous translation equivalence* when inducing TEUs and their subsumption structure from word-aligned sentence pairs. This means that discontiguous TEUs must always be embedded into larger, contiguous TEUs. But, is it consistent to allow only contiguous TEUs (phrase pairs) when performing the decomposition of word alignments, yet insist on making the mapping relations explicit, which may involve discontiguous mappings between parts of parts of (embedded) discontiguous TEUs? To answer this, we note that these are two quite different matters. The first one concerns a choice, namely the type of TEUs to allow in the decomposition of a word alignment while the second one concerns the consequence of what type of mapping relations this implies in order to get a complete description of Hierarchical Translation Equivalence. We chose to restrict to contiguous TEUs and require maximal decompositions of those. This implies that in decompositions contiguous TEUs are used whenever possible, but when contiguous TEUs embed discontiguous constructions, the discontiguous parts must still be added to complete them. And for such TEUs that embed discontiguous parts, discontiguous mapping relations are required to express how those parts came into being.

One might also wonder why to limit ourselves to contiguous TEUs as selected working units in the first place? Practical considerations play an important role in this, since the number of arbitrary discontiguous TEUs is *explosively exponential*,[13] and decoding with arbitrary discontiguous TEUs is equally intractable. But the choice for contiguous TEUs is not only pragmatic. In most languages, words that are close together form semantic units, which is one of the principles that explains that syntactic theory can work and constituency parsing gives useful structures. This doesn't means that discontiguous units do not exist or have merit, even possibly for monolingual parsing (van Cranenburgh and Bod, 2013). It just means that the *heuristic* bias of using *locality* as a selection criteria for choosing initial TEUs amongst the otherwise somewhat intractable exponential set is a sensible one. Finally we follow Chiang

[13]To see this consider a monotonic translation with n words on both sides. There are $(n * (n + 1))/2$ phrase pairs, namely all spans (i, j) such that $1 \geq i \leq j \leq n$. On the other hand, there are $2^n - 1$ discontiguous TEUs, basically any pair of linked words can be included or not, the only constraint is that at least one of them needs to be included.

(2005) by starting from contiguous TEUs (phrase pairs) but then relaxing the limitation of contiguity and generalizing those phrase pairs. The generalization is done by replacing one or two subsumed phrase pairs inside the original (contiguous) phrases by variables (so called "gaps"), thereby forming hierarchical rules. This is often considered a fair tradeoff between expressive power and complexity. It allows to stay within the framework of synchronous grammars, while still enabling most of the useful discontiguous constructions and greatly increasing generalizing power over simple phrase-based systems.

We have now given a description of hierarchical translation equivalence and explained why a faithful representation of all mapping relations is an important part of this concept. Earlier in section 4.3.1 we introduced the concept *mapping relation* in the context of PETs, for which these relations are always bijective and therefore simple to understand and represent. But we still need to explain how the synchronous mapping between source and target components taking part in the mapping relation is to be expressed formally for non-bijective translation equivalence relations. To this end we next introduce *set permutations*, as a formalism extending permutations that provides a compact and sufficient way to capture mapping relations for the full set of contiguous TEUs induced by any word alignment.

4.5 Set Permutations

Set permutations (s-permutations) are a simple and conservative extension to permutations, that allow to represent arbitrary many-to-many word alignments. Set permutations a lists of sets of target positions, one such list for every source position. For every source position i, the list element at position i gives the set of target positions to which that position is linked. Figure 4.9 shows an example of a word alignment for English–French with its associated set permutation.

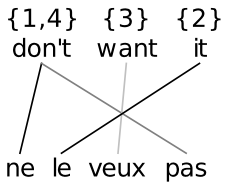


Figure 4.9: Example of English–French word alignment and corresponding set-permutation

It can be shown that set-permutations are equivalent to a canonical form representation of word alignments.[14] For a more detailed and formal description of

[14]It can be important to assure that a canonical form representation is used for word alignments,

set-permutations and their relation to HATs, we refer the reader to (Sima'an and Maillette de Buy Wenniger, 2013)

4.6 Hierarchical Alignment Trees (HATs)

An efficient and compact representation of hierarchical phrase pair translation equivalence is *Hierarchical Alignment Trees (*HATs*).* A HAT (Sima'an and Maillette de Buy Wenniger, 2013) is a hierarchical representation/decomposition of the phrase pairs in a word alignment; A HAT compactly represents a synchronous tree pair: a source and target tree pair with a node alignment relation between them. The recursive structure in a HAT shows the build up of phrase pairs from embedded phrase pairs. Hence, every node in a HAT represents the phrase pair at the fringe of the subtree under that node.

In Figure 4.12 we show what a visualization of a *Hierarchical Alignment Tree* for the example of Figure 4.10 looks like. There are actually *two* HATs displayed here. The upper tree shows the mapping from source to target while the bottom tree shows the mapping from target to source. Between the two trees we display the word alignments, making it directly clear how certain parts of the word alignment yield corresponding parts in the HATs. The filling and color/shade of the nodes represents the translation equivalence between the source and target side of phrases.

Like *normalized decomposition tree*s *(*NDTs*)* (Zhang et al., 2008a), HATs are *minimally branching decomposition* of word alignments into phrase pairs, i.e., every HAT node covers a phrase pair and it dominates the smallest number of translation units (the child nodes) that the phrase pair decomposes into. In Figure 4.12, the alignment underlying the phrase pair *our citizens unsern burgern* decomposes down minimally to two phrase pair nodes *our unsern* and *citizens burgern*. More intricate HATs may arise due to complex word alignments that involve many-to-many and discontiguous translation units.

Discontiguous translation units One important property of HATs is their explicit representation of discontiguous translation units. In Figure 4.12 the root node on the English/German side dominates, among others, two terminal nodes: this explicitly represents the discontiguous unit given by the alignment between *sind* + *schuldig* (positions 2 and 5) on the German side with a single English word *owe* (position 2). More generally, to differentiate between phrase pairs and separate parts of discontiguous translation units in the HAT representation, the latter are depicted as *terminal nodes* (nodes labeled with words without any children), whereas the former (phrase pairs) are represented as non-terminal nodes (circles with filling dominating a subtree).

which are essentially sets of link pairs. When represented as a list of link pairs, and not following a clear ordering convention, multiple alternative representations might be used for the same word alignments, which is undesirable.

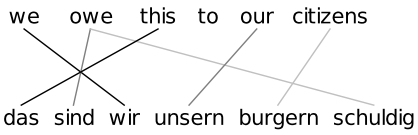


Figure 4.10: Example of alignment visualization for an aligned sentence pair (Europarl)

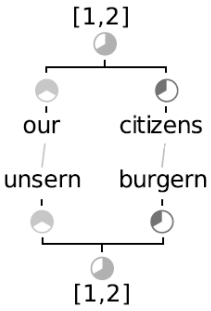


Figure 4.11: Example recursive composition of HATs

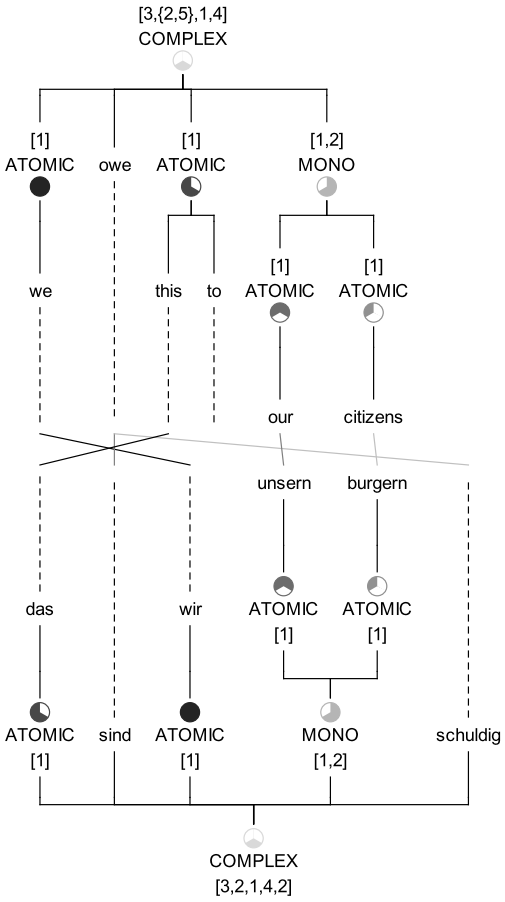


Figure 4.12: Visualization aspects: Filling (and shade) of the circles indicates the equivalence between the top source-to-target HAT and the mirrored target-to-source HAT displayed below it. Labels, such as [3,{2,5},1,4] at the top node, denote permutation-set reordering operations at nodes. The labels *ATOMIC*, *MONO* and *COMPLEX* in the HAT visualization indicate broad complexity categories for reordering.

Reordering operators Crucially, HATs extend NDTs by providing explicit representation of the reordering of the children under every node by a transduction operator, called a *set-permutation*, as well as the internal word alignments for atomic (non-decomposable) phrase pairs. Hence, a HAT is a decorated tree where the nodes are decorated with *set-permutations*. A set-permutation decorating a node in source side HAT is a list of integer sets denoting a transduction operation that applies to the children of that node to obtain the target side phrase reordering. We will exemplify HATs and set-permutations before we proceed further with discussing the properties of HATs.

Set-permutations such as [3,{2,5},1,4] (see the root node in Figure 4.12) denote reordering operations occurring under these nodes. In this example the source children 1, 2, 3, 4 map to target children (relative order) 3, {2,5}, 1, 4 respectively, where {2, 5} represents the fact that the second child is linked with two on the other side in positions 2 and 5. In the simpler monotone mapping *our citizens* | *unsern burgern* the set-permutation label is [1,2]. We also see the coarse reordering categories *ATOMIC*, *MONO* and *COMPLEX* in this figure, which are discussed next.

Complexity categories As mentioned above, every node is decorated with a set-permutation, specifying the relative mapping occurring directly below it. In the case of bijective mappings this describes a permutation. In the general case of arbitrary m-n mappings there are recurring target position in the mapping set of different source positions and/or multiple target positions occurring in the mapping set(s) of some source positions. Hence, the set-permutations can be grouped into coarser categories of mapping complexity. We distinguish the following five cases, ordered by increasing complexity:

1. *Atomic*: If the alignment does not allow the existence of smaller (child) phrase pairs: a subset of alignment positions that is not connected to the other positions while also forming a contiguous sequence on the source and target does not exists.

2. *Monotonic*: If the alignment can be split into two monotonically ordered parts.

3. *Inverted*: If the alignment can be split into two inverted parts.

4. *Permutation (Perm)*: If the alignment can be decomposed as a permutation of more than 2 parts.

5. *Complex (Comp)*: If the alignment cannot be decomposed as a permutation of parts, but the phrase does contain at least one smaller phrase pair.

Typically there are multiple HATs for a word alignment, corresponding to different possible minimally branching decompositions into phrase pairs. These alternative HATs can be efficiently computed and stored as a chart using a CYK-parser like chart

parsing algorithm that parses the alignment and builds a hypergraph of HATs in the process.[15]

A categorization of the complexity of the HAT as a whole is determined based on the complexity categories of the alignment mappings at its nodes. binary inversion-transduction trees (BITTs) is the least complex class consisting of only binary HATs that can be built for binarizable permutations (Huang et al., 2009), any HAT that contains only monotonic and/or inverted nodes belongs to this class. If a HAT contains at least one permutation node but no complex nodes it belongs to the category called PETs corresponding to general permutations (Zhang et al., 2008a; Satta and Peserico, 2005). Finally the occurrence of at least one complex node implies a HAT belongs to the set *HATs* which captures all possible many-to-many mappings.

4.6.1 The role of node operators

Node operators are crucial to the representative power of HATs and are essentially what distinguishes HATs from normalized decomposition trees. For this reason the semantics of node operators deserves some further explanation.

The semantics of a node operator that is a set-permutation (s-permutation) π over the child sequence of a node in a HAT is a generalization of the semantics of the operators in a permutation tree (PET), which are in turn a generalization of the inversion transduction tree (ITG) operators [] and $\langle\rangle$. The semantics of these operators is the same as the definition of s-permutations for standard word alignments, which are essentially a canonical form representation of word alignments, as mentioned before in section 4.5. The difference is that here the operators apply to the sequence of child nodes of respective nodes in HATs. As such they give a representation of the local reordering at those HAT nodes, which in the context of the entire HAT recursively combines with the operators of the other nodes back to the s-permutation for the original word alignment by which the HAT was induced. Thus the node operators in combination with the information about their hierarchical structure as provided by the HAT perfectly decompose the original word alignment, and are thus sufficient to reconstruct it entirely.

Algorithmically speaking the semantics of the s-permutation π over the children of the current node μ linked with a target side node μ_t is obtained as follows. Scanning π left to right: for the set of integers X in the i^{th} position in π generate child positions under μ_t corresponding to the integers in X and link each of these target positions with

[15]This is exactly what is done by our program. Note that in certain cases the number of HATs per alignment can become big, in particular for alignments that contain many monotone parts. One optimization we use in our algorithm is reasoning about null-aligned words outside the main algorithm. Computation is typically fast, provided enough memory is available. Rendering all HATs is done by enumerating all of them from the Hypergraph and writing their tree structures to a text file, then reading this unpacked forest from the text file by the tree visualization component. As this can become somewhat slow in case of many HATs, rendering all HATs can be turned on in the GUI, but only showing the first one is used as the default option.

the i^{th} child of μ.

Figure 4.13 exhibits three abstract word alignments that constitute minimal phrase pairs. In the HAT representation each of the three will be represented as a single linked node (pre-terminal level) dominating three terminal nodes. While the tree structure itself is the same, the node operators (s-permutations) on each are different. The node operators specify the internal alignment structure for, otherwise, hierarchically very difficult to represent word alignments. This way, the internal alignments of phrase pair units remain preserved.

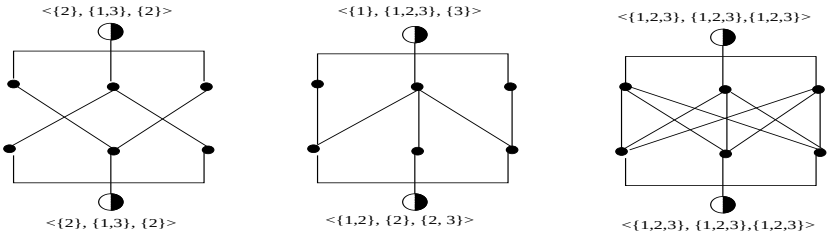


Figure 4.13: Three word alignments that constitute minimal phrase pairs and their HATs

Figure 4.12 exhibits an example word alignment (coming straight from our automatically aligned data) and a HAT on one side and its dual HAT on the other side, with links between the pairs of nodes (represented by a choice of circle filling). Note the discontinuous alignment of `owe` with `sind` and `schuldig` together with another crossing alignment. The HAT representation *reveals* a pair of linked nodes dominating the synchronous pair $\langle(X_1\ \texttt{owe}\ X_2\ X_3),\ (X_2\ \texttt{sind}\ X_1\ X_3\ \texttt{schuldig})\rangle$, where X_1, X_2 and X_3 stands for three aligned pairs of nodes in the HAT. The latter bilingual construction is a Chiang-style production. The pair of linked nodes is decorated with a s-permutation $\langle 3, \{2, 5\}, 1, 4\rangle$ on the English side and $\langle 3, 2, 1, 4, 2\rangle$ on the German side. Observe how these s-permutations actually link the second (`zijn`) and fifth (`schuldig`) children of the German node with the second child (`owe`) of the English linked node, thereby maintaining the lexical word alignment for such cases within the HAT structure.

4.7 Efficient Computation of HATs

In this section we describe what is necessary to efficiently implement the computation of HATs. This description will be based on our progressive insight and knowledge of the implementation described in (Zhang et al., 2008a). Our goal of this exposition is to explain concisely what is necessary for an efficient implementation, focusing on the most important parts of the implementation and building further on top of what is known from the implementation of the existing work of (Zhang et al., 2008a; Gildea et al., 2006; Zhang and Gildea, 2007) that we described before this. Our own implementation was initially developed independently, without knowledge of this line

of existing work, and as a result missed out on some of the efficiency tricks introduced by (Zhang et al., 2008a). Nevertheless, the central problem that we focus on, which is computing a chart representation of **all** HATs as opposed to a single canonical HAT requires a basic approach that does not change too much with or without knowledge of this earlier work. Basically, computing all HATs demands a CYK-style architecture that visits all source spans ordered from small to big, tests for each of these spans whether it constitutes a factorizable (reducible) sub-alignment, and if so computes all maximal decompositions with associated set-permutation labels and stores them in the corresponding chart entry of that span. What changes mainly with knowledge of (Zhang et al., 2008a) is that the efficiently computable reduction test function F can be exploited to test if a reduction is possible, a test which is more expensive without this smart trick yet not dominating the total algorithm complexity. The algorithm complexity is a function of the source length n. The total algorithm complexity is dominated by the need to visit all spans, of which there are $O(n^2)$ in combination with the need to compute all minimal reductions for each span. The latter is done most effectively by the well known VITERBI algorithm (Viterbi, 1969), also known as DIJKSTRA'S SHORTEST PATH algorithm (Dijkstra, 1959), which has $O(m^2)$ complexity, where m here is the length of the spans. The complexity of the complete HAT parsing algorithm is $O(n^4)$. The details of the reduction test implementation efficiency do not change this complexity.

4.7.1 CYK-style alignment parse algorithm

The main algorithm for the parsing of alignments and computation of a packed chart of HATs is called *constructHATs* (Algorithm 1). The algorithm creates the set of *induced* Hierarchical Alignment Trees for a triple of source sentence, target sentence and alignment, compactly represented as a packed forest. Our general approach exploits the fact that for general alignments as much as for permutations, it suffices to represent only one side (source or target) of the alignment directly, and build the packed forest by processing spans of that side. The information about the mapped to positions on the other side is represented where needed within the datastructures representing the spans for the side of the alignment chosen to work from.

Datastructures

The algorithm starts by creating a chart, which is a 2-dimensional array of chart entry (ChartEntry) instances. The chart entries corresponds to spans on the source side. In each of the chart entries, so called *inferences* are to be stored in a list. These inferences keep track of the alternative ways, to decompose/construct the reduction made at that chart entry. In the case of a chart entry corresponding to monotonic or inverted sub-alignments, the split point can be chosen at different positions, leading to different alternatives which are thus stored as different inferences in the inference list for the chart entry.

The chart used by the main algorithm is first initialized. Initialization consists of

first adding Inferences for the chart entries with span one, which cover just one source word. This is done by the method *addSpanOneInferences*. Finally in the main loop general inferences for increasingly long span lengths are added, for each chart entry by composition of partial HAT forests stored at smaller sub-entries for whom inferences have been added in previous iterations.

constructHATs
Data: triple< s, f, a >
Result:
chart - A packed chart containing the HATs that are consistent with the sentence pair and associated alignments

$n \leftarrow$ length(s) ;
chart ← **chartEntry***[n][n]* ;

`/* Initialize the leaf chart entries that cover just the source word and their associated target words */`
for $i \leftarrow 0$ **to** $n - 1$ **do**
  addSpanOneInferences(< s, f, a >,*i, chart*) ;
end

`/* Main loop. Find derivations for increasingly long span lengths, until the entire chart is filled. */`
for *spanLength* ← 2 **to** n **do**
  for *beginIndex* ← 0 **to** n − *spanlLength* + 1 **do**
    endIndex = *beginIndex* + *spanLength* − 1 ;
    `//Get the chart entry of length` *spanLength* `that starts from` *beginIndex*
    chartEntry = chart[beginIndex][endIndex] ;
    `//Find derivations for the chart entry`
    addInferencesForchartEntry(< s, f, a >,*chartEntry,chart*);
  end
end
return: chart;

Algorithm 1: An algorithm for constructing a packed chart containing all `implicated` HATs for a given triple< s, f, a >

Adding inferences for a chart entry consists of three main steps, which are summarized in the pseudocode of the function **addInferencesForChartEntry** (Algorithm 2):

1. Check that the source span corresponding to the chart entry can constitute valid reductions. This is done by the F function, function 4.2 we described before in

section 4.3.5. This corresponds to the line

if $F(chartEntry.getX(), chartEntry.getY()) == 0$ **then**

2. Find all minimal partitions of the source span, corresponding to the set of valid reductions for the source span. This corresponds to the next line

minComponentPartitions ← **findMinimalPartitions**(*entry*)

3. Compute the set of *Inferences* from the set of minimal partitions. This involves computing a set-permutation label for each of the partitions and combining it with the set of chart entries for the partition. These steps are done in the for loop, that starts after the minimal partition set computation.

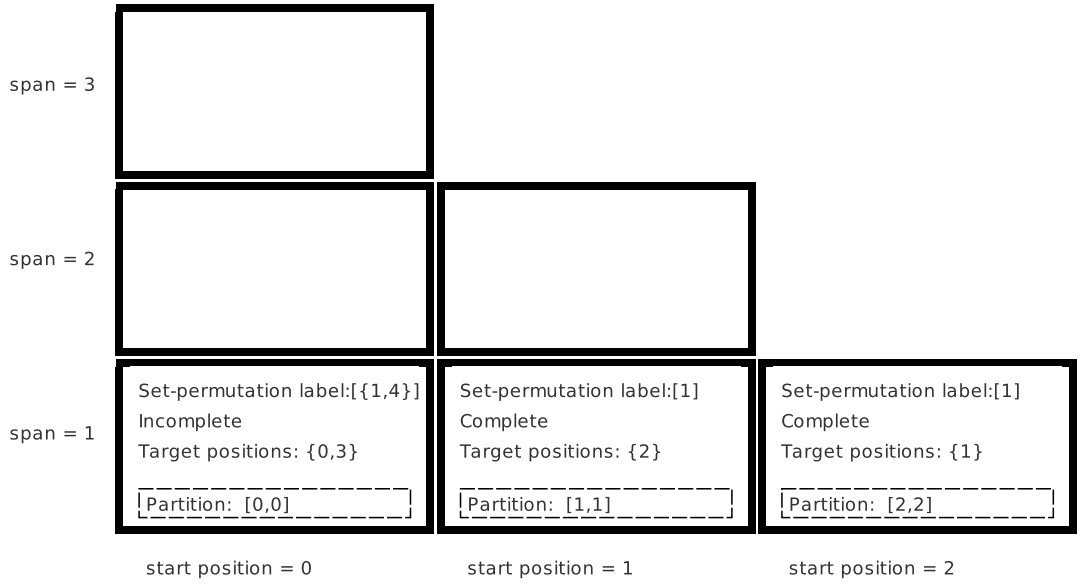
In the coming subsections, we will explain the second step *finding all minimal partitions* and third step *computing set-permutation labels* in more detail. But first we now illustrate the main algorithm using some examples.

4.7.2 Examples

Figures 4.14 and 4.15 illustrate how the HAT parsing algorithm (Algorithm 1) fills the chart. In this example, HATs are computed for the English–French sentence pair "⟨dont want it, ne le veux pas⟩" with discontiguous word alignment "0-0 0-3 1-2 2-1" . The algorithm and the chart only represents source and target positions, not actual words, but since words and positions map one to one we can think of them as being interchangeable. In Figure 4.14a, we show the chart just after initialization, in which inferences for source span 1, i.e. single source words, are added. Chart entries display: 1) the shared set-permutation label, 2) whether or not the added inferences are complete, 3) the set of target positions of the inferences in the chart entry, 4) alternative partitions, corresponding to alternative inferences (dashed boxes). Note that the inferences for the second and third source words are complete, but the inference for the first source word is incomplete. The reason is that the first source word maps to two discontiguous target positions: $\{0, 3\}$, and hence no complete contiguous TEU can be formed at this point.

Continuing after initialization, in Figure 4.14b in the main loop of the algorithm next inferences for source spans of length 2 are added. Notice that here an inference is only added for the span $[1, 2]$. For the span $[0, 1]$ no complete TEU can be formed, and so no inferences are added for it. Finally in Figure 4.15c, inferences are added for the source span $[0, 2]$ with length 3. Here the incomplete inference for the first source word is combined with the complete inference spanning the second and third source word, yielding an inference spanning the entire aligned sentence pair.

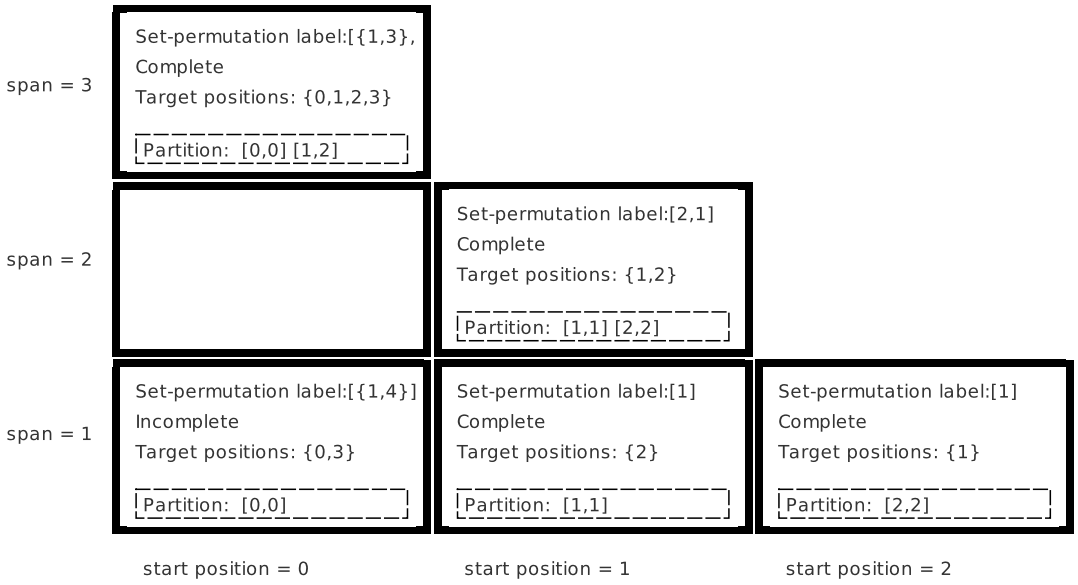
As one more example, Figure 4.16 shows the final chart for the HAT parsing algorithm for the sentence pair "⟨s1 s2 s3, t1 t2 t3⟩" with fully inverted word alignment



(a) After initialization: added inferences for span=1

(b) End of outer loop iteration 1: added inferences for span=2

Figure 4.14: Example of chart for HAT parsing algorithm run for the English–French sentence pair "⟨dont want it, ne le veux pas⟩" with discontiguous word alignment "0-0 0-3 1-2 2-1" . The vertical axis shows the length of the source spans, the horizontal axis shows the start position of the spans on the source side. The algorithm first adds inferences for span=1. Then the main loop of the algorithm incrementally adds inferences for increasingly longer spans.



(c) End of outer loop iteration 2: added inferences for span=3

Figure 4.15: Example of chart for HAT parsing algorithm run for the English–French sentence pair, continued from Figure 4.14.

"0-2 1-1 2-0". Here the important thing to notice is that there are two alternative inferences for the chart entry with source span $[0, 2]$ which covers the entire sentence pair. These two inferences correspond to alternative bracketings of this completely inverted word alignment.

4.7.3 An efficient Viterbi Algorithm to find the minimal Partitions

We now explain the algorithm to find the minimal partitions. This is a well known dynamic programming algorithm called the VITERBI algorithm, also known as DIJKSTRA's SHORTEST PATH algorithm. In our specific case, the algorithm incrementally computes the set of shortest paths (partitions) starting from the left boundary of a span, and incrementally increasing the right end until it reaches the right boundary of the span. While the algorithm is relatively simple, it does require some bookkeeping to enable the efficient (dynamic programing) computation it executes, and also to later retrieve the resulting shortest partition(s). Specifically, we need a datastructure *PartitionLatticeEntry* that keeps the minimal cost of a certain partition up to its first position on the left, as well as a list with back pointers to the Viterbi PartitionEntries that were used to arrive at this minimal cost. This is given below:

PartitionLatticeEntry - a datastructure containing :

- *minimalCost* : the cost of the best found partition of the left sub-range

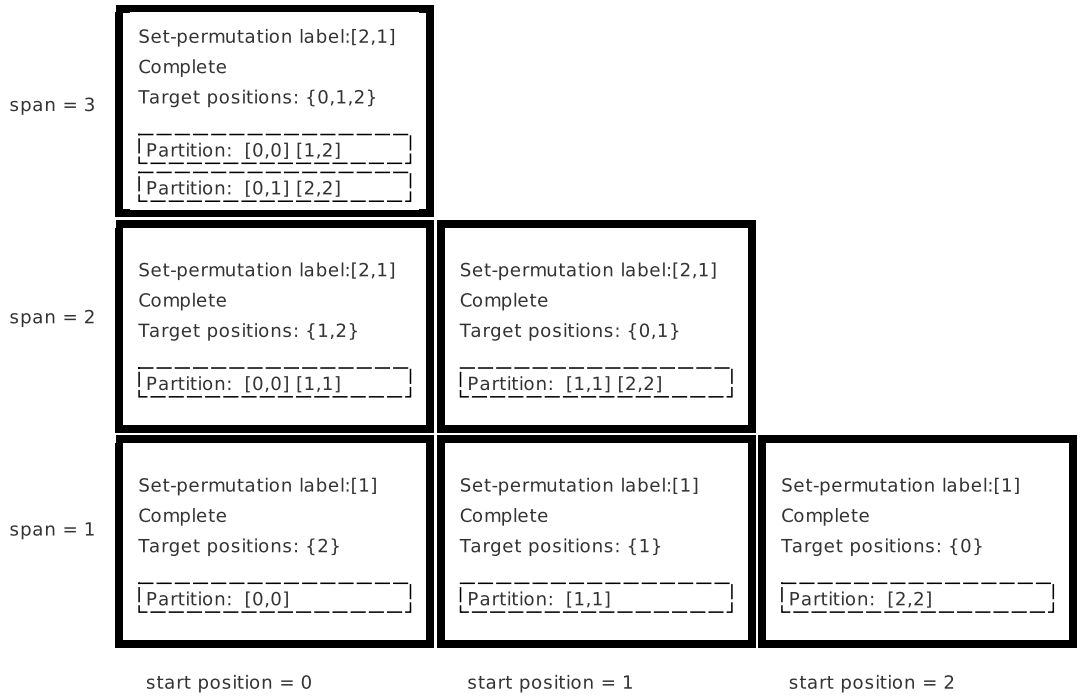


Figure 4.16: Example of chart for HAT parsing algorithm run for the sentence pair "⟨s1 s2 s3, t1 t2 t3⟩" with fully inverted word alignment "0-2 1-1 2-0" . Here, for brevity we show only the end result of the algorithm, after inferences have been added for all spans. Notice, that for the chart entry with span 2 there are multiple inferences. This reflects a well known phenomenon about building binary trees for fully monotone and fully inverted permutations: there are many alternative bracketings, leading to a number of derivations that is exponential in the length of such permutations.

- *viterbi* : a list of back pointers to previous partition entries that are in the partition of the left part with lowest cost
- *index* : the (relative) source position where this partition entry starts

A second datastructure *PartitionList* is used to store each actual (shortest) partition in its full unpacked form.[16] These unpacked partitions can then be used in the next steps to compute Inferences and their associated set-permutation labels.

PartitionList - a datastructure containing :

- *headEntry* : the partition entry that contains the head of the (possibly incomplete) list specifying the partition.

[16]Alternatively we could do this unpacking implicit and on the fly and, which would perhaps give a somewhat lower computational cost. We prefer explicit unpacking though since it gives the benefit of separation of responsibilities and loose coupling.

addInferencesForChartEntry
Data:
triple< s, f, a >
entry - the chart entry for which derivations are sought
chart - the chart

```
//If the chart entry has a F function value of 0, i.e. has
valid reductions
```
if $F(chartEntry.getX(), chartEntry.getY()) == 0$ **then**
  minComponentPartitions ← **findMinimalPartitions** (*entry*);
  for *minComponentPartition* ∈ *minComponentPartitions* **do**
    partitionChartEntries ← {};
    for *Pair* < i, j > ∈ *minComponentPartition* **do**
      partitionChartEntries.**add**(*chart[i][j]*);
    end
    setPermutationLabel ←
    findSetPermutationLabelForPartition(*minComponentPartition*);
    inference ← **createInference**(*partitionChartEntries,setPermutation*);
    entry.**addInference**(*inference*);
  end
end

Algorithm 2: Algorithm that finds inferences for a specific entry in the chart. The algorithm will find inferences that consist of the least possible parts, i.e. preferably BITTs but otherwise PETs or HATs with the least number of parts possible. In some cases the algorithm might not find any inference, which happens if the value of the F function for the source span of the chart entry has a value that is not zero.

- *partition* : a list of pairs of points, specifying the first and last indices of the subranges forming the partition.

Using this *PartitionLatticeEntry* datastructure and a n × n array of booleans *Inferable* that contains inferability information for the sub-entries of the entry for which we search the minimal partitions[17], we are now ready to specify the algorithm that finds the minimal partition lattice. This is shown as **generateMinimalPartitionLattice** (Algorithm 3).

The algorithm returns shortestPartitionLattice which contains both the cost of the minimal partitions, and back-pointers to the previous PartitionLatticeEntry entries that are part of these minimal partitions. The lattice shortestPartitionLattice stores shortest partitions for ranges that start at the left-most chart entry boundary, and grow on the left boundary, up to spanning the entire chat entry. It therefore has $n + 1$ elements, with

[17]Entries in *Inferable* are set to true, if they themselves correspond to phrase pairs, or if they correspond to single words in the source which may or may not constitute independent phrases.

```
generateMinimalPartitionLattice
Data: Inferable: a n × n array of booleans containing inferability information
Result: A Lattice containing the set of shortest partitions in compact form
shortestPartitionsLattice ← new PartitionLatticeEntry[n+1];
for i ← 0 to n do
    //Initialize all costs to the maximal cost possible
    shortestPartitionsLattice[i].minimalCost← n;
    shortestPartitionsLattice[i].index ← i;
end
shortestPartitionsLattice[0].minimalCost← 0
for i ← 0 to n do
    //Loop over Inferable entries starting from i
    for j ← i to n do
        if (Inferable[i][j]) then
            costNew ← shortestPartitionsLattice[i].minimalCost + 1;
            if (costNew < shortestPartitionsLattice[j+1].minimalCost) then
                //A better alternative has been found:  replace
                //the viterbi list and minimal cost
                shortestPartitionsLattice[j+1].minimalCost ← costNew ;
                shortestPartitionsLattice[j+1].viterbi ← { };
                shortestPartitionsLattice[j+1].viterbi.add(shortestPartitionsLattice[i]);

            else if (costNew == shortestPartitionsLattice[j+1].cost) then
                //An equally good alternative is found:
                //keep it as well
                shortestPartitionsLattice[j+1].viterbi.add(shortestPartitionsLattice[i]);

            end
        end
    end
end
return: shortestPartitionsLattice;
```

Algorithm 3: An algorithm for finding a lattice containing in compact form all partitions of an entry that contain the minimally possible number of sub-entries

its first element spanning nothing and having a cost of zero. Looking at Algorithm 3, we see how its main work is done within the innermost for loop, when the if statement "**if** ($Inferrable[i][j]$)" returns true. The latter implies that a phrase pair exists for the span $[i, j]$, and therefore a new partition can be made consisting of $shortestPartitionsLattice[i].minimalCost$ elements for the range $[0, i-1]$ and of one element for the span $[i, j]$ hence the "+1". This is compared with the current lowest cost for the span $[0, j]$, stored at $shortestPartitionsLattice[j+1].minimalCost$, were the "+1" in the index is explained by shortestPartitionLattice having $n+1$ elements, as

mentioned earlier. If the new cost is lower than the cost of the current shortest partition for the span $[0, j]$, the current element is replaced, and if the cost is equal the new partition is added as an alternative.

By traversing back using the back-pointers, a set of minimal partitions—defined by lists of split points or alternatively sub-ranges—can be easily computed. This is done by **findMinimalEntryPartitions** (Algorithm 4) which finds a list of partitions with a minimal number of involved entries. Every shortest partition itself is a list of pairs of integers, indicating the indices of the entries that make up the partition.

```
findMinimalEntryPartitions
Data: Inferable: a n × n array of booleans containing inferability information
Result: A set of shortest partitions

shortestPartitionsLattice ← generateMinimalPartitionLattice(Inferable);
incompletePartitionLists ← {};
shortestPartitions ← {};
incompletePartitionLists.add(PartitionList(shortestPartitionsLattice [n],{})) ;

while imcompletePartitionLists ≠ {} do
    newIncompletePartitionLists ← {};
    for list ∈ imcompletePartitionLists do
        if list.headEntry.index == 0 then
            shortestPartitions.add(list);
        else
            for parent ∈ list.headEntry.viterbi do
                newList ← clone(list);
                newPartition ← Pair(parent.index, list.headEntry.index − 1);
                newList.partitions.addFirst(newPartition);
                newList.setHeadEntry(parent);
                newIncompletePartitionLists.add(newList);
            end
        end
    end
    incompletePartitionLists = newIncompletePartitionLists;
end
return: shortestPartitions;
```

Algorithm 4: An algorithm for finding all partitions of an entry that contain the minimally possible number of sub-entries. The "real work" is done by **generateMinimalPartitionLattice** which generates a lattice of minimal partitions. This algorithm next unpacks this lattice and generates a list of lists of minimal partitions from it which is then finally returned.

4.7.4 Computing set-permutation labels for minimal partitions

Next we will describe how the set-permutation label is computed given a minimal partition, a chart entry and the chart itself as input. This computation is described below in the function **findSetPermutationLabelForPartition** (Algorithm 5) and involves two subroutines, namely **findShiftedTargetPositionsForChartEntry** (Algorithm 6) and **computeShiftedTargetPosTable** (Algorithm 7). The subroutine **computeShiftedTargetPosTable** computes a mapping table that maps from target positions to shifted target positions. This is done by first collecting all the target positions mapped to by components of the partition, in the case of components corresponding to proper reductions taking just the lowest mapped to target position for that component as a "representative". Next the resulting list of target positions is sorted from low to high. The resulting sorted list is finally used to produce a map, taking the sorted list elements as the map inputs and their corresponding list indices as the mapped to shifted target positions. This yields a map from target positions to shifted target positions. This map is next used by the subroutine **findShiftedTargetPositionsForChartEntry** (Algorithm 6) to produce a set of shifted target positions for each of the chart entries associated with the partition components. These lists of shifted target position are collected in a list, which in the end contains for every component of the partition a list of shifted target positions mapped to. Finally this list of sets of target positions now just needs to be mapped to a string by a standard *toString()* method for lists and that completes the set-permutation computation.

findSetPermutationLabelForPartition
Data:
minComponentPartition - the partition
entry - the chart entry for which derivations are sought
chart - the chart

listOfSets ← {};
shiftedTargetPosTable ← **computeShiftedTargetPosTable**(*entry*);
for *Pair* < *i*, *j* > ∈ *minComponentPartition* **do**
  shiftedTargetPosSet ←
    findShiftedTargetPositionsForChartEntry(*chart[i][j]*);
  listOfSets.add(*shiftedtTargetPosSet*);
end
`// The list of sets of shifted target positions is mapped`
`// to a simple string representation to get the final result`
return: listOfSets.toString();

Algorithm 5: Algorithm that computes the set-permutation label using the shifted target position table

findShiftedTargetPositionsForChartEntry
Data:
shiftedTargetPosTable - table for conversion to shifted target positions
entry - the chart entry for which the label is computed
chart - the chart

resultSet ← {};
if F(entry.getX(),entry.getY()) == 0 then
  lowTargetPos ← (l(*entry.getX(),entry.getY()*));
  resultSet.add(shiftedTargetPosTable.map*(lowTargetPos)*);
else
  for *targetPosition* ∈ **entry.getMappedTargetPositions() do**
    resultSet.add(shiftedTargetPosTable.map*(targetPosition)*);
  end
end
return: resultSet;

Algorithm 6: Algorithm that computes the set of shifted target positions

4.7.5 Important implementation details

In our description, for the purpose of brevity we have left out some details, we mention here the most important ones. These are:

- **HAT reconstruction**: How to store the *Inferences* in a properly indexed way within the chart entries, and also how the *Inference* information including the list of chart entries is used to reproduce the full HATs from the chart, starting from the top chart entry.

- **Unaligned words**: How unaligned words are dealt with.

HAT reconstruction is basically done by recursively looking up the inferences for the chart entry being processed, and then for each of these inferences taking the list of chart entries belonging to its associated partition and for each of those chart entries again recursively finding the list of *Inferences* and so on; in doing so recursively unpacking the forest of HATs compactly stored in the chart and using it to produce explicit representation of the HATs.[18]

Unaligned words: our method of choice of dealing with those is to "pre-normalize" the alignments before parsing them, taking out unaligned words, while keeping track where these unaligned words have been removed. This allows the whole HAT parsing to be done without the nuisance of the unaligned words, and the unaligned words are then added back to the thus produced HATs as a post-processing step.[19]

[18] In practice we actually produce the HATs one by one, rather than producing the exponentially large set top-down in one go. This requires a somewhat more complex enumeration algorithm that involving more state and more bookkeeping. Nevertheless the principle of recursively producing the HATs from the chart remains the same.

[19] Note that when adding unaligned words back, we still typically want to decide which words to

```
computeShiftedTargetPosTable
Data:
minComponentPartition - the partition
entry - the chart entry for which derivations are sought
chart - the chart

representingTargetPos ← {};
for Pair < i, j > ∈ minComponentPartition do
    if F(i, j) == 0 then
        representingTargetPos.add(l(i,j));
    else
        for targetPosition ∈ l(i,j) to u(i,j) do
            representingTargetPos.add(targetPosition);
        end
    end
end
//Sort the representing target positions
//to form an ascending list
sortedRepTargetPos ← representingTargetPos.sort();
shiftedTargetPositionLookupTable ← Map⟨integer,integer⟩();
for entryIndex ← 0 to sortedRepresentingTargetPositions − 1 do
    shiftedTargetPositionLookupTable.put(
      ⟨sortedRepTargetPos[entryIndex],entryIndex⟩);
end
return: shiftedTargetPositionLookupTable;
```

Algorithm 7: Algorithm that computes a shifted target position lookup table, a table telling for each of the relevant target positions what is its "shifted form" as required for computation of the set-permutation labels. The shifted indices are computed by first collecting all target positions, taking only the left-most position for chart entries that are themselves reducible (i.e. $F(i, j) == 0$). Then this list of collected target positions is sorted, and from the sorted list a map from target position to list index (shifted target position) is computed.

4.7.6 Summary

In the last section we described our implementation of HAT parsing. We saw how the requirements of computing and storing all HATs explicitly as a forest, and computing a *set-permutation* label for each of the HATs required significant algorithmic changes to the original existing algorithms for the computation of *normalized Decomposition*

group them with. As a simple heuristic, we allow only grouping with words on the left or right side of unaligned words, which still leaves exponentially many possibilities nonetheless. But since we take out the unknown words during parsing, and since hierarchical translation (typically) allows unaligned words only embedded inside (hierarchical) phrases, this does not make a difference from a practical viewpoint however.

Trees. In particular, computing and storing all HATs requires a CYK-style algorithm supported by the VITERBI algorithm to efficiently compute all maximal decompositions. Computing *set-permutation* labels mainly requires smartly shifting the mapping positions, by grouping together the position of contained reductions. The shifted positions can then be effectively computed using sorting. Finally we discussed some important implementation details with respect to HAT reconstruction and unaligned words.

4.8 Chapter Summary and outlook

This chapter started out with explaining the motivation for HATs, which finds its roots in the desire to coherently represent hierarchical translation equivalence. This can then be used in a way that yields (more) coherent hierarchical translations with a word order that better reflects reordering patterns in the training data and that better preserves the *meaning* of the original input. Following the introduction, we first gave an intuitive example-based description of HATs in section 4.2. Next in section 2.3.2 we described the notion of *translation equivalence*. We then gave a detailed summary of existing work on permutation trees (PETs) and normalized decomposition trees (NDTs) described in (Gildea et al., 2006; Zhang and Gildea, 2007) and (Zhang et al., 2008a), in section 4.3. Next, in section 4.4 we described our theoretical notion of *hierarchical translation equivalence* which besides enumeration of all TEUs, warrants an explicit representation of their *subsumption relations*. This motivates *set-permutation* labels, which compactly represent the local, recursive mapping relations of the TEUs (nodes) of a HAT; described in section 4.5. The theory of *hierarchical translation equivalence* is then combined with the notion of *set-permutation* labels to define HATs, as a proper extension of both PETs and NDTs. This is described in section 4.6. Finally, in section 4.7 we described our implementation of HAT parsing.

In the introduction we briefly discussed several applications of HATs. We now recap those applications belonging to the research done in the context of this thesis:

- Providing effective reordering labels that serve as soft reordering constraints which improve coherence of translations with respect to source-target word reordering and significantly improve state of the art hierarchical statistical machine translation.

- Exact measurement of alignment coverage and reordering complexity for different grammar formalisms, and also characterization of reordering complexity beyond specific grammars.

- Effective visualization and thereby facilitation of detailed study of empirical hierarchical translation equivalence. This application is described in (Maillette de Buy Wenniger and Sima'an, 2014b), and not worked out further in the context of this thesis.

In the next chapters, the first two applications will be described in more detail.

5

Bilingual Reordering Labels

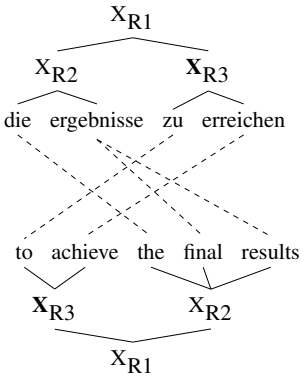
To my surprise, I found my first laboratory experiments absorbing, quite unlike the rather dry science I had been thought in college and medical school classrooms. In the laboratory, science is a means of formulating interesting questions about nature, discussing whether those questions are important and well formulated, and then designing a series of experiments to explore possible answers to a particular question.

– Eric R. Kandel, *In search of memory*

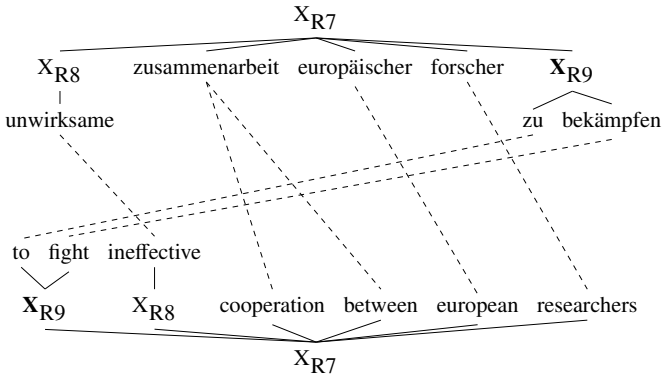
5.1 Introduction

Word order differences between languages constitute a major challenge in machine translation (MT). As early as the IBM models (Brown et al., 1988), the statistical MT (SMT) literature has produced a range of models aimed at predicting how the word order of the source sentence is transformed into a plausible target word order. Generally speaking, the existing reordering approaches that are integrated within translation (i.e., during decoding) can be grouped into the sequential (Tillmann, 2004; Galley and Manning, 2008) and the hierarchical (Chiang, 2005; Zollmann and Venugopal, 2006). While the sequential approach considers the reordering process as a finite-state process over word or phrase positions, the hierarchical approach (Hiero) works with a synchronous context-free grammar (SCFG). For a decade now, the hierarchical approach (Chiang, 2005) shows improved performance for language pairs with long-range reordering such as Chinese–English and Japanese-English (Chiang, 2005; Zollmann and Venugopal, 2006). The present work falls squarely within the hierarchical approach to reordering.

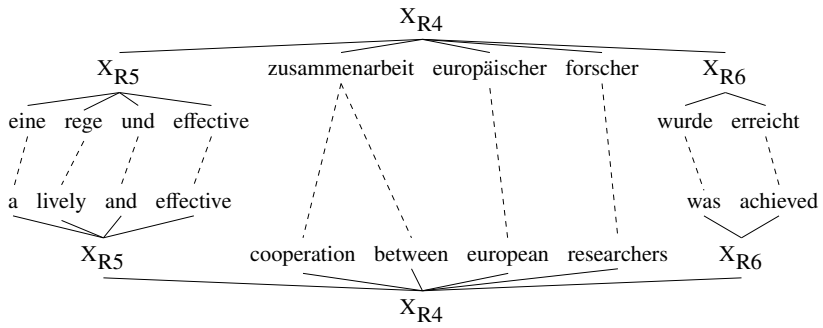
Hiero SCFG rules are extracted from a word-aligned parallel corpus. Like other phrase-based models (Och and Ney, 2004), the word alignment defines the set of translation rules that can be extracted from the parallel corpus. Hiero's rules are labeled with a single nonterminal label X, beside the start symbol of the SCFG. Hiero's



(a) (Left-binding) Inverted training example 1.



(b) (Left-binding) Inverted training example 2.



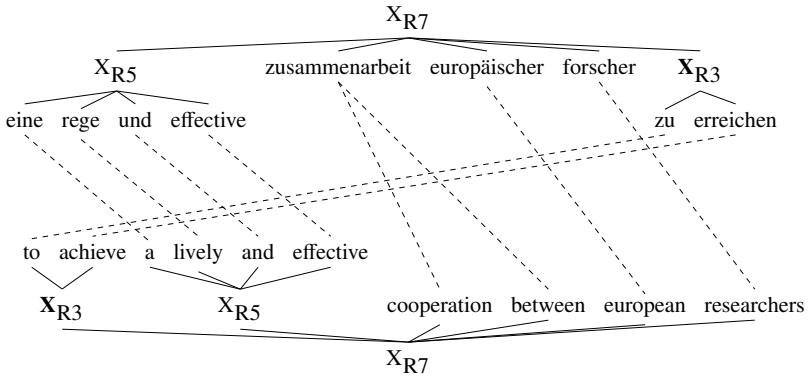
(c) Monotone training example.

Figure 5.1: Training examples, the labeled and indexed nodes represent (some of the) phrase pairs that can be extracted from the aligned sentence pairs.

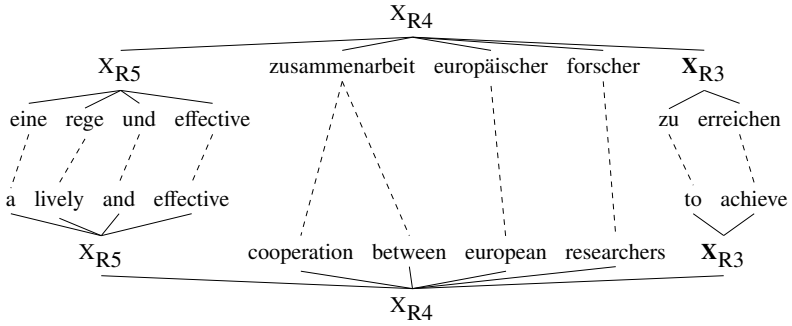
reordering patterns (straight/inverted) are embedded together with lexical context within synchronous rules, which makes local reordering within a rule sensitive to direct context. However, during decoding every rule may substitute on every nonterminal X, and thus it is independent of any other rule given the source string. This may result in suboptimal reordering as the following example shows.[1] Figure 5.1 shows a toy training parallel corpus of three word-aligned sentence pairs, decomposed into HIERO rules (hierarchical phrase pairs); the boxed *Ri* indices at the nodes stand for rule identities placed on the left-hand side of every rule. For example, in Figure 5.1c we find rule *R*5

$$X \rightarrow \langle \text{eine rege und effective}, \text{a lively and effective} \rangle$$

[1] This example is first introduced in chapter 1 in section 1.2, and is repeated here.



(a) Correct translation for new sentence that produces the right word order.



(b) Wrong alternative translation that can be produced by HIERO.

Figure 5.2: Translations of the new sentence "eine rege und effektive zusammenarbeit europäischer forscher zu erreichen".

and in Figure 5.1a, rule $R3$

$$X \rightarrow \langle \text{zu erreichen}, \text{to achieve} \rangle$$

By cutting out some of the embedded phrase pairs, we obtain HIERO rules with gaps. As an example, from the phrase pair at the root of the aligned sentence pair in Figure 5.1c, the hierarchical rule $R4$

$$\begin{aligned} X \rightarrow \langle \mathrm{X}_{\boxed{1}} \ & \text{zusammenarbeit europäischer forscher} \ X_{\boxed{2}}, \\ & X_{\boxed{1}} \ \text{cooperation between european researchers} \ X_{\boxed{2}} \rangle \end{aligned}$$

can be extracted by cutting out the two embedded phrase pairs $R5$ and $R6$ as gaps labeled X. Similarly, we obtain rule $R7$

$$\begin{aligned} X \rightarrow \langle \mathrm{X}_{\boxed{1}} \ & \text{zusammenarbeit europäischer forscher} \ X_{\boxed{2}}, \\ & X_{\boxed{2}} \ X_{\boxed{1}} \ \text{cooperation between european researchers} \ \rangle \end{aligned}$$

from the root phrase pair in Figure 5.1b. Note how the training examples for the English verbs “to fight" in Figure 5.1b and “was achieved" in Figure 5.1c are embedded within, respectively, monotone and inverted reordering patterns when translated into German.

We now exemplify how HIERO risks missing the correct word order and how labels from the surrounding word alignment context may help. In Figure 5.2a, translation rule $R7$ is combined with rule $R5$ and rule $R3$ to translate the new sentence “eine rege und effektive zusammenarbeit europäischer forscher zu erreichen”. Here starting from translation rule $R7$ and then substituting $R5$ and $R3$ on the two X nonterminals, the correct word order can be obtained. However, the rules extracted during training also permit a different translation of this sentence that produces the wrong word order, shown in Figure 5.2b. This translation is formed by combining $R4$ with $R5$ and $R3$. Both Hiero derivations are eligible, and the independence assumptions between rules suggest that there is no reason why Hiero’s synchronous grammar should be able to select the correct word order. The independence assumptions between the rules suggest also that the burden of selecting the correct reordering is left over to the target language model.

How could the use of word alignment context help produce preference for correct reordering in HIERO? Contrast the reordering structures in Figure 5.1a and Figure 5.1b to the structure in Figure 5.1c. In the first two the verb units, “to achieve" and “to fight" (labeled with a bold **X**), are inverted with respect to the embedding context, whereas in the latter example, the verb “was achieved" is monotone with respect to the embedding context. In this simple example, two types of verbs can be discriminated using word alignment types from the embedding rules, which can be used as HIERO labels. Such labeling can be obtained during HIERO rule extraction from the word-aligned training sentence pairs without need for other resources. By extracting such *reordering labels*, the incorrect substitution in Figure 5.2b could be either prevented or made far less likely than the correct alternative.[2] Phrases induce a certain reordering pattern with respect to their sub-phrases and with respect to the parent phrase that embeds them. We note that in a sentence-aligned, word-aligned parallel corpus, it turns out, there are many more reordering patterns than the binary choice of monotone/inverted.

The core idea in this work is to extract phrase labels from word alignments by first decomposing them recursively into their sub-component alignments. The decomposition we are interested in proceeds in the same way word alignments decompose recursively into phrase pairs (Zhang et al., 2008a). Such decomposition results in trees in which the nodes dominate phrase pairs. But the decomposition in this work maintains on every node also the alignment relation (called *node operator* or simply *operator*) which expresses how the sibling phrase pairs under that node

[2]One might wonder about the frequency of verbs that show such preferences for reordering: in the filtered test grammar (see experimental section) there are more than 27,000 phrase pairs, each with 2 words on both sides, that show such a preference for inversion relative to their embedding context. A large fraction of these phrase pairs corresponds to such verbal constructs. This itself is just a part of one of many types of reordering phenomena, selected for this example.

Sentence type	Sentence contents
Source Sentence	der handlungsspielraum der beiden betroffenen regierung ist also durch das internationale recht **begrenzt** .
Reference	any action by the two governments concerned is therefore **limited** by this international law .
Hiero	the margin for manoeuvre of two government is concerned by the international community limited .
Hiero-PCAL-SCD$_{B+S}$	the scope of the two governments concerned is therefore limited by international law .

Table 5.1: Source sentence, reference, baseline and best system output for sentence 543 from the German–English test set.

compose together at the target side relative to the source side.

Subsequently we bucket the resulting node operators into classes and use these classes as labels for Hiero rules. The ITG orientations (straight and inverted) turn out to be special cases of this general scheme, and in our experiments we show that limiting the choice to ITG, although beneficial, could be suboptimal sometimes.

Traditional grammar nonterminal labels signify hard categories, and substituting a rule with a left-hand side label X may take place only on the same nonterminal X. Like earlier labeling approaches, e.g., (Zhang et al., 2008a), we also find that exact match substitution for nonterminals is suboptimal. Following Chiang (2010), we devise a set of feature weights that allow any nonterminal label Y to substitute on any other label X with some cost determined by tuning the feature weights associated with the substitution. We call this approach *elastic-substitution decoding*, because during decoding the label substitution of Y on X with some cost can be seen as if the labels stretch during decoding to allow for as wide a set of translation hypotheses as needed.

We next show another example from real translation to further illustrate how reordering labels can help to overcome the limitations of Hiero. We are translating the German sentence *"der handlungsspielraum der beiden betroffenen regierung ist also durch das internationale recht* **begrenzt** *."* from Europarl. Table 5.1 shows this source sentence and the correct reference. To translate the German sentence correctly, the translation of the German verb *"***begrenzt***"* has to be moved in front of the translation of the phrase *"durch das internationale recht"*. Realizing that this word order is preferred requires looking beyond the language model's limited window. But without reordering labels, Hiero has no way to do so and relies only on it's language model. Consequently, it assigns higher weight to a different translation, in this case the one found by in Table 5.1, 3th row, which makes less rigorous changes to the word order. While piecewise locally such a translation might be reasonable, it globally gets the word order wrong, thereby also compromising the meaning of the sentence. In a sequence of suboptimal local choices, first the translation of *"begrenzt"* is put in the wrong place, then building on this mistake the order of *"concerned is"* is wrongly flipped to *"is concerned"* and finally the translation of *"internationale recht"* is wrongly translated into *"international community"* instead of *"international law"*. These faults all involve reordering errors explainable by choices that locally give better

translation and language model scores, but globally lead to erroneous reordering and suboptimal translations. This shows the danger of relying only on the language model to get a good word order. In contrast, using reordering labels in this example helps us to find the translation *"the scope of the two governments concerned is therefore limited by international law ."* (Table 5.1, 4th row) which while still not perfect, is clearly much better than the alternative proposed by HIERO.

After summarizing the HIERO model and discussing related work in some more detail, we propose a simple extension of normalized decomposition trees (NDTs) (Zhang et al., 2008a) with transduction operators that represent target-source phrase many-to-many mappings, including non-contiguous TEUs. Based on this extension, this chapter contributes:

- A novel labeling approach for Hiero, which exploits tree decompositions of word alignments, together with an effective proposal for features for *elastic-substitution decoding*,
- Extensive experiments on German–English and Chinese–English showing the superiority of this proposed labeling relative to HIERO and SAMT.
- Analysis of the experimental results showing the kind and source of improved performance.

5.2 Hierarchical models and closely related work

HIERO SCFGs (Chiang, 2005, 2007) are discussed in Chapter 3, in section 3.1. We do not repeat this discussion here, but instead refer the reader to this section.

In the next two subsections we will discuss work that is closely related to our work, followed by an overview of our contributions. Other, distantly related, work will be discussed in Section 5.8.

5.2.1 Lexicalized orientation models

We first look at work that distills reordering information from word alignments, sharing a general intuition with this work. Xiao et al. (2011) add a lexicalized orientation model to HIERO, akin to (Tillmann, 2004) and achieves significant gains. Nguyen and Vogel (2013a) extend upon this idea by integrating a phrase-based (non-hierarchical) lexicalized orientation model as well as a distance-based reordering model into HIERO. This involves adapting the decoder, so that rule chart items are extended to keep the first and last phrase pair for their lexical spans. Huck et al. (2013) overcome the technical limitations of both (Xiao et al., 2011) and (Nguyen and Vogel, 2013a) including a *hierarchical* lexicalized orientation model into HIERO. This requires making even more drastic changes to the decoder, such as delayed (re-)scoring at hypernodes up in the derivation of nodes lower in the chart whose orientations are affected by

them. Although sharing a similar intuition to our work, phrase-orientation models are not equivalent to HIERO/SCFG labeling mechanisms because formally they require extensions to SCFG. (which demand drastic changes in the decoder).

5.2.2 Soft constraints

Our approach towards soft constraints is based on (Chiang, 2010). Chiang's work uses labels similar to (Zollmann and Venugopal, 2006) with syntax on both sides. It applies Boolean features for rule-label and substituted-label combinations and uses discriminative training (MIRA) to learn which substitution combinations are associated with better translations. Their work also explores the usage of further rule extraction heuristics to extract a set of only non-crossing[3] rules, selected in order of relative linguistic robustness of the (partial) constituents for the left-hand-sides of the extracted rules. This yields a grammar that is even smaller than HIERO itself, while still giving similar results. In our case, without access to linguistic labels, this type of selection is not directly applicable and is therefore not used. Other approaches towards soft constraints will be discussed in the related work Section 5.8.

5.2.3 Innovations of the proposed method

This work is an extended version of an SSST 2014 workshop paper (Maillette de Buy Wenniger and Sima'an, 2014a) and differs substantially as follows. We provide a thorough motivation for our kind of labeling and explain Hierarchical Alignment Trees (absent in the SSST paper). We provide full detail of the label extraction approach (which was not discussed in detail in the short paper). Beside Hiero, we also report experiments for a new baseline, the syntactically-labeled SAMT (shortly discussed in subsection 5.6.1, page 144), both on German–English and Chinese–English. Comparing to a syntactically-labeled baseline gives a better feel for the performance differences to our approach. We discuss label-substitution features, our implementation of soft (label) matching constraints, in section 5.5 on page 140. Beside the basic label-substitution features found in SSST 2014, here we add a sparse label-substitution feature set, plus extensive additional experiments using this expanded feature set, which show how it further improves the results for German–English and Chinese–English translation. Finally, we provide qualitative analysis of the behavior of our model in terms of reordering and the role of the language model.

The labeling approach presented next differs from existing approaches. It is inspired by work on *elastic-substitution decoding* (Chiang, 2010) that relaxes the label matching constraints during decoding, but employs novel, non-linguistic bilingual labels. And it shares the bilingual intuition with phrase orientation models but it is

[3]Two extracted rules r_1 and r_2 *cross* when their associated source (and target) spans in the training data overlap, for example if r_1 spans source and target words 0–3, and r_2 spans words 3–4, these rules are crossing.

based on a new approach to SCFG labeling, thereby remaining within the confines of Hiero SCFG, avoiding the need to make changes inside the decoder.[4] Our approach is, to the best of our knowledge, the first to exploit labels extracted from decompositions of word alignments.

5.2.4 Some notes on terminology

A methodology of central importance in this paper is earlier the mentioned approach proposed by Chiang (2010) whereby the matching constraint is softened so that nonterminals can be substituted to other nonterminals with possibly different, mismatching labels. But besides softening the matching constraint, a second crucial component of this approach is the use of dedicated features, so called *label-substitution features*, that enable learning preferences over different types of label substitutions. Without addition of such features, labels would in fact be meaningless in a setting where strict matching of labels is not enforced. Unfortunately, no well established name exists in the literature for Chiang (2010)'s approach. In this paper we have chosen to use the term *elastic-substitution decoding* to refer to this approach. Some of the other names sometimes used in the literature for this approach are: *soft matching*, *fuzzy matching*, *soft labeling*, *soft matching constraints*. Finally, note that in earlier work (Maillette de Buy Wenniger and Sima'an, 2014a), we have in fact used multiple different terms for this concept. Here we have strived to improve this here, by using only a single term: *elastic-substitution decoding* which implies both: 1) the softening of the (label) matching constraint during decoding, 2) the usage of some set of label-substitution features.

5.3 Bilingual reordering labels by alignment decomposition

In the following we describe how reordering labels are formed. In particular, the rules we extract are identical to Hiero rules (Chiang, 2007) (see section 5.2, page 130) except for their labels. Following Zhang et al. (2008a), we view Hiero SCFG rule extraction from the hierarchical perspective of word alignment decomposition as a two step process. Initially, every word alignment in the training corpus is decomposed recursively into a canonical normalized decomposition tree (NDT). This results in a kind of training treebank of NDTs. Subsequently, the Hiero rules are extracted from these NDTs as in (Zhang et al., 2008a).

[4] Soft-constraint decoding can easily be implemented without adapting the decoder, through a smart application of "label bridging" unary rules. This is done by adding a set of unary rules, one rule for any combination of nonterminals; in combination with adding a marker to left-hand-side and right-hand-side nonterminals in order to avoid unary rule chains. In practice however, adapting the decoder turns out to be computationally more efficient, therefore we used this solution in our experiments.

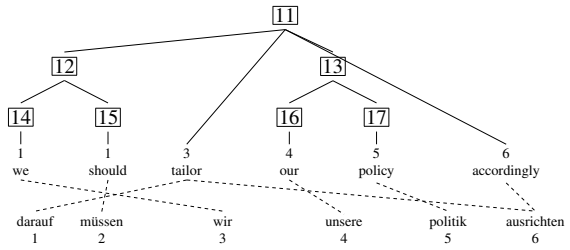


Figure 5.3: Example alignment from Europarl

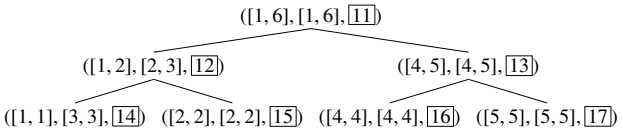


Figure 5.4: Normalized decomposition tree (Zhang et al., 2008a) extended with pointers to original alignment structure from Figure 5.3

It is useful here to exemplify the decomposition of word alignments into NDTs because it helps understand how we extend NDTs and the extracted rules with the bilingual reordering labels. Figure 5.3 shows an alignment from Europarl German–English (Koehn, 2005) along with a maximally decomposed phrase pair tree structure. Figure 5.4 shows the NDT for Figure 5.3, extended with pointers (boxed integers) to the original phrase pair tree in Figure 5.3. The boxed integers indicate how the phrase pairs in the two representations correspond. In an NDT, the fringe of every subtree is a phrase pair with spans indicated by the ranges of the two pairs of integers that decorate the root node of that subtree. Every composite phrase pair is recursively split up into a minimum number (two or greater) of contiguous parts. In Figure 5.4 the root node covers the source and target span from words [1,6], and it embeds two phrase pairs: the first covers the source-target spans ([1,2],[2,3]), and the second covers source-target spans ([4,5],[4,5]). From the source-target ranges that decorate the NDT nodes it is easy to compute bijective phrase permutation information: the two children of the root node in Figure 5.4 have ranges ([1,2],[2,3]) and ([4,5],[4,5]) respectively which shows that they are ordered in binary straight orientation. Note, however, that together these two phrase pairs in the example NDT do not explicitly show the build-up of their entire parent phrase pair ([1,6],[1,6]) because of a discontinuous translation equivalence involving `tailor ... accordingly/ darauf ... ausrichten`. The NDT does not explicitly show this discontinuity, nor does it show the internal word alignment within. In short, the NDT shows how phrase pairs maximally decompose

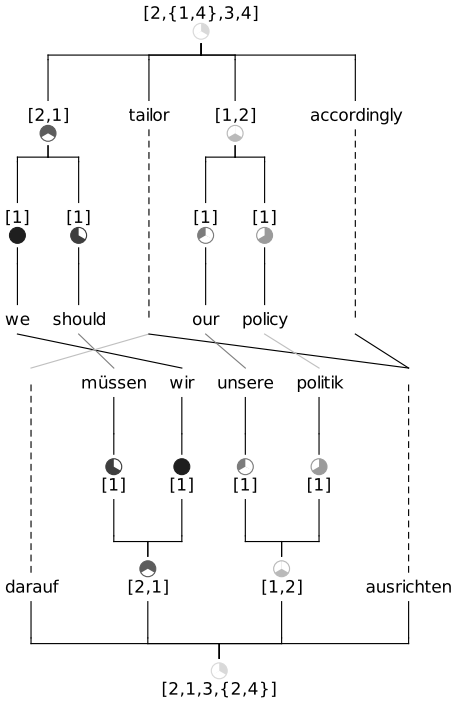


Figure 5.5: Hierarchical Alignment Tree corresponding to the example of Figure 5.3 and Figure 5.4. The trivial set-permutation label "[1]" of single-word pair phrase pairs is left out for brevity. Note that while in this case there is only one HAT for the alignment, in general a set of alternate HATs is induced; corresponding to alternate maximal decompositions for an alignment and encoded as a chart (*packed forest*).

into other phrase pairs and how these permute at each tree level, but NDTs abstract away from aspects of word alignments that are important for representing cases of discontiguous TEUs and other non-bijective alignments (many-to-many or unaligned words) internal to phrase pairs. This should not be an issue as long as phrase and rule extraction is the sole goal. However, for extracting labels capturing the types of reordering occurring inside or around phrase pairs, we propose that other alignment information is also needed at the NDT nodes. For this purpose we need Hierarchical Alignment Trees (HATs), which are decompositions of word alignments that retain all alignment information at the tree nodes. These representations were discussed earlier in this thesis, in chapter 4. We will next see how bucketing is used to define effective reordering labels based on HAT node operators.

5.3.1 Nonterminal labels by bucketing node operators

The node operators on HAT nodes encode decomposed word-alignment information. The HAT representation exposes the shared operators between different word alignments across a large corpus. In this work we propose to bucket these operators and employ the buckets as labels for the HIERO rules while extracting them. The bucketing is technically needed for various reasons. Firstly, it results in a manageable number of labels and avoids problems with sparsity. In a strict-matching version, for example, having more labels leads to more spurious derivations and splitting of the probability mass. Similarly, when working with elastic-substitution decoding, just one labeled version per Hiero rule type is used (see *canonical labeled rules*, end of section 5.5). But while necessary to keep the approach efficient and coherent, keeping just one labeled version does introduce uncertainty. Therefore the number of labels should be restricted, to avoid spreading out the probability mass over many different alternatives, making the selected rule versions and thereby the labels in general ultimately less reliable. Thirdly, the most common and well-known operators *monotone* ([0,1]) and inverted ([1,0]) have only one variant, while there are many variants of more complex operators for permutation and discontinuous reorderings. To avoid having the simpler but more frequent operators get obscured by a heap of complex but rare distinct operators, we bucket them to keep the total number of operators limited. Finally, in a soft-matching setting, reducing the number of labels helps to keep the number of features down (while also reducing complexity and problems with search errors) and is altogether important to keep the soft constraints learnable by the tuner.

In what follows we define two approaches for bucketing the HAT operators. The first approach simply uses the identity of the bucket of the operator on the current node itself (hence 0th order), whereas the second approach employs a bucketing of the operator on the parent of the current node (1st order).[5]

Phrase-centric (0^{th}-order) labels are based on the view of looking inside a phrase pair to see how it decomposes into sub-phrase pairs. The operator signifying how the sub-phrase pairs are reordered (target relative to source) is bucketed into a number of "permutation complexity" categories. As baseline labeling approach, we can start out by using the two well known cases of inversion transduction grammars (ITG) {*Monotone*, *Inverted*} and label everything[6] that falls outside these two categories with a default label "*X*" (leaving some HIERO nodes unlabeled). This leads to the following

[5] We think of our labels as implementing a Markov approach to SCFG labeling. The first (0th order) labeling approach just describes the reordering information at the phrase pairs itself, analogous to the way syntactic labels describe the syntactic category for the source and/or target side of phrase pairs in syntactic hierarchical SMT. The second (1st order) labeling approach describes the reordering relative to an embedding parent phrase, thereby looking not at the local reordering but at the reordering context of the parent.

[6] Non-decomposable phrase pairs (an example is the "Atomic" phrase pair in figure 5.7) will still be grouped together with Monotone phrase pairs (an example is the "Monotone" phrase pair in figure 5.7), since they are more similar to this category than to the catchall "*X*" category.

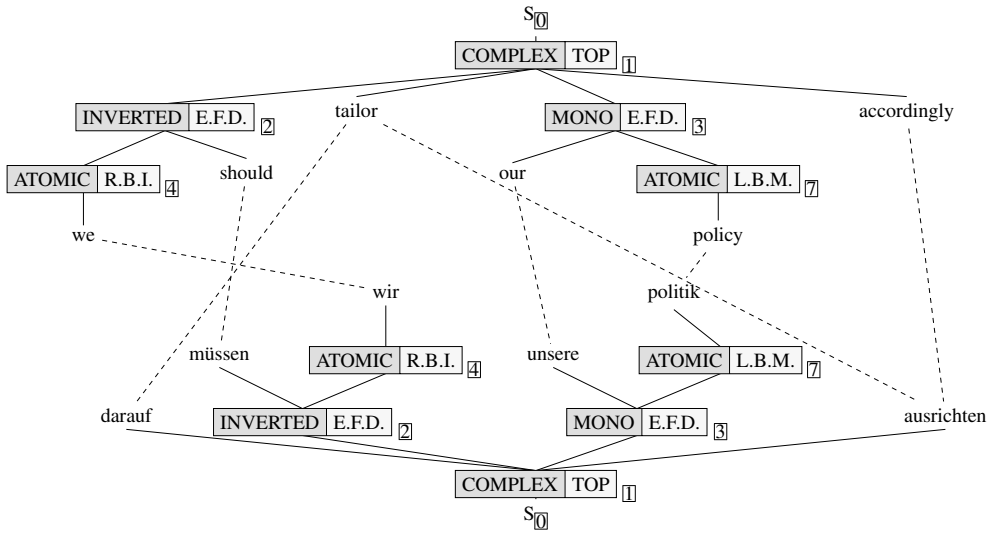


Figure 5.6: Synchronous trees (implicit derivations) based on differently labeled HIERO grammars. The figure shows alternative labeling for every node: *Phrase-Centric (*0^{th}*-order)* (gray) and *Parent-Relative (*1^{st}*-order)* (very light gray). The abbreviations for the Parent-Relative labels are: E.F.D.: embedded fully discontinuous; R.B.I.: right-binding inverted; L.B.M.: left-binding monotone.

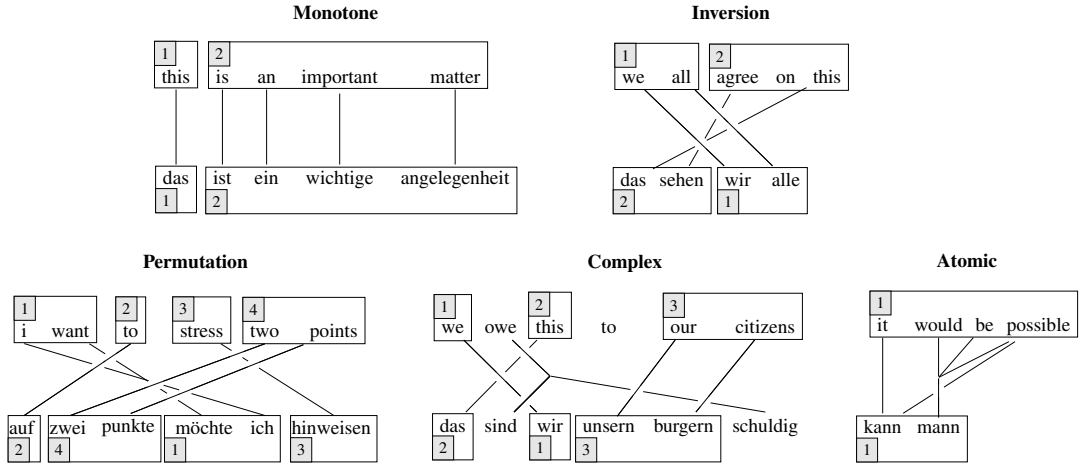


Figure 5.7: Different types of Phrase-Centric Alignment Labels

coarse phrase-centric labeling scheme, which we name $0^{th}{}_{ITG+}$:

1. *Monotonic (Mono)*: binarizable, fully monotone plus non-decomposable phrase pairs.
2. *Inverted (Inv)*: binarizable, fully inverted
3. *X*: decomposable phrase pairs that are not binarizable.

A clear limitation of the above ITG-like labeling approach is that all phrase pairs that decompose into complex non-binarizable reordering patterns are not further distinguished. Furthermore, non-decomposable phrase pairs are lumped together with

decomposable monotone phrase pairs, although they are in fact quite different. To overcome these problems we extend ITG in a way that further distinguishes the non-binarizable phrase pairs and also distinguishes non-decomposable phrase pairs from the rest. This gives a labeling scheme we will call simply 0^{th}-order labeling, abbreviated 0^{th}, consisting of a more fine-grained set of five cases, ordered by increasing complexity (see examples in Figure 5.7):

1. *Atomic*: non-decomposable phrases pairs.
2. *Monotonic (Mono)*: binarizable, fully monotone.
3. *Inverted (Inv)*: binarizable, fully inverted.
4. *Permutation (Perm)*: decomposes into a permutation of four or more sub-phrases.[7]
5. *Complex (Comp)*: does not decompose into a permutation and contains at least one embedded phrase pair.

In Figure 5.6, we show a phrase-complexity labeled derivation for the example of Figure 5.3. Observe how the phrase-centric labels reflect the relative reordering at the node. For example, the *Inverted* label of node-pair 2 corresponds to the inversion in the alignment of ⟨we should, müssen wir⟩; in contrast, node-pair 1 is complex and discontinuous and the label is *Complex*.

Parent-relative (1^{st}-order) labels capture the reordering that a phrase undergoes relative to an embedding parent phrase pair. This can be seen as a first-order view on reordering (if the phrase-centric type is considered a zero-order).

1. For a binarizable parent phrase pair with orientation $X_o \in \{Mono, Inv\}$, the source side of the phrase pair itself can either group to the left only *Left-Binding-*X_o, right only *Right-Binding-*X_o, or with both sides (*(Embedded) Fully-*X_o) of the source side of the embedding parent phrase pair.
2. *(Embedded) Fully-Discontinuous*: Any phrase pair within a non-binarizable permutation or complex alignment containing discontinuity.
3. *Top*: phrase pairs that span the entire aligned sentence pair.

In cases where multiple labels are applicable, the simplest applicable label is chosen according to the following preference order: {*Fully-Monotone*, *Left/Right-Binding-Monotone*, *Fully-Inverted*, *Left/Right-Binding-Inverted*, *Fully-Discontinuous*, *TOP*}.

In Figure 5.6 the parent-relative labels in the derivation reflect the reordering taking place at the phrase pairs with respect to their parent node. Node 4 has a parent node that inverts the order and the sibling node it binds is on the right on the source side, therefore it is labeled "right-binding inverted" (R.B.I.); E.F.D. and

[7] A permutation of length 3 can always be decomposed into a set if simpler nested permutations of length 2. As an example, the permutation [3,1,2] can be decomposed as the simpler nested permutation [2,[1,2]]. Equally, any SCFG of rank 3 can always be converted into a SCFG of rank 2, but not all SCFGs with rank ≥ 3 are binarizable.

L.B.M. are similar abbreviations for "(embedded) fully discontinuous" and "left-binding monotone" respectively. As yet another example node 7 in Figure 5.6 is labeled "left-binding monotone" (L.B.M.) since it is monotone, but the alignment allows it only to bind to the left at the parent node, as opposed to only to the right or to both sides which cases would have yielded "right-binding monotone" R.B.M. and "(embedded) fully monotone" (E.F.M.) parent-relative reordering labels respectively.

There is some similarity between the information gained in parent-relative reordering labels (by distinguishing left and right side binding directions) with the information gained in lexicalized orientation models that keep track of orientation in both left-to-right and right-to-left direction, i.e., (Galley and Manning, 2008; Huck et al., 2013). For the these models, determining the orientation in both directions slightly improves performance. Because in lexicalized orientation models keeping orientation in two directions helped, and since the binding direction for our monotone and inverted labels has similarity with it, we expected this binding direction to be also helpful for improving word order. Nevertheless, more fine-grained labels also increase sparsity and consequently make the learning problem more difficult. For this reason, the net effect of distinguishing binding direction remained hard to predict and could still have been be negative. We therefore also formed a set of *coarse* parent-relative labels ("$1^{st}{}_{Coarse}$") by collapsing the label pairs *Left/Right-Binding-Mono* and *Left/Right-Binding-Inverted* into single labels *One-Side-Binding-Mono* and *One-Side-Binding-Inv*[8]. This coarse variant was tested in all settings, but gave in general comparable or lower results than the original, more fine-grained version, and is therefore left out to increase readability of the reported result tables.[9]

5.4 Features : Relations over labels

In this section we describe the features we use in our experiments. These features are quite similar to the ones used by SAMT as described in the chapter 2, in section 3.3.1. Most of the symbols we use here are defined in that section, and we refer the reader to this section for basic terminology. But our features are not exactly the same as those used by SAMT, due to a different interpretation of the source/target we use which includes source/target side the left-hand-side. In what follows we use $src(r)$ to indicate the source side of the rule, including the source side of the left-hand-side label. Similarly $tgt(r)$ is the target side of the rule, including the target side of the left-hand-

[8] We could also further coarsen the 1^{st} labels by removing entirely all sub-distinctions of binding-type for the binarizable cases, but that would make the labeling essentially equal to the earlier mentioned $0^{th}{}_{ITG+}$ except for looking at the reordering occurring at the parent rather than inside the phrase pair itself. We did not explore this variant in this work, as the high similarity to the already explored $0^{th}{}_{ITG+}$ variant made it not seem to add much extra information.

[9] The coarse version does perform sometimes better in combination with sparse features. We attribute this to the fact that sparse features can lead to overfitting, but only to a lesser degree with a coarser (and therefore smaller) label set, since the number of sparse features is a polynomial function of the number of labels.

side label. Furthermore $un(src(r))$ is the source side without any nonterminal labels, and analogous for $un(tgt(r))$.

5.4.1 Basic Features

We use the following phrase probability features:

- $\hat{p}(tgt(r)|src(r))$: Phrase probability target side given source side
- $\hat{p}(src(r)|tgt(r))$: Phrase probability source side given target side

When decoding with strict matching, we reinforce those by adding phrase probability smoothing features. Smoothing is done by removing the labels on source and/or target side in all combinations. The additional phrase probability smoothing features for the labeled systems are:

- $\hat{p}(tgt(r)|un(src(r)))$
- $\hat{p}(un(src(r))|tgt(r))$
- $\hat{p}(un(tgt(r))|src(r))$
- $\hat{p}(src(r)|un(tgt(r)))$
- $\hat{p}(un(tgt(r))|un(src(r)))$
- $\hat{p}(un(src(r))|un(tgt(r)))$

We also add the following features:

- $\hat{p}_w(tgt(r)|src(r))$, $\hat{p}_w(src(r)|tgt(r))$: Lexical weights based on terminal symbols as for phrase-based and hierarchical phrase-based MT.
- $\hat{p}(r|lhs(r))$: Generative probability of a rule given its left-hand-side label

We use the following set of basic binary features, with 1 values by default, and a value $exp(1)$ if the corresponding condition holds:

- $\phi_{glue}(r)$: $exp(1)$ if rule is a glue rule
- $\phi_{lex}(r)$: $exp(1)$ if rule has only terminals on right-hand side
- $\phi_{abs}(r)$: $exp(1)$ if rule has only nonterminals on right-hand side
- $\phi_{st\_without\_tt}(r)$: $exp(1)$ if rule has terminals on the source side but not on the target side
- $\phi_{tt\_without\_st}(r)$: $exp(1)$ if rule has terminals on the target side but not on the source side
- $\phi_{mono}(r)$: $exp(1)$ if rule has no inverted pair of nonterminals

Furthermore we use the :

- $\phi_{pp}(r)$: Phrase penalty, $exp(1)$ for all rules.
- $exp(\phi_{wp}(r))$: Word penalty, exponent of the number of terminals on the target side
- $\phi_{rare}(r)$: $exp(\frac{1}{\#(r)})$: Rarity penalty, with $\#(r)$ being the count of rule r in the training corpus. This allows penalization of phrases using rarer rules.

5.5 Features for elastic-substitution decoding

Labels used in hierarchical SMT are typically adapted from external resources such as taggers and parsers. Like in our case, these labels are typically not fitted to the training data – with very few exceptions e.g., (Mylonakis and Sima'an, 2011; Mylonakis, 2012; Hanneman and Lavie, 2013). Unfortunately this means that the labels will either overfit or underfit, and when they are used as strict constraints on SCFG derivations they are likely to underperform. Experience with mismatch between syntactic labels and the data is abundant and using elastic-substitution decoding with suitable label-substitution features or a similar approach has been shown to be an effective solution (Venugopal et al., 2009; Marton et al., 2012; Chiang, 2010). The intuition behind elastic-substitution decoding is that even though heuristic labels are not perfectly tailored to the data, they do provide useful information provided the model is "allowed to learn" to use them only in as far as they can improve the final evaluation metric (usually BLEU). Next we introduce the set of label-substitution features used in our experiments.

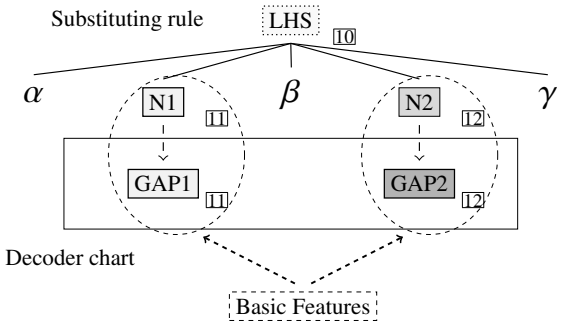
Basic label-substitution features consist of a unique feature for every pair of labels $\langle L_\alpha, L_\beta \rangle$ in the grammar, signifying a rule with left-hand-side label L_β substituting on a gap labeled L_α. These features are combined with two more coarse features, "*Match*" and "*Nomatch*", indicating if the substitution involves labels that match or not.

Figure 5.8 illustrates the concept of label-substitution features schematically. In this figure the substituting rule is substituted onto two gaps in the chart, which induces two label-substitution features indicated by the two ellipses. The situation is analogous for rules with just one gap. To make things concrete, let's assume that both the first nonterminal of the rule *N1* as well as the first gap it is substituted onto *GAP1* have label *MONO*. Furthermore let's assume the second nonterminal *N2* has label *COMPLEX* while the label of the gap *GAP2* it substitutes onto is *INV*.

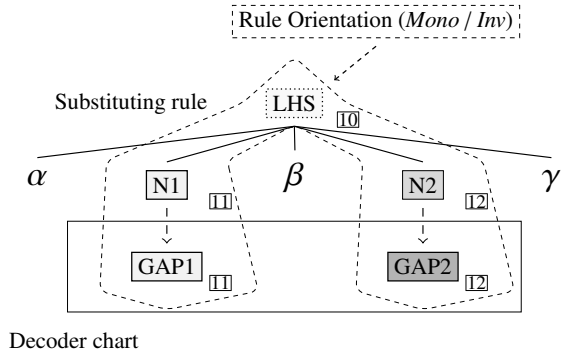
This situation results in the following two specific label-substitution features:

- subst(*MONO*,*MONO*)
- subst(*INV*,*COMPLEX*)

Sparse label-substitution features. Every applied rule, abstracted by its orientation plus reordering label signature, is enriched with information regarding the nature of the labeled gaps it is substituting onto. This information is encoded as sparse features defined as follows: For every non-empty ordered set of gaps denoted *gaps* and rule substituting on it *rule* with left-hand-side $LHS(rule)$ and nonterminals $N(rule)$, binary features are added for the specific combinations of four tuples: $\langle LHS(rule), N(rule), L(gaps), O(rule) \rangle$, where $O(rule)$ is reordering orientation (inverted/monotone) internal to *rule* and $L(gaps)$ the ordered set of labels belonging to *gaps* in the derivation. This is illustrated in Figure 5.8b by the dashed curve, which indicates these elements defining the sparse label-substitution feature for this rule



(a) Basic label-substitution features



(b) Sparse label-substitution feature

Figure 5.8: Label substitution features, schematic view. Labels/Gaps with same filling in the figures correspond to the situation of a nonterminal/gap whose labels correspond (for N1/GAP1). Fillings of different shades (as for N2/GAP2 on the right in the two figures) indicates the situation where the label of the nonterminal and the gap is different.

substitution. Assuming again the label assignment mentioned before: $N1 = MONO$, $L(GAP1) = MONO$, $N2 = COMPLEX$, $L(GAP1) = INV$ and furthermore assuming the

left-hand-side of the rule is *MONO* and the orientation of the rule is monotone, we would get the following sparse label-substitution feature:

- $\langle MONO, \{MONO, COMPLEX\}, \{MONO, INV\}, monotone \rangle$.

Canonical labeled rules. Typically when labeling Hiero rules there can be many different labeled variants of every original Hiero rule. With elastic-substitution decoding this leads to prohibitive computational cost. This also has the effect of making tuning the features more difficult. In practice, elastic-substitution decoding usually exploits a single labeled version per Hiero rule, which we call the "canonical labeled rule". Following Chiang (2010), this canonical form is the most frequent labeled variant.

System Name	Label Order	Label Granularity	Matching Type	Label Substitution Features Set
Hiero-0^{th}_{ITG+}	0^{th} order	Coarse	Strict	None
Hiero-0^{th}	0^{th} order	Fine	Strict	None
Hiero-1^{st}_{Coarse}	1^{th} order	Coarse	Strict	None
Hiero-1^{st}	1^{th} order	Fine	Strict	None
Hiero-0^{th}_{ITG+}-Sft$_B$	0^{th} order	Coarse	Soft	Basic
Hiero-0^{th}-Sft$_B$	0^{th} order	Fine	Soft	Basic
Hiero-1^{st}_{Coarse}-Sft$_B$	1^{th} order	Coarse	Soft	Basic
Hiero-1^{st}-Sft$_B$	1^{th} order	Fine	Soft	Basic
Hiero-0^{th}_{ITG+}-Sft$_{B+S}$	0^{th} order	Coarse	Soft	Basic + Sparse
Hiero-0^{th}-Sft$_{B+S}$	0^{th} order	Fine	Soft	Basic + Sparse
Hiero-1^{st}_{Coarse}-Sft$_{B+S}$	1^{th} order	Coarse	Soft	Basic + Sparse
Hiero-1^{st}-Sft$_{B+S}$	1^{th} order	Fine	Soft	Basic + Sparse

Table 5.2: An overview of our labeling schemes, their system names and the components they exploit. The suffixes in the system names in the first table column are abbreviations, directly corresponding to the system dimensions in the other columns: label order $\{0^{th}, 1^{th}\}$, label granularity ("*ITG+/Coarse*" indicating the coarse variant of $0^{th}/1^{th}$ order labels respectively), matching type (default is strict,"Sft" denotes elastic-substitution decoding), label substitution type ("B" denoting basic and "$B+S$" basic + sparse label substitution features).

5.6 Experiments

We evaluate our models on three language pairs: German–English in both directions and Chinese as source with English as target. The choice for these two pairs is driven by the knowledge that while in German word order is often tied with morphological changes at the word level, this is not the case for Chinese. Hence, we may expect different behavior on the two language pairs, which could provide insight into the limitations of our approach (reordering approaches in general) for dealing with languages where word order and morphology are tied together. Hence, we may expect different behavior on Chinese–English and the two language pairs involving German, which could provide insight into the limitations of our approach (reordering approaches in general) for dealing with languages where word order and morphology are tied together.

All data is lowercased as a last pre-processing step. In all experiments we use our own grammar extractor for the generation of all grammars, including the baseline Hiero grammars. This enables us to use the same features (as far as applicable given the grammar formalism) and assures that the grammars under comparison are identical in terms of using exactly the same set of extracted rules (differing only in labels and associated label features).

German–English and English–German The training data for our German–English and English–German experiments is extracted from parliament proceedings coming from the Europarl corpus (Koehn, 2005). We used WMT-07 data for development and testing. We used a maximum sentence length of 40 for filtering the training data. We use 995,909 sentence pairs for training, 2,000 for development and 2,000 for testing (single reference per source sentence). An overview of these and other statistics about

Data Set	#sentence pairs	#source words	Mean/Std source sentence length	#target words	Mean/Std target sentence length
	German–English				
training	994,861	20,358,970	20.46 ± 8.88	21,449,488	21.56 ± 9.14
development	2,000	55,526	27.76 ± 15.93	59,425	29.71 ± 16.99
testing	2,000	55,580	27.79 ± 15.67	59,153	29.58 ± 16.85
	English–German				
training	994,861	21,449,488	21.56 ± 9.14	20,358,970	20.46 ± 8.88
development	2,000	59,425	29.71 ± 16.99	55,526	27.76 ± 15.93
testing	2,000	59,153	29.58 ± 16.85	55,580	27.79 ± 15.67
	Chinese–English				
training	7,340,000	147,609,854	20.11 ± 9.25	159,567,205	21.74 ± 9.91
development	1,812	46,864	25.86 ± 13.02	54,665.5	30.17 ± 15.89
testing	1,912	48,250	25.24 ± 12.8	58,844.0	30.78 ± 16.69

Table 5.3: Size statistics for the training, development and testing datasets used in the experiments. Note that the German–English datasets are the same as the English–German datasets, but used in opposite direction. This is reflected by the number of sentences and other statistics for these corpora. For the Chinese–English dataset, there are 4 references for the target side of the development set and testing set. Hence, for these datasets the number of target words (#target words) is a mean taken over the four references, and the mean/std target sentence length is computed from the four references combined into one.

Language Pair	#sentences	#words	Mean/Std sentence length
German–English	994,861	21,449,488	21.56 ± 9.14
English–German	994,861	20,358,970	20.46 ± 8.88
Chinese–English	5,427,696	135,635,561	24.99 ± 15.99

Table 5.4: Language model training corpora sizes.

the training, development and testing dataset is shown Table 5.3. Both source and target of all datasets are tokenized using the Moses (Koehn et al., 2007) tokenization script. We do not use compound splitting as part of the data preparation.[10] We use GIZA++ (Och and Ney, 2003) for word alignment[11], with the *grow-diag-final-and* scheme (see Appendix A.1) for symmetrization of alignments in two directions. For these experiments both the baseline and our method use a 4-gram language model with Kneser-Ney smoothing trained on the target side of the full original training set (995,909 sentences). Statistics about the data used to train the language models is shown in Table 5.4.

Chinese–English The training data for our Chinese–English experiments is formed by combining the full sentence-aligned *MultiUN* (Eisele and Chen, 2010; Tiedemann,

[10] Although compound-splitting could be important for building the best possible system, this was not the goal in our experiments. Our goal was to create an experimental setup that allows for a fair, replicable comparison of our systems against Hiero and SAMT. As we believe that the potential disadvantage of omitting compound-splitting should affect all compared systems equally, given our goal, we judged that for the sake of simplicity it was reasonable to do so.

[11] For this language pair, we used the following model iterations for GIZA++ alignment: model 1: 4, HMM model: 3, model 3: 3, model 4: 3.

2012)[12] parallel corpus with the full sentence-aligned *Hong Kong Parallel Text*[13] parallel corpus from the Linguistic Data Consortium[14]. The *Hong Kong Parallel Text* data is in *traditional Chinese* and is thus first converted to *simplified Chinese* to be compatible with the rest of the data[15]. We used a maximum sentence length of 40 for filtering the training data. The combined dataset has 7,340,000 sentence pairs. The *MulitUN* dataset contains translated documents from the United Nations, similar in genre to the parliament domain. The *Hong Kong Parallel Text* in contrast contains a richer mix of domains, namely Hansards, Laws and News. For the development and test sets we use the *Multiple-Translation Chinese* datasets from LDC, parts 1–4[16], which contain sentences from the News domain. We combined part 2 and 3 to form the development set (1813 sentence pairs) and part 1 and 4 to form the test set (1912 sentence pairs). For both development and testing we use 4 references. The Chinese source side of all datasets is segmented using the Stanford Segmenter(Chang et al., 2008)[17]. The English target side of all datasets is tokenized using the Moses tokenization script. We again use GIZA++ (Och and Ney, 2003) for word alignment[18], with the *grow-diag-final-and* scheme for symmetrization of alignments in two directions.

For these experiments both the baseline and our method use a 4-gram language model with Kneser-Ney smoothing trained on 5,427,696 sentences of *domain specific*[19] news data taken from the "Xinhua" sub-corpus of the English Gigaword corpus of LDC.[20]

5.6.1 Experimental Structure

We compare our reordering-labeled systems against two baseline systems: the (unlabeled) HIERO and the target-language syntax-labeled variant known as SAMT. In our experiments we explore the influence of three dimensions of bilingual reordering labels on translation accuracy. These dimensions are:

- *label order* : the type/order of the labeling $\{0^{th}, 1^{st}\}$

[12]Freely available and downloaded from http://opus.lingfil.uu.se/

[13]The *Hong Kong Parallel Text* corpus contains a significant amount of duplicate sentence pairs. We removed these duplicates and kept only one copy per unique sentence pair.

[14]The LDC catalog number of this dataset is LDC2004T08

[15]Using a simple conversion script downloaded from http://www.mandarintools.com/zhcode.html

[16]LDC catalog numbers: LDC2002T01, DC2003T17, LDC2004T07 and LDC2004T07

[17]Downloaded from
http://nlp.stanford.edu/software/segmenter.shtml

[18]For this language pair, we used the following model iterations for GIZA++ alignment: model 1: 5, HMM model: 5, model 3: 3, model 4: 3.

[19]For Chinese–English translation the different domain of the train data (mainly parliament) and development/test data (news) requires usage of a domain specific language model to get optimal results. For German–English, all data is from the parliament domain, so a language model trained on the (translation model) training data is already domain-specific.

[20]The LDC catalog number of this dataset is LDC2003T05

- *label granularity* : granularity of the labeling {Coarse,Fine}
- *matching type* : the type of label matching performed during decoding {Strict,Soft}
- *label substitution feature set* : the type of label substitution features that is used during decoding, if any.

An overview of the naming of our reordering labeled systems is given in Table 5.2.

SAMT We use the original label extraction scheme, as described in (Zollmann and Venugopal, 2006). In particular we allow the "\", "/" and "+" operators with **two** arguments. To keep the grammar size manageable, we do not allow "double plus" (A+B+C) type labels, and we do not allow non-lexicalized rules. The choice not to allow non-lexicalized rules was made to keep SAMT (like our systems) comparable to Hiero, apart from the labels. This avoids giving SAMT additional reordering capacity (through abstract rules) which Hiero lacks, and thereby also keeps decoding times more workable.[21] Finally, we use SAMT, as in the original work, with strict matching.[22]

Training and decoding details Our experiments use Joshua (Ganitkevitch et al., 2012) with Viterbi best derivation. Baseline experiments use normal decoding, whereas elastic-substitution decoding experiments relax the label matching constraints while adding label-substitution features to facilitate learning of label substitution preferences. For training we use standard Hiero grammar extraction constraints (Chiang, 2007) (phrase pairs with source spans up to 10 words; abstract rules are forbidden). During decoding a maximum span of 10 words on the source side is maintained. With Hiero and the labeled systems in soft-matching setups we use the Hiero phrase probabilities in both directions (Chiang, 2007), making the labeled systems weakly equivalent to Hiero apart from their label-substitution features. For the labeled systems in the strict matching systems, we follow (Zollmann and Venugopal, 2006) in using the phrase probabilities that use the labels as well as all smoothed versions of these phrase probabilities, as discussed in section 5.4.[23]

[21] For example (Li et al., 2012b) shows that such abstract rules can by themselves provide performance gains on top of improvements from the labels used on normal Hiero rules.

[22] We spent major effort at implementing elastic-substitution decoding for SAMT as in (Chiang, 2010) but faced huge scalability issues due to the number of labels which gives problems for the implementation of dot items in Joshua.

[23] We use only the Hiero phrase probability features for the labeled systems in the soft-matching setting, to keep them as close as possible to Hiero, so that the effect of the label-substitution features can be measured purely. But for the systems in the strict matching setting we use the phrase probability features and phrase probability smoothing features that involve the labels. In this setting we involve the labels to allow them to influence the translation decisions through the phrase probabilities. But using the labels in the phrase probabilities, the smoothed variants are necessary to avoid sparsity problems, particularly with the sparse SAMT labels (Zollmann, 2011).

We train our systems using (batch k-best) MIRA (Cherry and Foster, 2012) as borrowed by Joshua from the Moses codebase, allowing up to 30 tuning iterations. Following standard practice, we tune on BLEU, and after tuning we use the configuration with the highest scores on the development set with actual (corpus level) BLEU evaluation. We report lowercase BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), BEER (Stanojević and Sima'an, 2014a) and TER (Snover et al., 2006) scores for the test set. We also report average translation length as a percentage of the reference length for all systems. This is useful for analysis. In our experiments we repeated each experiment three times to counter unreliable conclusions due to optimizer variance. The scores are averages over three runs of tuning plus testing. We use Multeval version 0.5.1.[24] for computing these metrics. We also use MultEval's implementation of statistical significance testing between systems, which is based on multiple optimizer runs and approximate randomization. Differences that are statistically significant with respect to the Hiero baseline and correspond to improvement/worsening are marked with $\triangle H/\triangledown H$ at the $p \leq .05$ level and $\blacktriangle H/\blacktriangledown H$ at the $p \leq .01$ level. For average translation length either higher or lower may be better, depending on whether the baseline length was too low or too high. We therefore use $\square H/\blacksquare H$ in case of length to mark significant *change* with respect to the baseline at the $p \leq .05$ / $p \leq .01$ level. Apart from computing the statistical significance of differences with respect to the Hiero baseline, we also computed statistical significance of differences with respect to the SAMT baseline. The significance for these differences are analogously marked $\triangle S/\triangledown S/\square S$ at the $p \leq .05$ level and $\blacktriangle S/\blacktriangledown S/\blacksquare S$ at the $p \leq .01$ level. We also report the Kendall Reordering Score (KRS), which is the reordering-only variant of the LR-score (Birch et al., 2010; Birch and Osborne, 2010) (without the optional interpolation with BLEU) and which is a sentence-level score. For the computation of statistical significance of this metric we use our own implementation of the *sign test* [25] (Dixon and Mood, 1946), as also described in (Koehn, 2010). For every experiment we use boldface to accentuate the highest score across systems for all metrics except TER and length. Since TER is an error metric, lower values are better, and we therefore instead mark the lowest value with boldface for it. For length, neither higher or lower is necessarily better. Arguably, lengths closer to the reference length are better, but to keep things simple and not obscure the meaning of boldface as indicating "best" for the performance metrics, in case of length we don't boldface a "best" value.

[24] https://github.com/jhclark/multeval

[25] To make optimal usage of the 3 runs we computed equally weighted improvement/worsening counts for all possible 3×3 baseline output / system output pairs and use those weighted counts in the sign test. While traditionally the procedure of dealing with ties in the sign test is discarding them, there is in fact no real consensus with respect to their correct treatment. However, recent literature explains that it may sometimes be better to equally divide the ties between two systems (Rayner and Best, 1999); intuitively a more "conservative" approach which we adopted in our experiments.

[26] Statistical significance is dependent on the variance of resampled scores, and hence sometimes different for same mean scores across different systems.

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length
	German–English					
Hiero	28.39	32.94	19.01	58.01 ▼*S*	67.44	100.60 ■*S*
SAMT	28.32	32.88	18.81	57.70 ▲*H*	67.63	100.07 ■*H*
Hiero-$0^{th}{}_{ITG+}$-Sft_{B+S}	28.48 △*H* △*S*	32.93	18.97	57.69 ▲*H*	**67.37**	100.08 ■*H*
Hiero-0^{th}-Sft_{B+S}	**28.57** ▲*H* ▲*S*	32.92	18.99	**57.65** ▲*H*	67.41	100.16 ■*H*
Hiero-$1^{st}{}_{Coarse}$-Sft_{B+S}	28.43	33.00 △*H* ▲*S*	19.06	57.77 ▲*H*	67.46	100.46 □*H* ■*S*
Hiero-1^{st}-Sft_{B+S}	28.47	**33.03** ▲*H* ▲*S*	**19.07**	57.77 ▲*H*	67.45	100.59 ■*S*
	English–German					
Hiero	20.89	40.62	18.79	**64.85**	65.65 ▼*S*	98.90 ■*S*
SAMT	20.87	40.73	18.97	64.98	**65.97** ▲*H*	99.64 ■*H*
Hiero-$0^{th}{}_{ITG+}$-Sft_{B+S}	20.95	**40.79** ▲*H* ▲*S*	**19.24**	65.06 ▼*H* ▼*S*	65.91	99.73 ■*H* ■*S*
Hiero-0^{th}-Sft_{B+S}	20.97	40.67	18.97	65.17 ▼*H* ▼*S*	65.86 △*H*	99.66 ■*H* ■*S*
Hiero-$1^{st}{}_{Coarse}$-Sft_{B+S}	**21.00**	40.74 △*H* △*S*	18.94	65.07 ▼*H* ▼*S*	65.89	99.71 ■*H* ■*S*
Hiero-1^{st}-Sft_{B+S}	20.90	40.63	19.00	65.03 ▽*H* ▽*S*	65.76 ▽*S*	99.63 ■*H* ■*S*
	Chinese–English					
Hiero	31.63 ▽*S*	30.56	13.15	59.28 ▲*S*	58.03 ▼*S*	97.15 ■*S*
SAMT	31.87 △*H*	30.61	13.38	59.97 ▼*H*	59.94 ▲*H*	98.46 ■*H*
Hiero-$0^{th}{}_{ITG+}$-Sft_{B+S}	32.02 ▲*H*	30.66 ▲*H*	13.20	**59.12** △*H* ▲*S*	58.66 ▲*H*▼*S*	97.41 ■*H* ■*S*
Hiero-0^{th}-Sft_{B+S}	32.43 ▲*H* ▲*S*	30.96 ▲*H* ▲*S*	13.54	60.33 ▼*H* ▼*S*	**60.17** ▲*H*△*S*	99.35 ■*H* ■*S*
Hiero-$1^{st}{}_{Coarse}$-Sft_{B+S}	32.68 ▲*H* ▲*S*	**31.03** ▲*H* ▲*S*	**13.73** ▲*H*	59.92 ▼*H*	59.88 ▲*H*	99.05 ■*H*
Hiero-1^{st}-Sft_{B+S}	**32.69** ▲*H* ▲*S*	31.01 ▲*H* ▲*S*	13.65 △*H*	60.02 ▼*H*	59.99 ▲*H*	99.10 ■*H* ■*S*

Table 5.5: Primary mean results bilingual labels with elastic-substitution decoding and basic plus sparse label-substitution features.[26] In this and the following result tables, statistically significant improvement/worsening/change with respect to the HIERO baseline is marked with $\triangle H/\triangledown H/\square H/$ at the $p \leq .05$ level and $\blacktriangle H/\blacktriangledown H/\blacksquare H$ at the $p \leq .01$ level. Analogously, statistically significant improvement/worsening/change with respect to the SAMT baseline is marked with $\triangle S/\triangledown S/\square S/$ and $\blacktriangle S/\blacktriangledown S/\blacksquare S$.

Label Type	German–English		English–German		Chinese–English	
	Absolute Size	Relative Size	Absolute Size	Relative Size	Absolute Size	Relative Size
Hiero	17.2	1	25.9	1	33.4	1
SAMT	74.9	4.35	94.0	3.63	154.7	4.63
Hiero-$0^{th}{}_{ITG+}$	19.1	1.11	28.3	1.09	38.4	1.15
Hiero-0^{th}	28.7	1.67	41.6	1.60	55.7	1.67
Hiero-1^{st}	23.6	1.37	34.3	1.32	48.9	1.46

Table 5.6: Filtered test grammar sizes for different label types and different language pairs. Absolute sizes are in millions of rules. Relative sizes are with respect to the HIERO (baseline) grammar, or equivalently with respect to the grammars used in the elastic-substitution decoding experiments, which are equal in size to Hiero. Grammars are taken from the strict matching systems for the label types.[27]

5.6.2 Primary results: Soft bilingual constraints and basic+sparse label-substitution features

Table 5.5 shows the primary results of our full labeling scheme which uses elastic-substitution decoding both with basic and sparse label-substitution features. *Hiero* is the HIERO baseline, beneath it are shown the systems that use elastic-substitution decoding (Sft): Hiero-$0^{th}{}_{ITG+}$-Sft and Hiero-0^{th}-Sft using 0^{th}-order labels. Hiero-1^{st}-Sft corresponds to the system with 1^{st}-order, parent-relative labels.

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length
	German–English					
Hiero	28.39	32.94	**19.01**	58.01 ▼*S*	67.44	100.60 ■*S*
SAMT	28.32	32.88	18.81	**57.70** ▲*H*	**67.63**	100.07 ■*H*
Hiero-$0^{th}{}_{ITG+}$-Sft$_B$	28.45 △*H* △*S*	32.94	18.96	57.75 ▲*H*	67.32	100.03 ■*H*
Hiero-0^{th}-Sft$_B$	28.45	32.98 ▲*S*	18.98	57.73 ▲*H*	67.51	100.21 ■*H*
Hiero-$1^{st}{}_{Coarse}$-Sft$_B$	**28.48** △*S*	32.98 ▲*S*	18.99	57.79 ▲*H*	67.36	100.25 ■*H*
Hiero-1^{st}-Sft$_B$	28.45	**33.00** △*H* ▲*S*	**19.01**	57.79 ▲*H*	67.45	100.52 ■*S*
	English–German					
Hiero	20.89	40.62	18.79	**64.85**	65.65 ▼*S*	98.90 ■*S*
SAMT	20.87	40.73	18.97	64.98	65.97 ▲*H*	99.64 ■*H*
Hiero-$0^{th}{}_{ITG+}$-Sft$_B$	21.00	**40.79** ▲*H* ▲*S*	**19.07**	65.00 ▽*H* ▽*S*	65.93 △*H*	99.58 ■*H* ■*S*
Hiero-0^{th}-Sft$_B$	21.01	40.73 △*H* △*S*	18.84	65.04 ▼*H* ▼*S*	65.70 ▽*S*	99.34 ■*H* ■*S*
Hiero-$1^{st}{}_{Coarse}$-Sft$_B$	20.99	40.76 △*H* △*S*	**19.07**	65.03 ▽*H* ▽*S*	**66.03** △*H*	99.67 ■*H* ■*S*
Hiero-1^{st}-Sft$_B$	**21.05** ▲*H* ▲*S*	**40.79** ▲*H* ▲*S*	19.02	64.94	66.00 △*H*	99.61 ■*H* ■*S*
	Chinese–English					
Hiero	31.63 ▽*S*	30.56	13.15	59.28 ▲*S*	58.03 ▼*S*	97.15 ■*S*
SAMT	31.87 △*H*	30.61	13.38	59.97 ▼*H*	**59.94** ▲*H*	98.46 ■*H*
Hiero-$0^{th}{}_{ITG+}$-Sft$_B$	31.93 ▲*H*	30.37 ▼*H* ▼*S*	12.84 ▼*S*	58.86 ▲*H* ▲*S*	57.60 ▽*H*▼*S*	96.48 ■*H* ■*S*
Hiero-0^{th}-Sft$_B$	32.20 ▲*H* ▲*S*	30.74 ▲*H* ▲*S*	13.27	59.45 ▽*H* ▲*S*	58.92 ▲*H*▼*S*	97.89 ■*H* ■*S*
Hiero-$1^{st}{}_{Coarse}$-Sft$_B$	32.55 ▲*H* ▲*S*	30.86 ▲*H* ▲*S*	13.41	59.57 ▼*H* ▲*S*	59.03 ▲*H*▼*S*	98.21 ■*H* ■*S*
Hiero-1^{st}-Sft$_B$	**32.61** ▲*H* ▲*S*	**30.98** ▲*H* ▲*S*	**13.58** △*H*	**60.19** ▼*H* ▽*S*	59.84 ▲*H*	99.03 ■*H* ■*S*

Table 5.7: Mean results bilingual labels with elastic-substitution decoding only basic label-substitution features.[18]

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length
	German–English					
Hiero	28.39	32.94	19.01	58.01 ▼*S*	67.44	100.60 ■*S*
SAMT	28.32	32.88	18.81	**57.70** ▲*H*	**67.63**	100.07 ■*H*
Hiero-$0^{th}{}_{ITG+}$	28.36	32.90 ▽*H*	18.86	57.83 ▲*H*	67.30	100.27 ■*H* □*S*
Hiero-0^{th}	**28.39**	**33.03** ▲*H* ▲*S*	**19.07**	57.75 ▲*H*	67.55	100.28 ■*H* □*S*
Hiero-$1^{st}{}_{Coarse}$	28.22 ▽*H*	32.90	18.93	57.93 ▽*S*	67.47	100.37 ■*H* ■*S*
Hiero-1^{st}	28.27	32.80 ▼*H* ▽*S*	18.78	57.95 ▼*S*	67.39	100.20 ■*H*
	English–German					
Hiero	20.89	40.62	18.79	**64.85**	65.65 ▼*S*	98.90 ■*S*
SAMT	20.87	40.73	18.97	64.98	65.97 ▲*H*	99.64 ■*H*
Hiero-$0^{th}{}_{ITG+}$	**20.90**	**40.79** ▲*H* ▲*S*	**19.25**	65.08 ▼*H* ▼*S*	65.95 △*H*	99.74 ■*H* ■*S*
Hiero-0^{th}	20.87	40.70	19.14	65.04 ▽*H* ▽*S*	66.05 ▲*H*	99.67 ■*H* ■*S*
Hiero-$0^{th}{}_{Coarse}$	20.82	40.62	18.80	65.08 ▼*H* ▼*S*	66.05 ▲*H*	99.45 ■*H* ■*S*
Hiero-1^{st}	20.76	40.58	18.73	65.16 ▼*H* ▼*S*	**66.06** ▲*H*	99.50 ■*H* ■*S*
	Chinese–English					
Hiero	31.63 ▽*S*	30.56	13.15	**59.28** ▲*S*	58.03 ▼*S*	97.15 ■*S*
SAMT	31.87 △*H*	30.61	13.38	59.97 ▼*H*	**59.94** ▲*H*	98.46 ■*H*
Hiero-$0^{th}{}_{ITG+}$	**31.94** ▲*H*	**30.84** ▲*H* ▲*S*	13.37	60.76 ▼*H* ▼*S*	59.45 ▲*H*	99.13 ■*H* ■*S*
Hiero-0^{th}	31.90 ▲*H*	30.79 ▲*H* ▲*S*	**13.45**	60.11 ▼*H*	59.68 ▲*H*	98.65 ■*H*
Hiero-$1^{st}{}_{Coarse}$	31.57 ▽*S*	30.57	13.09	59.58 ▼*H* ▲*S*	59.13 ▲*H*▼*S*	97.59 ■*H* ■*S*
Hiero-1^{st}	31.77	30.62	13.20	60.13 ▼*H*	59.89 ▲*H*	98.47 ■*H*

Table 5.8: Mean results bilingual labels with strict matching.[18]

German–English: Hiero-0^{th}-Sft$_{B+S}$ (with BLEU score of 28.57) slightly outperforms both HIERO and SAMT baselines by almost 0.2 BLEU points, which is statistically significant. We remind the reader that German–English is rather difficult because word order in German is tied with morphological variations at the word level,

for which our model, like other models of reordering, does not have a proper solution. This goes to highlight the limitations of these kind of models in general.

English–German: We see a similar picture. As we shall see next in the next ablation study, for English–German there is no clear gain from using the sparse features.

Chinese–English: Hiero-1^{st}-Sft_{B+S} has the highest score for BLEU for all tested systems on Chinese–English translation, outperforming Hiero and SAMT by approximately 0.8–1.1 BLEU points. The coarsely labeled variant of this system Hiero-$1^{st}{}_{Coarse}$-Sft_{B+S} yields similar improvements, with slightly better scores for METEOR, BEER and TER and slightly worse scores for BLEU and KRS. For metric TER, all labeled systems, including SAMT, suffer performance loss in comparison to Hiero. We found out that the length ratio for the output of the Hiero-1^{st}-Sft_{B+S} system to the reference is 0.99 whereas the ratio for Hiero's output is 0.97, i.e., it seems that TER is penalizing more heavily longer output even if it is closer in length to the reference. This turns out largely due to the fact that the 4-gram LM tuned weight for the labeled systems is always far lower than for Hiero, suggesting that the 4-gram LM has a smaller contribution during tuning for BLEU. Tuning for BLEU is not guaranteed to give improved performance on all metrics, but we do see here improved performance for three out of four metrics.

In Table 5.6 we show the absolute and relative sizes of the grammars for the different label types. The reported sizes are for grammars that are filtered for the test set and that are taken from the systems that use the labels in a strict matching setting. Note that for the systems that use the bilingual reordering labels as soft bilingual constraints, the grammar size is always equal to that of Hiero. The reason for this is that, as we mentioned earlier, with elastic-substitution decoding we use only one canonical labeled rule per Hiero rule. Looking now at the grammar sizes in the table, we see that the size of the grammar for SAMT is on average more than a factor 4 bigger than the one used by Hiero and the reordering labeled systems in the soft-matching setting. At the same time, the improvement over both SAMT and Hiero by the reordering labeled systems is considerable, especially for Chinese–English. Even in the strict matching setting, the reordering labeled systems have still grammar sizes that are much smaller than SAMT, at most 1.67 times the size of the baseline Hiero grammar. And in what follows we will see that also for these systems the reordering labeled systems are performing as good as SAMT and Hiero or better.

Next we will do ablation experiments where we isolate the effects of using sparse features on top of the basic ones, and after that the using elastic-substitution decoding vs. the traditional mere strict label matching./

[27]This means that, in contrast to the soft-matching system that allows only one canonically labeled rule per Hiero (unlabeled) rule type, there can be multiple alternative labeled rule versions per Hiero rule type.

5.6.3 Experiments with elastic-substitution decoding with basic label-substitution features only

Now we isolate out the sparse features and use only the basic label-substitution features with elastic-substitution decoding. The results are shown in Table 5.7.

German–English: There are only minor improvements for BLEU and METEOR over the HIERO and SAMT baselines, with somewhat bigger improvements for TER. But SAMT has the highest improvements for TER and KRS over HIERO on this language pair.

English–German: there are already bigger improvements with the Hiero-1^{st}-Sft_B system achieving the highest improvements for BLEU (+0.16) and METEOR (+0.17). A considerable, though not the highest improvement of KRS scored is also achieved by this system (+0.35). Importantly, this system also scores better than SAMT on these three metrics, which only has superior performance for TER. Notice that this system, not the system with basic+sparse label-substitution features performs best for English–German.[28]

Chinese–English: the improvements are considerable, +0.98 BLEU improvement over the HIERO baseline for Hiero-1^{st}-Sft as well as +0.42 METEOR and +1.81 KRS. TER is worsening with +0.85 for this system. For Chinese–English the *Fine* version of the labels gives overall superior results for both 0^{th}-order and 1^{st}-order labels.

Compared with the primary system (basic+sparse label-substitution features) results in Table 5.5 we see that the added nuances of sparse label-substitution features can make a difference, strongly so for German–English and to a lesser degree also for Chinese–English.

5.6.4 Experiments with strict label matching: No added softeners

Now we explore the added value of soft label matching features by excluding them out and using the labels as traditional grammar labels (hard constraints). In contrast to the elastic-substitution decoding experiments were only canonical labeled rules are included in the grammar, in this setting all labeled rule variants are used. The motivation for this difference is that in a strict label matching setting coverage loss problems arise during translation. Using all labeled rule variants, as common in strict labeling approaches (e.g. (Zollmann and Venugopal, 2006)), does not solve these problems but at least reduces them as much as possible in this setting.

The results are shown in Table 5.8. For the computation of SAMT for Chinese–English we initially had problems with grammar extraction due to the enormous size of even the filtered grammar. We finally overcame this by extracting the grammar in parts and merging them. To be exact, this does make some of the feature values which involve normalization potentially slightly different from what they would have been

[28] Part of the reason why the latter performs worse may be overfitting, we get back to this point in the analysis section.

if they were directly computed for the full grammar in one go. However, due to the inherently highly heuristic nature of these features, this is assumed to not have a real effect on the actual results. All systems in the table use the default decoding with strict label matching.

German–English: The effect of strict bilingual labels is mostly positive: although we have no improvement for BLEU we do achieve significant improvements for METEOR and TER on the test set. Also the syntax-labeled system SAMT does not improve over HIERO in terms of BLEU while it does improve in terms of KRS and TER.

English–German: the syntax-labeled SAMT system is comparable to HIERO. We also see insignificant improvement on the test set only for METEOR, but we see a statistically significant worsening in TER. Interestingly, Kendall Reordering Score (KRS) is an exception as it improves with 0.40 (significant at the $p \leq 0.01$ level).

Chinese–English: overall Hiero-$0^{th}{}_{ITG+}$ shows the biggest improvements, namely significant improvements of +0.31 BLEU, +0.28 METEOR and +1.42 KRS. TER is the only metric that worsens, and considerably so with +1.48 point. This system is also superior to SAMT for BLEU and METEOR, but not for TER and KRS. SAMT achieves the highest improvement of KRS, namely 1.91 point higher than the HIERO baseline. Just like the reordering labeled systems, SAMT also loses performance on TER over HIERO though.

5.6.5 Summary of findings from experiments

We may summarize the experiments with the following conclusions:

- Whereas for German–English and English–German the performance improvement is rather modest, for Chinese–English we see considerable improvements and overall the best results for the combination of elastic-substitution decoding, with the *Fine* 1^{st}-order variant of the labeled systems using basic plus sparse label-substitution features (Hiero-1^{st}-Sft_{B+S}).

- Crucial for performance is the use of a soft-constraint approach to label matching, as opposed to strict-matching.

- Particularly interesting is the comparison to the ITG+ labeled variant of our scheme. While ITG+ labeling already obtains improved performance, we do see that a more elaborate labeling scheme (as simple as our bucketing) may bring about even further improvement.

- In the setting were the sparse label substitution feature set is used, the coarse variant of the 1^{st}-order labels works as good as (for Chinese–English) or better than (for English–German) the fine variant. However, with only basic label substitution features the fine variant Hiero-1^{st}-Sft_B is superior to the coarse variant Hiero-$1^{st}{}_{Coarse}$-Sft_B. This may be explained by the fact that refining the

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length
	German–English					
Hiero	23.67	31.27	16.32	61.19	65.79	99.18
Hiero-0^{th}-Sft$_B$	23.99 ▲H	31.36 ▲H	16.60	61.57 ▼H	**66.30** ▲H	100.65 ■H
Hiero-0^{th}-Sft$_{B+S}$	24.16 ▲H	**31.40** ▲H	16.68	61.07	66.03 △H	99.89 ■H
Hiero-1^{st}-Sft$_B$	24.15 ▲H	31.37 ▲H	16.67	61.09	66.01	99.87 ■H
Hiero-1^{st}-Sft$_{B+S}$	**24.32** ▲H	31.39 ▲H	**16.78**	**61.02** △H	66.00	**99.94** ■H
	English–German					
Hiero	15.71	37.29	15.87	68.58	64.06	97.57
Hiero-0^{th}-Sft$_B$	16.10 ▲H	37.51 ▲H	16.08	68.74	64.63 ▲H	98.35 ■H
Hiero-0^{th}-Sft$_{B+S}$	**16.19** ▲H	37.53 ▲H	16.05	68.68	64.64 ▲H	98.26 ■H
Hiero-1^{st}-Sft$_B$	16.07 ▲H	**37.60** ▲H	16.11	68.33 ▲H	64.45 △H	97.74 □H
Hiero-1^{st}-Sft$_{B+S}$	16.04 ▲H	37.59 ▲H	**16.19**	**68.29** ▲H	**64.65** ▲H	97.88 ■H
	Chinese–English					
Hiero	20.23	27.88	9.50	66.57	58.56	98.60
Hiero-0^{th}-Sft$_B$	20.26	**27.98** ▲H	**9.66**	66.10 ▲H	59.00 ▲H	98.43 □H
Hiero-0^{th}-Sft$_{B+S}$	20.31	27.94 ▲H	9.63	66.31 ▲H	**59.04** ▲H	98.64
Hiero-1^{st}-Sft$_B$	**20.51** ▲H	27.91	9.51	**66.36** ▲H	58.82	98.42 □H
Hiero-1^{st}-Sft$_{B+S}$	20.49 ▲H	27.93	9.55	66.81 ▼H	58.78	**98.78** □H

Table 5.9: Results for extra analysis translation experiments using only a Unigram Language Model.[18]

reordering labels and using sparser label substitution features both have similar effects in refining the conditioning context of rules. This suggest that in order to get the best results, the combined effect on the conditioning context of both reordering labels and label substitution features needs to be optimized for, rather than considering these factors separately.

- Finally, the different reordering labeled systems outperform SAMT on BLEU and METEOR and also for TER and/or KRS. Interestingly, the reordering-labeled grammars are comparable in size to Hiero's, i.e., less than one third of SAMT grammar size.

In conclusion, we find it encouraging that our automatic labeling approach, which does not demand additional (monolingual) resources beyond a parallel corpus[29], gives comparable or better performance improvement to syntax-labeling approaches. We hypothesize that the two types of labeling capture complimentary reordering information, particularly that target syntax in SAMT allows more fluent MT output, strengthening the target language model.

[29] As usual in contemporary SMT, our approach also needs an adequate target language model in the complete system in order to achieve state of the art performance. In particular, as is also the case for syntactic labeling approaches, we do not aim to replace the language model with our labels. We rather build upon the already reasonable translation afforded by a good language model, and strive to use this as a basis to improve performance further.

5.7 Analysis

In this section we will give a deeper analysis of the qualitative results obtained and discussed so far. We have seen how soft reordering constraints can significantly improve the results. These constraints are effectuated by using bilingual reordering labels in combination with elastic-substitution decoding decoding and label-substitution features. The question is how exactly these constraints contribute to the performance. We will focus on four dimensions of analysis, each dimension chosen to shed light on a different aspect of the effect of the reordering constraints. The dimensions we will look at are:

1. Interaction between reordering constraints and language model: to what extent does the function of soft reordering constraints overlap with the function of a strong language model, and to what extent does it add information that even a strong language model cannot capture?

2. Basic informativeness of the labels: are the used labels at all informative from the point of view of conditional probability? That is, do we observe non-uniform distributions for the conditional probabilities of gap labels given left-hand-side parent labels?

3. Label usage and glue rule usage: do more successful systems tend to produce relatively more matching substitutions? Is there a clear change in the relative usage of glue rules across systems?

4. Can we get some qualitative understanding of where the quantitative improvements come from, and whether these improvements are valid?

Each of these two dimensions of analysis will now be discussed in detail in the following subsections.

5.7.1 An experiment with a unigram language model: How good is the reordering model?

In this subsection we try to better understand the interaction between reordering model and language model. Here we contrast the experiments from the preceding section with new experiments with the same SCFG-based reordering models but integrated with a unigram LM (instead of a 4-gram LM).[30]

A unigram language model informs about prior lexical preferences but leaves the word order to the SCFG.based reordering model. This should provide some insight

[30]In all experiments with 4-gram language models we have used *kenlm* (Heafield, 2011).But for the unigram language model experiments, we used *berkeleylm* instead (Pauls and Klein, 2011), since *kenlm* did not support working with unigram language models.

into two aspects: (1) the importance of the LM for final performance, and (2) the role the bilingual labels in affecting final word order choices.

Table 5.9 shows the results for these experiments on German–English, English–German and Chinese–English and clearly the results dropped substantially from the experiments in the preceding section with a 4-gram LM. Some specific remarks per language pair are due.

German–English and English–German: The relative improvement of the labeled systems over the Hiero baseline has increased to +0.65 BLEU point for German–English and +0.48 BLEU point for English–German respectively. Similar increase in improvement can also been seen for METEOR, BEER and KRS. The labeling seems to give a better reordering model, albeit the 4-gram LM seems to catch up with it and reduce the margin of improvement.

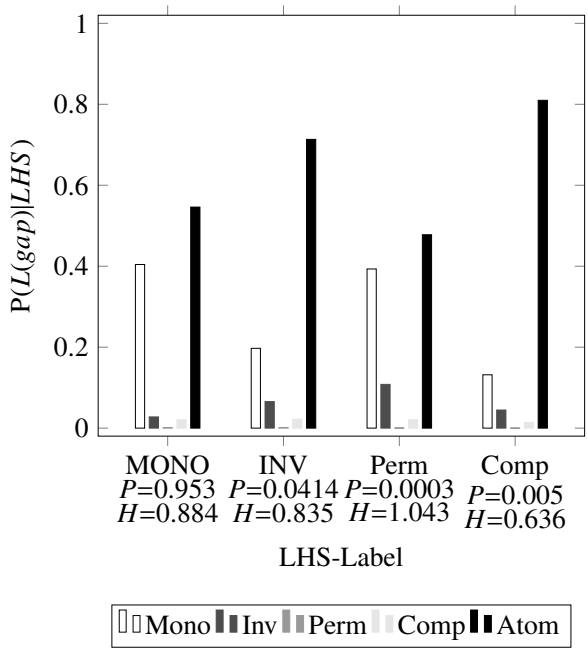
Chinese–English: For Chinese–English the relative improvements over the baseline are considerably smaller in the unigram LM experiments relative to the 4-gram LM. Nevertheless, the best system still achieves approximately +0.3 BLEU improvement while also improving TER with approximately -0.2. Apparently, the labeled reordering model here is better than Hiero in reranking the hypothesis but there is a set of top-ranking hypothesis that cannot be differentiated well without a strong LM.

The drop for Chinese–English (-12 BLEU) is markedly larger than the drop for German–English and English–German (about -5 BLEU for both) suggesting that the 4-gram LM could be specifically important for discriminating between the top ranking reorderings among the hypothesis translations for Chinese–English.

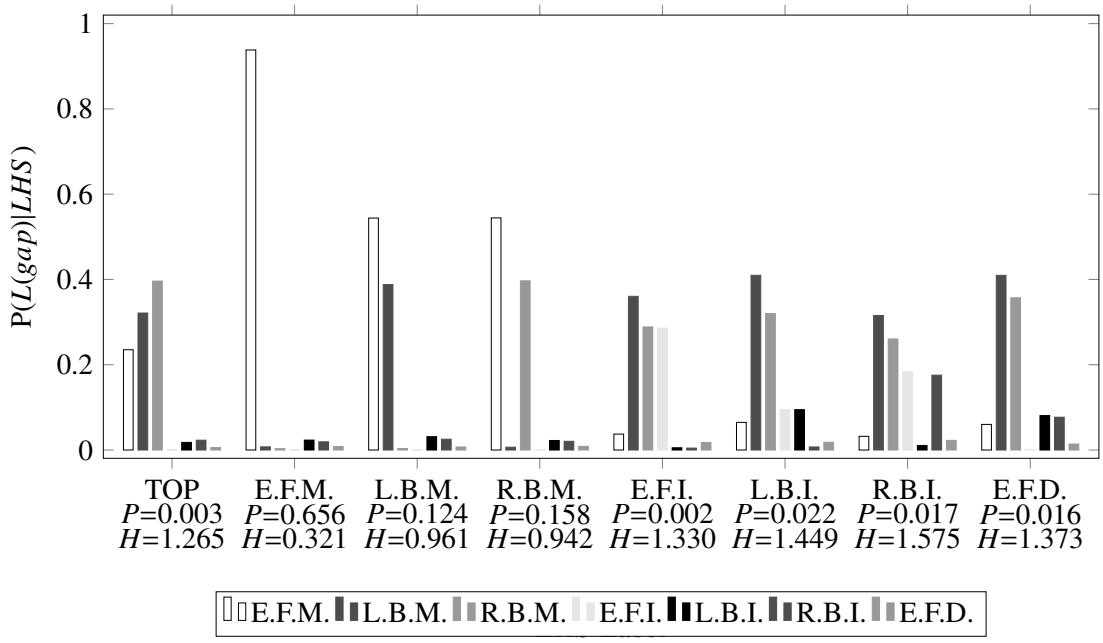
5.7.2 Label Conditional Probability Analysis

As a sanity check of the bilingual labels, we investigate the conditional probabilities of gap labels given their left-hand-side parent labels. The motivation is that if the labels add information, we expect conditional probabilities that clearly reveal preferences for a certain gap label given a left-hand-side parent. If the labels are not informative, almost uniform distributions are to be expected.

Figure 5.9 shows two bar charts of conditional probabilities (relative frequency estimated) of gap labels given their left-hand-sides over grammar rules filtered on a dev-set. Both the 0^{th}-order (a) and the 1^{th}-order Markov labels (b) show far sharper distinctions than uniform which Hiero's single label assumes; this is also shown in the far lower entropy for parent-child relations in labeled Hiero rules as compared to the maximum entropy base expectation for uninformative labels. Figure 5.9 (a) shows that in the case of 0^{th}-order labels, sequences of monotone embedded phrases are very likely, whereas a permutation/complex labels is not likely to contain another complex phrase. Figure 5.9 (b) shows some interesting patterns in the case of 1^{th}-order labels. First there is the pattern that any label, be it some form of monotone, inverted or fully discontinuous gives high probability to produce gap labels of some monotone type. This captures the general trend that monotone is more likely in general, also seen for the 0^{th}-order labels. Somewhat more remarkable is the trend for left/right-



(a) Weighted entropy labeled H_{avg}=0.882. Entropy uniform H_{max}=1.609



(b) Weighted entropy labeled H_{avg}=0.568.
Entropy uniform H_{max}=1.946

Figure 5.9: Conditional probabilities gap label given left-hand-side (LHS) $P(L(gap)|LHS)$, entropy (H) and prior probabilities for the left-hand-sides (P).

binding labels to produce again the same left/right-binding labels, and not labels of the same type but with opposite binding direction. For example, left-binding-inverted (L.B.I) has high chance to again produce left-binding-inverted, but not right-binding-inverted (R.B.I). Analogous for the right-binding and monotone cases. In the case of inverted labels we think this captures the fact that it is not likely that an inversion operation is directly mostly undone by another consecutive inversion operator in real data. In the case of monotone labels, producing two monotone labels with different binding directions is unlikely since in most cases it would imply that the gap label is in fact embedded by monotone parents on both sides, which would yield the label E.F.M. instead. Another observation is that all labels involving inversion are most likely to produce left-binding-monotone (L.B.M.) labels as gap labels. Some of these observations are not too surprising as they can be—at least partially—explained from the way the labels are chosen. Nevertheless, this actually only reinforces the intuition that the design of the labels indeed allows them to enforce the word order contexts of phrases to be coherent. Since encouraging coherence of reordering is exactly the goal of the labels, we conclude that the picture that arises from this sanity check is encouraging.

5.7.3 Label Usage Analysis

In this section we give additional support for the hypothesis that systems that benefit from the use of reordering labels do so by learning a relative preference for matching substitutions of these labels. This will be done by performing a direct analysis of label-substitution behavior, as measured in the translations produced for the development and test sets in our experiments.[31]

Historically, labels that have been used to improve translation were mainly syntactic and used in a strict matching setting, e.g. (Zollmann and Venugopal, 2006). Even in this strict matching setting such labels often gave significant improvements over a Hiero baseline. In fact we saw that for certain language pairs and systems reordering labels also give improvement when used in a strict matching setup. And for Chinese–English for the phrase-centric reordering labeled systems this improvement, while small, is already statistically significant. But we have seen in our results that using reordering labels as soft constraints works much better, and this is in line with observations by others, e.g. (Chiang, 2010; Xiao and Zhu, 2013; Almaghout et al., 2012).

Labels used as soft constraints are in a way similar to labels used as strict constraints, except that they can be ignored to a smaller or larger extend when the tuner learns that this increases performance.[32] Intuitively we expect labels yielding

[31] We would like to thank Wilker Aziz for the suggestion that measuring label-substitution behavior directly in this way is more appropriate and reliable than inferring it indirectly based on the learned weights of label-substitution features.

[32] In theory the tuner could even implicitly learn that certain labels can be associated to groups, substituting relatively freely (without significant penalty) to labels within their group, but suffering a

System Name	Match	NoMatch	Glue Rule Subst	dev BLEU	test BLEU
	Chinese–English				
Hiero Baseline	0.76	—	0.24	31.70	31.63
Hiero-0^{th}-Sft$_B$	0.37	0.38	0.25	32.04	32.20
Hiero-0^{th}-Sft$_{B+S}$	0.36	0.36	0.28	32.23	32.43
Hiero-1^{st}-Sft$_B$	**0.55**	0.22	0.23	32.63	32.61
Hiero-1^{st}-Sft$_{B+S}$	0.49	0.26	0.25	**33.02**	**32.69**
	German–English				
Hiero Baseline	0.62	—	0.38	27.90	28.39
Hiero-0^{th}-Sft$_B$	0.33	0.35	0.31	28.01	28.45
Hiero-0^{th}-Sft$_{B+S}$	0.35	0.31	0.34	28.20	**28.57**
Hiero-1^{st}-Sft$_B$	0.43	0.20	0.37	28.13	28.45
Hiero-1^{st}-Sft$_{B+S}$	**0.44**	0.23	0.33	**28.32**	28.47
	English–German				
Hiero Baseline	0.71	—	0.29	20.36	20.89
Hiero-0^{th}-Sft$_B$	**0.50**	0.19	0.31	20.57	21.01
Hiero-0^{th}-Sft$_{B+S}$	0.46	0.23	0.31	**20.75**	20.97
Hiero-1^{st}-Sft$_B$	0.44	0.24	0.32	20.54	**21.05**
Hiero-1^{st}-Sft$_{B+S}$	0.43	0.24	0.33	20.70	20.90

Table 5.10: Relative frequencies of rule substitutions and corresponding dev and test BLEU scores for different systems and language pairs. The frequencies are computed as means over the observed translations in N-Best lists (N=300) for three runs per system. Match/NoMatch are the relative frequencies of matching/mismatching rule substitutions, excluding substitutions to glue rules. Glue Rules Subst is the relative frequency of rules that substitute to glue rules. For every language pair the highest occurring relative frequency of matching substitutions, as well as the highest dev BLEU and test BLEU scores are shown in **bold**.

relatively more helpful constraints on the way rules may be combined, to induce relatively higher preference for matching label substitutions. Conversely we expect improvements from systems that add (reordering) labels as soft constraints but are otherwise the same as Hiero (our used setup) to come from a learned relative preference for matching substitutions of these labels. What label set works best can vary across languages, but we think that for every language pair and label set improvements must come from preferring certain substitutions over others and in particular matching over mismatching substitutions.

This gives the following more specific hypotheses:

- H1: The relative preference for matching substitutions (with the best performing labels) must be higher for the language pair were the highest improvements are achieved[33], than the relative preference for matching substitutions (with the best performing labels) for other language pairs.

- H2: The best performing label set for a language pair also yields the relatively

high penalty when substituting to labels outside it. Thus constitutes a kind of implicit clustering of labels through the learning of substitution preferences.

[33] Chinese–English in our case.

highest label matching preferences.[34]

Additionally we want to check if reordering features also lead to a relative decrease in the amount of glue rules that is used in comparison to HIERO. Knowing this will help to further clarify the possible causes for our improvements.

Label substitution statistics

We next introduce the statistics used to test our hypotheses and explain how it was created. Table 5.10 shows statistics of label substitution and substitution to glue rules in a selection of our experiments. The first three columns show the relative frequencies (fractions) of matching rule substitutions, mismatching rule substitutions and substitutions to glue rules. All three categories are mutually exclusive and their relative frequencies add up to one. In case of the HIERO baseline, the distinction between matching and mismatching rule substitutions is not applicable as rules are trivially matching. This is because there is only one label "X" for HIERO rules and also strict matching is enforced. In addition to the relative label-substitution frequencies, we show the BLEU scores on the dev- and test-set for all systems in two parallel columns. This information will further help in testing our hypotheses.

The relative substitution frequencies are computed by summing over substitution counts in the N-best lists (N = 300) produced during translation, summing over different translations and runs. Computation is done by marking and combining information of the feature values of label-substitution feature count features and the various rule count features from the derivations. Here we report the statistics for the test-set, but the statistics for the dev-set are very similar, showing at most one percentage point differences in some cases.[35] Additionally we computed statistics using only the 1-best instead of N-best translations, but again for brevity we omit this table here since the difference with N-best was only marginal: at most 1 percentage point in some cases.

Interpretation of the statistics

We notice from the statistics in table 5.10 that the relative frequency of glue rule substitutions is mostly similar for the reordering labeled systems and the HIERO baseline. This negates the hypothesis that a mere change in the relative frequency of hierarchical rule substitutions over glue rule substitutions explains the differences in

[34] It is possible that during tuning labels work well on the dev set and consequently a high matching preference is learned. But on the test-set this preference can sometimes turn out to be too strong: overfitting. As we did not properly control for overfitting by using a held out set to choose the best tuning settings for testing, we therefore believe that for this analysis it is more reasonable to look at dev instead of test results, when assessing the validity of this hypothesis.

[35] Also, for brevity standard deviations are omitted because they do not add much extra information for the purpose of our analysis plus they are all similar, somewhere in the range between 4.1E-2 and 1.1E-1.

performance. We also notice that the relative frequency of matching labels (*Match*) is in nearly all systems at least as high as the relative frequency of mismatching (*NoMatch*), while it is nearly twice that of matching for Chinese–English and German–English for the top performing systems on the test-set. When choosing instead the best system based on scores on the dev-set, the latter holds for all three language pairs. We conclude from this that the first intuition that improvements are mainly achieved by learning preference for matching substitutions of effective labels appears to be validated by the results.

Moving on to the more specific hypotheses, looking at the relative matching preference for the best test/dev system for Chinese–English 0.49/0.49 versus the preference for German–English 0.35/0.44 and English–German 0.44/0.46, we notice that the results give no ground to give up H1. Looking next at the second hypothesis H2, we see that for all three language pairs, the system with the highest relative matching frequency also produces the highest dev score. Both in case of H1 and H2, the hypotheses are supported by the data, while the data also remains insufficient to draw statistical conclusions from it. If we consult the test instead of dev scores H2 will not hold, since the best scores for German–English and English–German are obtained by systems with labels that do not yield the highest relative matching scores. These language pairs display an apparent overfitting during learning, whereby the systems performing best on the dev-set fall below on the test-set. We think that a too strong or too specific matching preference is learned for these systems, which is suboptimal on the test-set. We believe this effect might disappear when instead of using the dev-set scores, scores for held-out data is used to choose which iteration to take the weights from during tuning, as proposed by (Chiang, 2010).

As a last observation, we note that for Chinese–English and German–English, the best performance is obtained by using the basic and sparse features, which in both cases slightly modifies the matching preferences by effectively allowing them to take more context into account, which leads to an effective reduction in the amount of matches for Chinese–English and a slight increase for German–English. Sparse label-substitution features thus allow to fine-tune the label-substitution behavior, and they increase dev scores compared to systems with the basic label-substitution features, for all three language pairs and both label types. But these features yield only modest additional gains on the test set for most systems, while even decreasing performance in case of English–German. This suggests that these features, while helpful, also increase the risk of overfitting, something that should ideally by protected against by using a held-out data for final weight selection.

Discussion

In this analysis we have focused on the relative differences in the frequency of matching label substitutions across systems and also the difference between relative frequency of matching and mismatching substitutions. Yet the question remains, what the baseline frequency for matching substitutions is, when implicit canonical form

labels for HIERO rules are marked, but no explicit labels nor soft constraints are used.[36] This is an open question. Using a simple distribution P_{naive} that naively assumes all labels are equally likely in all contexts the a-priori chance for matching labels is just $\frac{1}{N}$ while the chance for mismatching is $\frac{N-1}{N}$. This would suggest that even just having an equal relative frequency for matching and mismatching substitutions indicates a much higher preference for matching than what could be expected by mere chance. On the other hand, matching substitutions probably also correlate with better outputs according to a strong language model. The unigram language model experiments in the last section showed that the relative benefit of reordering labels increases when the language model is weaker. This suggests that when using a strong language model rules will to some extend automatically be combined in a way that yields good translations, which most likely involves a higher amount of matching than predicted by P_{naive}. This makes it hard to predict the expected relative frequency of matching substitutions, for a system that is (implicitly) labeled but that does not use label-substitution features. For this reason we have focused here on the relative matching frequencies of labeled systems. Our current analysis unfortunately does not constitute a definite proof that relative matching is increased over the baseline (the latter would require baseline relative frequencies for matching substitutions[37]). But it shows that stronger matching preference coincides with better dev-scores and language pairs for which larger relative improvement over the baseline is attained. This seem to add evidence to the intuition that improvements are made by learning a substitution preference that predominantly encourages matching substitutions, as opposed to learning some arbitrary other preference pattern.

In this section we took yet another view on analyzing our results. Rather than establishing that improvements are made or researching the interaction with the language model as was done in the last subsection, here we assessed whether basic beliefs about the origins of improvement are validated by observation. Despite its shortcomings, we think this analysis adds insight by providing more evidence for the intuition that improvements in our soft-matching labeled systems are broadly made by learning a preference for matching substitutions. This coincides with giving more preference to reordering patterns as they were seen in the training data. It therefore yields additional evidence for our main hypothesis in this work, that reordering labels can facilitate more coherent and more context sensitive reordering decisions, thereby improving word order in hierarchical translation.

[36] An actual baseline for the frequency of matching labels could be estimated as the frequency of matching substitutions produced by implicitly labeled HIERO rules, assuming every HIERO rule has an associated single set of hidden (canonical form) reordering labels. One way to compute it is to adapt the decoder to output derivations instead of translations and rerunning the experiments, and then matching the unlabeled rules in the produced derivations back to labeled rules in (reordering) labeled grammars. Alternatively one could translate with reordering labeled grammars and soft constraints, but set the weight of all soft constraints to 0, and not allow the tuner to change it. Both approaches are conceptually simple but technically hard as well as time intensive.

[37] As computing these frequencies is technically complicated, we leave it for future work.

Sentence type	Sentence contents
Source Sentence	泰 还 未 摆脱 危机
Reference1	france drawing up military withdrawal plan from bosnia-herzegovina
Reference2	france studying plan to withdraw its troops from bosnia-herzegovina
Reference3	france studies plan to withdraw troops from bosnia and herzegovina
Reference4	france considering troops withdrawal from bosnia and herzegovina
Hiero	law research plan for withdrawal from bosnia
Hiero-1st-Sft$_{B+S}$	law is studying plans to withdraw its troops from bosnia and herzegovina
Source Sentence	一 位 南韩 政府 官员 说 ， 昨天 南韩 已 向 北韩 发出 邀请 ， 请 他们 派 观察员 来 观摩 这 次 军事 演习 。
Reference1	a south korean government official said that south korea issued an invitation to north korea yesterday , asking them to send observers to watch this military exercise .
Reference2	a south korea government official said south korea had already sent an invitation to north korea yesterday asking them to send observers to view and learn from the military exercise .
Reference3	a south korean government official said that south korea offered an invitation to north korea yesterday to send their observers to watch the military exercise
Reference4	a south korean official said that south korea sent invitation to north korea yesterday , asking it to send observers to the drills .
Hiero	a south korean government official said that south korea has issued invitations to north korea , yesterday invited them to attend the military exercises as observers .
Hiero-1st-Sft$_{B+S}$	a south korean government official said that south korea has issued invitations to north korea yesterday , asking them to send observers to attend the meeting of the military exercise
Source Sentence	外交 人员 将 搭乘 第五 架 飞机 返国 。
Reference1	diplomatic staff will take the fifth plane home .
Reference2	diplomatic staff would go home in a fifth plane .
Reference3	diplomats are to come back home aboard the fifth plane .
Reference4	diplomatic staff would be airlifted on a fifth plane .
Hiero	diplomats fifth aircraft will fly for repatriation .
Hiero-1st-Sft$_{B+S}$	diplomatic personnel will travel on to the fifth aircraft for repatriation .

Table 5.11: Source sentence, reference, baseline and best system output for sentences 959, 1442 and 1239 from the Chinese–English testset. These are amongst the test sentences with the highest improvement on METEOR (leaving out some very short sentences and sentences with unknown words).

5.7.4 Qualitative analysis

A qualitative analysis can give some additional insight about what is going in the actual translations, which quantitative scores fail to provide. On the other hand qualitative analysis has the disadvantage of being biased and relying on a very small sample. While we cannot really alleviate the drawback of a small sample here, we do try to alleviate the problem of selection bias by looking at some of the most improved test sentences according to METEOR as well as some test sentences that gave the highest drop in METEOR score. We used METEOR as opposed to BLEU for the selection, because it is better as a sentence-level score (Macháček and Bojar, 2013), and has a more explicit reordering component. By considering an equal number of improved and worsened examples in this way, selected by a clear criterion (highest gain/drop in METEOR score), we hope to be able to form a somewhat more objective opinion based on the selected examples. Here we chose to look at Chinese–English examples, since Chinese–English is the language pair where we observe the greatest benefits from our method in terms of improving word order, as indicated by BLEU, METEOR and KRS improvements.

Looking first at the three improved examples, we see that in all three cases there is a clear improvement in word order (structure) as well as lexical selection. Looking next at the examples where performance worsens, the errors seem to be mainly in

Sentence type	Sentence contents
Source Sentence	埃及 航空 于 今年 1 月 正式 开通 首 条 开罗 至 北京 的 直飞 航线 。
Reference1	egyptair officially opened the first direct flight route from cairo to beijing in january this year .
Reference2	egypt air set up the first direct flight between cairo and beijing in january this year .
Reference3	the egyptian airline officially opened its first direct flight from cairo to beijing this january .
Reference4	air egypt formally opened the first direct flight line from cairo to beijing in january .
Hiero	egypt air in january this year was officially opened its first direct flight from cairo to beijing .
Hiero-1st-Sft$_{B+S}$	egyptian aviation was officially opened in january this year . the first non-stop service from beijing to cairo .
Source Sentence	颁奖 仪式 在 菲律宾 文化 中心 隆重 举行 。
Reference1	the awarding ceremony was solemnly held at the philippines cultural center .
Reference2	the award ceremony was solemnly held in the philippine cultural center .
Reference3	the presentation ceremony was solemnly held at the philippine culture center .
Reference4	the awarding ceremony was performed solemnly in philippine cultural center .
Hiero	the award ceremony held in the philippines cultural center grand .
Hiero-1st-Sft$_{B+S}$	a ceremony held at the cultural center grand .
Source Sentence	这些 国家 已经 停止 进口 巴拉圭 牛肉 。
Reference1	these countries have already stopped beef imports from paraguay .
Reference2	these countries have suspended the import of paraguayan beef .
Reference3	these countries have stopped importing beef from paraguay .
Reference4	these countries have stopped importing beef from paraguay .
Hiero	these countries have to stop importing beef in paraguay .
Hiero-1st-Sft$_{B+S}$	these countries have put an end to the paraguayan beef imports .

Table 5.12: Source sentence, reference, baseline and best system output for sentences 1870, 937 and 1833 from the Chinese–English testset. These are amongst the test sentences with the highest performance loss on METEOR (leaving out some very short sentences and sentences with unknown words).

lexical selection. A reason could be that the labeled system assigns a relatively lower weight to the language model, which may make it more susceptible to make such lexical selection mistakes. At the same time these examples do seem to support the expected increased capability in getting right the global reordering structure of the reordering labeled system.

While the drawbacks of qualitative analysis discourage us to draw strong conclusions from this, the examples do seem to give some additional support for the thesis that in Chinese–English translation reordering labels help to improve word order and global sentence structure. This improvement seems to sometimes come at a price in the quality of lexical selection, possibly due to the relatively lower weight of the language model in comparison to the HIERO baseline.

5.7.5 Summary of findings from analysis

The conclusions from these different analyses can be summarized as follows:

- There is overlap between the work done by a strong language model and work done by the labeled reordering models. However, HIERO's performance depends to a larger extent on the 4-gram language model than the labeled reordering systems, suggesting that the latter systems give more adequate reordering.

- The label conditional probability analysis shows that for both label types, the conditional probability of gap labels given their left-hand-side parent is far from

uniform, and shows clear patterns. This is another modest source of evidence that our reordering labels are sensible and informative.

- The label usage analysis seems to validate the hypothesis that larger improvements over the baseline typically go together with relatively more matching substitutions. While this analysis in the current form is insufficient to draw statistical conclusions, it is good to know that basic intuitions about the source of improvement of soft constraints for label matching are at least not violated by the data.

- The qualitative analysis of a small number of examples shows that also on the qualitative level there is evidence that the labels are effective in improving both reordering and lexical selection.

5.8 Related Work

5.8.1 Syntax-based labels

A range of (distantly) related work exploits syntax for Hiero models, e.g. (Liu et al., 2006; Huang et al., 2006; Mi et al., 2008a; Mi and Huang, 2008; Zollmann and Venugopal, 2006; Almaghout et al., 2010, 2012; Li et al., 2012b). In terms of labeling Hiero rules, SAMT (Zollmann and Venugopal, 2006; Mylonakis and Sima'an, 2011) exploits a "softer notion" of syntax by fitting the CCG-like syntactic labels to non-constituent phrases.

5.8.2 Label clustering methods

Approaches to automatically coarsen the label set used by SAMT are explored by Hanneman and Lavie (2011, 2013). In this approach, the similarity between conditional probability distributions of labels is used to merge labels. The conditional probability of a source label s_i given a target label t_j is computed with simple relative frequency estimation, counting the frequency of s_i and t_j together and dividing by the total frequency of t_j in combination with any source label $s_i \in S$. The computation of conditional probabilities for target labels given source labels goes analogous. Based on these distributions $L1$ distances are computed for all pairs of labels, in source-to-target and target-to-source direction. Finally, the pair of labels with the smallest $L1$ distance between corresponding label distributions in either direction, is merged. This is further improved upon by Mino et al. (2014) who propose an alternative clustering algorithm based on the exchange algorithm (Uszkoreit and Brants, 2008), which obtains comparable results, but which runs an order of magnitude faster.

5.8.3 Soft constraints

Soft syntactic constraints have been around for some time now (Zhou et al., 2008; Venugopal et al., 2009; Chiang, 2010; Xiao and Zhu, 2013). Zhou et al. (2008) reinforce Hiero with a linguistically motivated prior. This prior is based on the level of syntactic homogeneity between pairs of non-terminals and the associated syntactic forests rooted at these nonterminals, whereby tree kernels[38] are applied to efficiently measure the amount of overlap between all pairs of sub-trees induced by the pairs of syntactic forests. Crucially, the syntactic prior encourages derivations that are more syntactically coherent but does not block derivations when they are not. Venugopal et al. (2009) associate distributions over compatible syntactic labelings with grammar rules, and combine these preference distributions during decoding, thus achieving a summation rather than competition between compatible label configurations. The latter approach requires significant changes to the decoder and comes at a considerable computational cost. Soft constraints as proposed by (Chiang, 2010) and adopted in this paper were discussed in earlier in subsection 5.2.2 and will not be repeated here. Xiao and Zhu (2013) focus on unsupervised learning of sub-tree alignment based on synchronous tree-substitution grammars in combination with the *expectation maximization* (EM) algorithm (Dempster et al., 1977) or a Bayesian learning approach. The translation approach in their work in contrast to ours is based on tree-to-tree translation. It uses syntax on both sides and works with rule sets that even with the labels removed still differ significantly from HIERO. But in line with our work, this work also requires elastic-substitution decoding (Chiang, 2010) to get the best results.

5.8.4 Learning labels

Improving the quality of the extracted syntactic rules and their labels for syntactic translation with the help of the EM-algorithm is explored by Wang et al. (2010). This work uses re-structuring: binarization of syntactic trees, to make more translation patterns available. It also uses re-labeling to improve the adequacy of syntactic labels and re-aligning to improve word alignments.

Learning labels in a robust way is also explored by Mylonakis and Sima'an (2011). This work uses a special variant of the EM algorithm called Cross-Validated EM to avoid the standard problems of EM with overfitting. The algorithm is then used to learn a distribution of source-labeled hierarchical rules with labels of different levels of specificity. The labels are based on the SAMT labeling approach and also include some basic information about relative orientation with respect to parent rules.

Learning latent-variable SCFG. for hierarchical translation is explored by Saluja et al. (2014). This work uses spectral learning or the EM algorithm to learn tensors that capture the latent variable information of rules. These are used by means of tensor-

[38] Informally, tree kernels are operators that efficiently compute a function $K(T, T')$ of two input tree arguments T and T', for example the number of common subtrees. The efficient computation of the function by tree kernels is often achieved by a form of dynamic programming.

vector products, somewhat similar to the way label preferences are propagated in (Venugopal et al., 2009). Learning of labels is done based on the covariances between sparse feature vectors for inside and outside trees for rules in the training corpus.[39] The work uses minimal rules to avoid the complex problem of simultaneously learning the latent variables and the segmentations of word alignments.

5.8.5 Bilingual Language Models

Niehues et al. (2011) propose bilingual language models as another way to add more context to the translation process. The bilingual language models presented in this work are based on tokens that combine target words with the words in the source aligned to those target words. Applied to phrase-based translation, this model gives improvements on various language pairs. In addition to this, a bilingual language model that uses POS-tags of the source and target words instead of the words themselves is used to further improve performance.

Garmash and Monz (2014) propose an extension to this approach, whereby dependency parsing is applied to select a richer set of source context POS-tags for each target word using the dependency information in addition to the alignment information. Because this approach as well as the approach of Niehues et al. (2011) uses at any time only source information and information of already translated target words, both approaches can be directly integrated in the decoder without requiring additional approximations. In a similar spirit, Garmash and Monz (2015) propose a new way to adapt the *structured language model* proposed in (Chelba and Jelinek, 2000) to SMT, and use the resulting bilingual language model to improve in particular the word order of output translation.

A clear difference with our own approach, apart from focusing on phrase-based instead of hierarchical phrase-based translation, is that these works focus on implicit notions of word order as induced by the state sequences of the bilingual language models in combination with syntactic information. Our model in contrasts puts the hierarchical reordering structure central. It focuses on improving the coherence of composition as hierarchical rules are combined, and requires no syntactic information.

5.8.6 Improvement and evaluation of reordering with permutation trees

Stanojević and Sima'an (2015) propose a method for inducing reordering grammars based on permutation trees (PETs) for preordering. Their work uses a modified form

[39] The concepts *inside* and *outside tree* are defined in terms of another concept called *skeletal tree*. The skeletal tree for an aligned sentence pair is the synchronous tree composed of the set of synchronous rules in the derivation of the aligned sentence pair. Since only minimal rules are used, there is always only one unique derivation. The inside tree for a rule in the training contains the entire sub-tree at and below the left-hand-side nonterminal, and the outside tree is everything else in the synchronous skeletal-tree except the inside tree.

of PETs (Gildea et al., 2006) in combination with variable splitting for the permutation labels of PET nodes (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006). The reported results show significant improvements over no preordering, a rule-based preordering baseline (Isozaki et al., 2010) and an ITG-based preordering baseline (Neubig et al., 2012). Usage of all PETs yields better results than working with a single PET in the reported experiments. This work is relevant in the context of ours because it also shows that working with PETs gives significant improvement over using only ITG reordering operators. There are large differences with our work. Our work uses all HATs in combination with bucketing to form labels for elastic-substitution decoding, improving hierarchical translation within the decoder. Stanojević and Sima'an (2015) instead restrict the set of used HATs to only PETs (bijective mappings), and learn the labels. Nevertheless, both contribute evidence to the thesis that word order can be significantly improved without using syntax.

Stanojević and Sima'an (2014a) propose a new and highly successful machine translation evaluation method called BEER. This metric uses a multitude of weighted features, with weights that are directly trained to maximize correlation with human ranking. As such, the metric shows very high correlation with human evaluation of translation performance. Training is done for pairwise rankings using learning-to-rank techniques in a way that is similar to PRO MT system tuning (Hopkins and May, 2011). Some of the successful new features that are proposed are character n-grams and features based on PETs. The latter features are concerned with reordering and turn out to be an important component in the success of the metric. In the context of this work, the effectiveness of PETs in characterizing the correctness of translation word order as part of the complete evaluation, gives yet another ground to believe that the information present in PETs and more generally HATs may be particularly suitable for improving word order in SMT.

5.9 Conclusion

We presented a novel method to enrich Hierarchical Statistical Machine Translation with bilingual labels that help to improve the translation quality. Considerable and significant improvements of the BLEU, METEOR and KRS are achieved simultaneously for Chinese–English translation while tuning on BLEU, where the Kendall Reordering Score is specifically designed to measure improvement of reordering in isolation. Significant improvements of the BLEU score are achieved for both German–English and English–German translation, while the Kendall Reordering Score shows significant improvement for English–German. Our work differs from related approaches that use syntactic or part-of-speech information in the formation of reordering constraints in that it needs no such additional information. It also differs from related work on reordering constraints based on lexicalization in that it uses no such lexicalization but instead strives to achieve more globally coherent translations, afforded by global, holistic constraints that take the local reordering history of the

derivation directly into account. Our experiments also once again reinforce the established wisdom that soft, rather than strict constraints, are a necessity when aiming to include new information to an already strong system without the risk of effectively worsening performance through constraints that have not been directly tailored to the data through a proper learning approach. While lexicalized constraints on reordering have proven to have great potential, un-lexicalized soft bilingual constraints, which are more general and transcend the rule level have their own place in providing another agenda of improving translation which focuses more on the global coherence direction by directly putting soft alignment-informed constraints on the combination of rules. Finally, while more research is necessary in this direction, there are strong reasons to believe that in the right setup these different approaches can be made to further reinforce each other.

6

Empirical Analysis Hierarchical Translation Equivalence

"Quite the contrary. It is cognition that is the fantasy," the man cut in. "Granted, everything I tell you now is mere words. Arrange them and rearrange them as I might, I will never be able to explain to you the form of Will the Boss possesses. My explanation would only show the correlation between myself and that Will by means of a correlation on the verbal level. The negation of cognition thus correlates to the negation of language. For when those two pillars of Western humanism, individual cognition and evolutionary continuity, lose their meaning, language loses meaning. Existence ceases for the individuum as we know it, and all becomes chaos. You cease to be a unique entity unto yourself, but exist simply as chaos. And not just the chaos that is you; your chaos is also my chaos. To wit, existence is communication, and communication, existence."

– Haruki Murakami, *A Wild Sheep Chase*

This chapter conceptually consist of two parts, corresponding to two major foundations necessary to properly understand and analyze Hierarchical Translation Equivalence in an exact and formally grounded way. The first part, based on the work by Maillette de Buy Wenniger and Sima'an (2013a) focuses on the formal foundations of what it means to parse a word alignment using a synchronous grammar. The second part uses the foundational work of the first part to explain how based on sets of synchronous trees called HATs, introduced in chapter 4, alignments requiring grammar formalisms of arbitrary complexity can be analyzed and characterized in a uniform way. This permits a very general and exact notion of grammatical coverage for arbitrary grammar formalisms, and more importantly, offers a unified view on decomposition and reordering complexity that goes beyond the details of any arbitrary specific grammar formalism. We now give a short overview of both parts.

Part 1

Part 1 explains how parsing a word alignment with a synchronous grammar requires both word alignments and synchronous grammars to work with the same formalism of translation equivalence. First, given an unaligned sentence pair, a synchronous grammar is producing a set of derivations equivalent with a set of synchronous trees such that these trees are consistent with the grammar applied to the sentence pair. The nodes of the trees/derivations correspond to a set of translation equivalence units (TEUs) permissible by the grammar given the sentence pair. Second an alignment is interpreted as equivalent to a set of synchronous trees consistent with the word alignment, such that these trees are minimally complex but maximally compact in their encoding of the hierarchical translation equivalence induced by the alignment. Again the nodes of the trees consistent with the alignment correspond to a set of TEUs warranted by the word alignment. Finally, intersection of the set of TEUs consistent with the grammar and induced by the word alignment reveals whether all the TEUs induced by the word alignment are also producible by the grammar, in which case we can speak of "coverage of the word alignment by the grammar". This concept is explained and studied in much detail for the well known case of Normal Form inversion transduction grammar (NF-ITG). It is shown that the choice for what units are considered to be valid atomic (minimal) TEUs can have considerable effect on what kind of word alignments are considered "coverable by ITG".

Part 2

In part 2, the HATs introduced in chapter 4 are combined with the foundational work of the first part. It is shown that since HATs correspond to maximal decompositions of word alignments, using HATs the type of grammar formalism required to produce a word alignment under a chosen semantics of translation equivalence can be directly deduced. The HAT-based analysis allows reasoning over sets of synchronous tree/grammar formalisms of increasing complexity, while allowing certain details of the formalism within one set, such as the branching factor, to vary within certain limits. Such a formalism-group based analysis is important because it gives insight into the complexity of hierarchical translation equivalence in actual empirical data. Analysis that focuses on a single specific grammar formalism such as normal form ITG fails to deliver this.

Finally HATs allow analysis of the properties of the mapping relations in actual hierarchical translation equivalence. This can give quantitative insights into the complexity of reordering, in a way that explicit grammar-based analysis approaches cannot. HATs give insight in the specifics of the mapping relations that goes beyond the decomposition or the number of parts, which is important in this type of analysis. These specifics are made explicit in HATs by the set-permutation labels, enabling analysis at increasing levels of detail and specificity, which no other work up till now has permitted.

6.1 A formal characterization of word alignment parsing

The training data used by current statistical machine translation (SMT) models consists of source and target sentence pairs aligned together at the word level (*word alignments*). For the hierarchical and syntactically-enriched SMT models, e.g., (Chiang, 2007; Zollmann and Venugopal, 2006), this training data is used for extracting *statistically weighted Synchronous Context-Free Grammars (*SCFGs*)*. Formally speaking, a synchronous grammar defines a set of (source-target) sentence pairs derived synchronously by the grammar. Contrary to common belief, however, a synchronous grammar (see e.g., (Chiang, 2005; Satta and Peserico, 2005)) does not accept (or parse) word alignments. This is because a synchronous derivation generates a tree pair with a bijective binary relation (links) between their *non-terminal* nodes. For deciding whether a given word alignment is generated/accepted by a given synchronous grammar, it is necessary to *interpret* the synchronous derivations down to the lexical level. However, the question is how to unambiguously interpret the synchronous derivations of a synchronous grammar as word alignments. One major difficulty is that synchronous productions, in their most general form, may contain *unaligned* terminal sequences. Consider, for instance, the relatively non-complex synchronous production

$$\langle X \rightarrow \alpha\, X^{(1)}\, \beta\, X^{(2)}\, \gamma\, X^{(3)},\ X \rightarrow \sigma\, X^{(2)}\, \tau\, X^{(1)}\, \mu\, X^{(3)} \rangle$$

where superscript (i) stands for aligned instances of nonterminal X and all Greek symbols stand for arbitrary non-empty terminals sequences. Given a word-aligned sentence pair it is necessary to bind the terminal sequence by links consistent with the given word alignment, and then parse the word alignment with the thus enriched grammar rules. This is not complex if we assume that each of the source terminal sequences is contiguously aligned with a target contiguous sequence, but difficult if we assume arbitrary alignments, including many-to-one and non-contiguously aligned chunks.

One important goal of this chapter is to propose a formal characterization of what it means to synchronously parse a word alignment. Our formal characterization is borrowed from the "parsing as intersection" paradigm, e.g., (Bar-Hillel et al., 1961; Lang, 1988; van Noord, 1995; Nederhof and Satta, 2004). Conceptually, our characterization makes use of three algorithms. Firstly, parse the *unaligned* sentence pair with the synchronous grammar to obtain a set of synchronous derivations, i.e., trees. Secondly, interpret a word alignment as generating a set of synchronous trees representing the recursive translation equivalence relations of interest[1] perceived in the word alignment. And finally, *intersect* the sets of nodes in the two sets of synchronous trees to check whether the grammar can generate (parts of) the word alignment. The formal detail of each of these three steps is provided in sections 2.3.2 to 6.4.

[1]The translation equivalence relations of interest may vary in kind as we will exemplify later. The known phrase pairs are merely one possible kind.

We think that alignment parsing is relevant for current research because it highlights the difference between alignments in training data and alignments accepted by a synchronous grammar (learned from data). This is useful for literature on learning from word-aligned parallel corpora (e.g., (Zens and Ney, 2003; DeNero et al., 2006; Blunsom et al., 2009; Cohn and Blunsom, 2009; Riesa and Marcu, 2010; Mylonakis and Sima'an, 2011; Haghighi et al., 2009; McCarley et al., 2011)). A theoretical, formalized characterization of the alignment parsing problem is likely to improve the choices made in empirical work as well. We exemplify our claims by providing yet another empirical study of the stability of the ITG hypothesis. Our study highlights some of the technical choices left implicit in preceding work as explained in the next section.

6.2 First application to the ITG hypothesis

A grammar *formalism* is a whole set/family of synchronous grammars. For example, ITG (Wu, 1997) defines a family of *inversion-transduction grammars* differing among them in the exact set of synchronous productions, terminals and non-terminals. Given a synchronous grammar *formalism* and an input word alignment, a relevant theoretical question is *whether there exists an instance synchronous grammar* that generates the word alignment exactly. We will refer to this question as the *alignment coverage* problem. In this chapter we propose an approach to the alignment coverage problem using the three-step solution proposed above for parsing word alignments by arbitrary synchronous grammars.

Most current use of synchronous grammars is limited to a subclass using a pair of nonterminals, e.g., (Chiang, 2007; Zollmann and Venugopal, 2006; Mylonakis and Sima'an, 2011), thereby remaining within the confines of the ITG formalism (Wu, 1997). On the one hand, this is because of computational complexity reasons. On the other hand, this choice relies on existing empirical evidence of what we will call the "ITG hypothesis", freely rephrased as follows: the ITG formalism is sufficient for representing a major percentage of reorderings in translation data in general.

Checking whether a word alignment can be generated by ITG is far simpler than checking the ability to generate the alignment for arbitrary synchronous grammars. Nevertheless, there is a striking variation in the approaches taken in relation to measuring the coverage of word alignments by a grammar in the existing literature, e.g., (Zens and Ney, 2003; Wellington et al., 2006; Søgaard and Wu, 2009; Wu, 2007; Søgaard and Kuhn, 2009a; Søgaard, 2010). Søgaard and Wu (Søgaard and Wu, 2009) observe justifiably that the literature studying the ITG alignment coverage makes conflicting choices in method and data, and reports significantly diverging alignment coverage scores. We hypothesize here that the major conflicting choices in method (what to count and how to parse) are likely due to the absence of a well-understood, formalized method for parsing word alignments even under ITG. In this paper we apply our formal approach to the ITG case, contributing new empirical evidence concerning

the ITG hypothesis.

For our empirical study we exemplify our approach by detailing an algorithm dedicated to ITG in Normal-Form (NF-ITG). While our algorithm is in essence equivalent to existing algorithms for checking binarizability of permutations, e.g.,(Wu, 1997; Huang et al., 2009), the formal foundations preceding it concern nailing down the choices made in parsing arbitrary word alignments, as opposed to (bijective) permutations. The formalization is our way to resolve some of the major points of differences in existing literature.

We report new coverage results for ITG parsing of manual as well as automatic alignments, showing the contrast between the two kinds. While the latter seems built for phrase extraction, trading-off precision for recall, the former is heavily marked with idiomatic expressions. Our coverage results make explicit a relevant dilemma. To hierarchically parse the current automatic word alignments *exactly*, we will need more general synchronous reordering mechanisms than ITG, with increased risk of exponential parsing algorithms (Wu, 1997; Satta and Peserico, 2005). But if we abandon these word alignments, we will face the exponential problem of learning reordering arbitrary permutations, cf. (Tromble and Eisner, 2009). Our results also exhibit the importance of explicitly defining the units of translation equivalence when studying (ITG) coverage of word alignments. The more complex the choice of translation equivalence relations, the more difficult it is to parse the word alignments.

6.3 Grammatical translation equivalence

The derivations of a synchronous grammar can be interpreted as deriving a partially ordered set of TEUs as well. A finite derivation $S \rightarrow^{+} \langle \mathbf{f}, \mathbf{e}, \mathbf{a}_G \rangle$ of an instance grammar G is a finite sequence of term-rewritings, where at each step of the sequence a single nonterminal is rewritten using a synchronous production of G. The set of the finite derivations of G defines a language, a set of triples $\langle \mathbf{f}, \mathbf{e}, \mathbf{a}_G \rangle$ consisting of a source string of terminals $\mathbf{f}$, a target string of terminals $\mathbf{e}$ and an alignment between their grammatical constituents. Crucially, the alignment $\mathbf{a}_G$ is obtained by *recursively interpreting* the alignment relations embedded in the synchronous grammar productions in the derivation for all constituents and concerns constituent alignments (as opposed to word alignments).

Grammatical translation equivalence units $\mathbf{TE}_G(\mathbf{f}, \mathbf{e})$ A synchronous derivation $S \rightarrow^{+} \langle \mathbf{f}, \mathbf{e}, \mathbf{a}_G \rangle$ can be viewed as a deductive proof that $\langle \mathbf{f}, \mathbf{e}, \mathbf{a}_G \rangle$ is a *grammatical* translation equivalence unit (grammatical TEU). Along the way, a derivation also proves other *constituent-level* (sub-sentential) units as TEUs.

We define a *sub-sentential* grammatical TEU of $\langle \mathbf{f}, \mathbf{e}, \mathbf{a}_G \rangle$ to consist of a triple $\langle \mathbf{f}_x, \mathbf{e}_x, \mathbf{a}_x \rangle$, where $\mathbf{f}_x$ and $\mathbf{e}_x$ are two *subsequences*[2] (of $\mathbf{f}$ and $\mathbf{e}$ respectively), derived

[2]A subsequence of a string is a subset of the word-position pairs that preserves the order but do not

synchronously from the same constituent X in some non-empty "tail" of a derivation $S \rightarrow^{+} \langle \mathbf{f}, \mathbf{e}, \mathbf{a}_G \rangle$; importantly, by the workings of G, the alignment $\mathbf{a}_x \subseteq \mathbf{a}_G$ fulfills the requirement that a word in $\mathbf{f}_x$ or in $\mathbf{e}_x$ is linked to another by $\mathbf{a}_G$ iff it is also linked that way by $\mathbf{a}_x$ (i.e., no alignments start out from terminals in $\mathbf{f}_x$ or $\mathbf{e}_x$ and link to terminals outside them). We will denote with $\mathbf{TE}_G(\mathbf{f}, \mathbf{e})$ the *set of all grammatical* TEUs for the sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ derived by G.

Subsumption relation $<_{G(\mathbf{f},\mathbf{e})}$ Besides deriving TEUs, a derivation also shows *how* the different TEUs *compose* together into larger TEUs according to the grammar. We are interested in the *subsumption relation*: one grammatical TEU/constituent (u_1) subsumes another (u_2) (written $u_2 <_{G(\mathbf{f},\mathbf{e})} u_1$) iff the latter ($u_2$) is derived within a finite derivation of the former (u_1).[3]

The set of grammatical TEUs for a finite set of derivations for a given sentence pair is the union of the sets defined for the individual derivations. Similarly, the relation between TEUs for a set of derivations is defined as the union of the individual relations.

6.4 Alignment coverage by intersection

Let a word-aligned sentence pair $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$ be given, and let us assume that we have a definition of an ordered set $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ with partial order $<_{\mathbf{a}}$. Note that these two concepts were introduced before in chapter 4, section 2.3.2, as the basis of the notion of hierarchical translation equivalence which is fully and compactly represented by HATs.

We will say that a *grammar formalism covers* $\mathbf{a}$ iff there exists an instance grammar G that fulfills two intersection equations simultaneously:[4]

$$\textbf{(1)} \quad \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a}) \cap \mathbf{TE}_G(\mathbf{f}, \mathbf{e}) = \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$$

$$\textbf{(2)} \quad <_{\mathbf{a}} \cap <_{G(\mathbf{f},\mathbf{e})} = <_{\mathbf{a}}$$

In the second equation, the intersection of partial orders is based on the standard view that these are in essence also sets of ordered pairs. In practice, it is sufficient to implement an algorithm that shows that G *derives every* TEU in $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$, and that the subsumption relation $<_{\mathbf{a}}$ between TEUs in $\mathbf{a}$ must be realized by the derivations of G that derive $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$. In effect, this way every TEU that subsumes other TEUs must be derived recursively, while the minimal, atomic units (not subsuming any others) must

necessarily constitute contiguous substrings.

[3] Note that we define this relation exhaustively thereby defining the set of paths in synchronous trees derived by the grammar for $\langle \mathbf{f}, \mathbf{e} \rangle$. Hence, the subsumption relation can be seen to define a forest of synchronous trees.

[4] In the description of our formal framework we have restricted this definition to full coverage. But other similar measures can be based on the cardinality (size) of the intersection in terms of covered TEUs, in following of measures found in (Søgaard and Kuhn, 2009a; Søgaard and Wu, 2009). Some variants of these types of measures are introduced and applied later in this chapter, in section 6.9.

be derived using the lexical productions (endowed with internal word alignments) of NF-ITG. Again, the rationale behind this choice is that the atomic units constitute fixed translation expressions (idiomatic TEUs) which cannot be composed from other TEUs, and hence belong in the lexicon. We will exhibit coverage algorithms for doing so for NF-ITG for the two kinds of semantic interpretations of word alignments.

A note on dedicated instances of NF-ITG Given a translation equivalence definition over word alignments $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$, the lexical productions for a *dedicated* instance of NF-ITG are defined[5] by the set $\{X \rightarrow u \mid u \in \mathbf{TE}_{\mathbf{Atom}}(\mathbf{f}, \mathbf{e}, \mathbf{a})\}$. This means that the lexical productions have atomic TEUs at the righthand side including alignments between the words of the source and target terminals. In the sequel, we will only talk about dedicated instances of NF-ITG and hence we will not explicitly repeat this every time.

Given two grammatical TEUs u_1 and u_2, an NF-ITG instance allows their concatenation either in monotone [] or inverted <> order iff they are adjacent on the source and target sides. This fact implies that for every composed translation equivalence unit $u \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ we can check whether it is derivable by a dedicated NF-ITG instance by checking whether it recursively decomposes into adjacent pairs of TEUs down to the atomic TEUs level. Note that by doing so, we are also implicitly checking whether the subsumption order between the TEUs in $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ is realized by the grammatical derivation (i.e, $<_{G(\mathbf{f},\mathbf{e})} \subseteq <_{\mathbf{a}}$). Formally, an aligned sentence pair $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$ is split into a pair of TEUs $\langle \mathbf{f}_1, \mathbf{e}_1, \mathbf{a}_1 \rangle$ and $\langle \mathbf{f}_2, \mathbf{e}_2, \mathbf{a}_2 \rangle$ that can be composed back using the [] and <> productions. If such a split exists, the splitting is conducted recursively for each of $\langle \mathbf{f}_1, \mathbf{e}_1, \mathbf{a}_1 \rangle$ and $\langle \mathbf{f}_2, \mathbf{e}_2, \mathbf{a}_2 \rangle$ until both are atomic TEUs in $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$. This recursive splitting is the check of *binarizability* and an algorithm is described in (Huang et al., 2009).

6.5 A simple algorithm for ITG

We exemplify the grammatical coverage for (normal form) ITG by employing a standard tabular algorithm based on CYK (Younger, 1967). The algorithm works in two phases creating a chart containing TEUs with associated inferences. In the initialization phase (Algorithm 8), for all source spans that correspond to TEUs and which have no smaller TEUs they contain, *atomic* TEUs are added as atomic inferences to the chart. In the second phase, based on the atomic inferences, the simple rules of NF-ITG are applied to add inferences for increasingly larger chart entries. An inference is added (Algorithms 9 and 10) iff a chart entry can be split into two sub-entries for which inferences already exist, and furthermore the union of the sets of target

[5] Unaligned words add one wrinkle in this scheme: informally, we consider a TEU u formed by attaching unaligned words to an atomic TEU also as atomic iff u is absolutely needed to cover the aligned sentence pair.

positions for those two entries form a consecutive range.[6] The *addMonotoneInference* and *addInvertedInference* in Algorithm 10 mark the composite inferences by monotone and inverted productions respectively.

InitializeChart
Input : $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$
Output: Initialized chart for atomic units

for $spanLength \leftarrow 2$ **to** n **do**
  for $i \leftarrow 0$ **to** $n - spanLength + 1$ **do**
    $j \leftarrow i + spanLength - 1$
    $\mathbf{u} \leftarrow \{\langle X, Y \rangle : X \in \{i...j\}\}$
    if $(\mathbf{u} \in \mathbf{TE_{Atom}}(\mathbf{f}, \mathbf{e}, \mathbf{a}))$ **then**
      $addAtomicInference(chart[i][j], \mathbf{u})$
    end
  end
end

Algorithm 8: Algorithm that initializes the Chart with atomic sub-sentential TEUs. In order to be atomic, a TEU may not contain smaller TEUs that consist of a proper subset of the alignments (and associated words) of the TEU.

6.6 Experiments

Data Sets We use manually and automatically aligned corpora. Manually aligned corpora come from two datasets. The first (Graca et al., 2008) consists of six language pairs: Portuguese–English, Portuguese–French, Portuguese–Spanish, English–Spanish, English–French and French–Spanish. These datasets contain 100 sentence pairs each and distinguish *Sure* and *Possible* alignments. Following (Søgaard and Kuhn, 2009a), we treat these two equally. The second manually aligned dataset (Padó and Lapata, 2006) contains 987 sentence pairs from the English–German part of Europarl annotated using the Blinker guidelines (Melamed, 1998). The automatically aligned data comes from Europarl (Koehn, 2005) in three language pairs (English–Dutch, English–French and English–German). The corpora are automatically aligned using GIZA++ (Och and Ney, 2003) in combination with the grow-diag-final-and heuristic. With sentence length cutoff 40 on both sides these contain respectively 945k, 949k and 995k sentence pairs.

[6] We are not treating unaligned words formally here. For unaligned source and target words, we have to generate the different inferences corresponding to different groupings with their neighboring aligned words. Using pre-processing we set aside the unaligned words, then parse the remaining word alignment fully. After parsing, by post-processing, we introduce in the parse table atomic TEUs that include the unaligned words.

ComputeTEUsNFITG
Input : $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$
Output: TRUE/FALSE for coverage

InitializeChart(chart)
for $spanLength \leftarrow 2$ **to** n **do**
  for $i \leftarrow 0$ **to** $n - spanLength$ **do**
    $j \leftarrow i + spanLength - 1$
    if $chart[i][j] \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ **then**
      continue
    end
    for $splitPoint \leftarrow i + 1$ **to** j **do**
      $\mathbf{a}' \leftarrow (chart[i][k-1] \cup chart[k][j])$ **if** $(chart[i][k-1] \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a}))$
      $\wedge\ (chart[k][j] \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})) \wedge (\mathbf{a}' \in \mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a}))$ **then**
        $addTEU(chart, i, j, k, \mathbf{a}')$
      end
    end
  end
end
if $(chart[0][n-1] \neq \emptyset)$ **then**
  return TRUE
else
  return FALSE
end

Algorithm 9: Algorithm that incrementally builds composite TEUs using only the rules allowed by NF-ITG

addTEU
Input :
chart - the chart
i,j,k - the lower, upper and split point indices
$\mathbf{a}'$ - the TEU to be added
Output: chart with TEU $\mathbf{a}'$ added in the intended entry

if $Max_{Y_t}(\{Y_t : \langle X_s, Y_t \rangle \in chart[i][k-1]\}) < Max_{Y_t}(\{Y_t : \langle X_s, Y_t \rangle \in chart[k][j]\})$ **then**
  $addMonotoneInference(chart[i][j], \mathbf{a}')$
else
  $addInvertedInference(chart[i][j], \mathbf{a}')$
end

Algorithm 10: Algorithm that adds a TEU and associated Inference to the chart

Grammatical Coverage (GC) is defined as the percentage word alignments (sentence pairs) in a parallel corpus that can be covered by an instance of the grammar (NF-ITG) (cf. Section 6.4). Clearly, GC depends on the chosen semantic interpretation of word alignments: contiguous TEUs (phrase pairs) or discontiguous TEUs. Note that a formal definition of contiguous TEUs, is given in chapter 4, definition 2.3.4. Also, note that here "discontiguous TEUs" means the full set of TEUs with no restrictions with respect to contiguity, as opposed to the other interpretation of requiring all TEUs to be discontiguous, i.e. both contiguous and discontiguous TEUs are allowed.

Alignments Set	GC contiguous TEUs	GC discontiguous TEUs
Hand aligned corpora		
English–French	76.0	75.0
English–Portuguese	78.0	78.0
English–Spanish	83.0	83.0
Portuguese–French	78.0	74.0
Portuguese–Spanish	91.0	91.0
Spanish–French	79.0	74.0
LREC Corpora Average	80.83±5.49	79.17±6.74
English–German	45.427	45.325
Automatically aligned Corpora		
English–Dutch	45.533	43.57
English–French	52.84	49.95
English–German	45.59	43.72
Automatically aligned corpora average	47.99±4.20	45.75±3.64

Table 6.1: The grammatical coverage (GC) of NF-ITG for different corpora dependent on the interpretation of word alignments: contiguous Translation Equivalence or discontiguous Translation Equivalence

Results Table 6.1 shows the Grammatical Coverage (GC) of NF-ITG for the different corpora dependent on the two alternative definitions of *translation equivalence*. The first thing to notice is that there is just a small difference between the Grammatical Coverage scores for these two definitions. The difference is in the order of a few percentage points, the largest difference is seen for Portuguese–French (79% v.s 74% Grammatical Coverage), for some language pairs there is no difference. For the automatically aligned corpora the absolute difference is on average about 2%. We attribute this to the fact that there are only very few discontiguous TEUs that can be covered by NF-ITG in this data.

The second thing to notice is that the scores are much higher for the corpora from the LREC dataset than they are for the manually aligned English–German corpus. The approximately double source and target length of the manually aligned English–

German corpus, in combination with somewhat less dense alignments makes this corpus much harder than the LREC corpora. Intuitively, one would expect that more alignment links make alignments more complicated. This turns out to not always be the case. Further inspection of the LREC alignments also shows that these alignments often consist of parts that are *completely linked*. Such completely linked parts are by definition treated as atomic TEUs, which could make the alignments look simpler. This contrasts with the situation in the manually aligned English–German corpus where on average less alignment links exist per word. Examples 2.5 and 2.6 show that dense alignments can be simpler than less dense ones. This is because sometimes the density implies idiomatic TEUs which leads to rather flat lexical productions. We think that idiomatic TEUs reasonably belong in the lexicon.

When we look at the results for the automatically aligned corpora at the lowest rows in the table, we see that these are comparable to the results for the manually aligned English–German corpus (and much lower than the results for the LREC corpora). This could be explained by the fact that the manually aligned English–German is not only Europarl data, but possibly also because the manual alignments themselves were obtained by initialization with the GIZA++ alignments. In any case, the manually and automatically acquired alignments for this data are not too different from the perspective of NF-ITG. Further differences might exist if we would employ another class of grammars, e.g., full SCFGs.

One the one hand, we find that manual alignments are well but not fully covered by NF-ITG. On the other hand, the automatic alignments are not covered well by NF-ITG. A reason why these automatic alignments are difficult to cover by NF-ITG could be that these alignments are built heuristically by trading precision for recall cf. (Och and Ney, 2003). Sogaard (Søgaard, 2010) reports that full ITG provides a few percentage points gains over NF-ITG.

Overall, we find that our results for the LREC data are far higher than Sogaard's (Søgaard, 2010) results but lower than the upper bounds of (Søgaard and Wu, 2009). A similar observation holds for the English–German manually aligned Europarl data, albeit the maximum length (15) used in (Søgaard and Wu, 2009; Søgaard, 2010) is different from ours (40). We attribute the difference between our results and Sogaard's approach to our choice to adopt lexical productions of NF-ITG that contain own internal alignments (the detailed version) and determined by the atomic TEUs of the word alignment. Our results differ substantially from (Søgaard and Wu, 2009) who report upper bounds (indeed our results still fall within these upper bounds for the LREC data).

6.7 Related Work

The array of work described in (Zens and Ney, 2003; Wellington et al., 2006; Søgaard and Wu, 2009; Søgaard and Kuhn, 2009a; Søgaard, 2010) concentrates on methods for calculating *upper bounds* on the alignment coverage for all ITGs, including NF-

ITG. Interestingly, these upper bounds are determined by *filtering/excluding complex alignment phenomena* known formally to be beyond (NF-)ITG. None of these earlier efforts discussed explicitly the dilemmas of instantiating a grammar formalism or how to formally parse word alignments.

The work in (Zens and Ney, 2003; Søgaard and Wu, 2009), defining and counting TEUs, provides a far tighter upper bound than (Wellington et al., 2006), who use the disjunctive interpretation of word alignments, interpreting multiple alignment links of the same word as alternatives. We adopt the conjunctive interpretation of word alignments like a majority of work in MT, e.g., (Ayan and Dorr, 2006; Fox, 2002; Søgaard and Wu, 2009; Søgaard, 2010).

In deviation from earlier work, the work in (Søgaard and Kuhn, 2009a; Søgaard and Wu, 2009; Søgaard, 2010) discusses TEUs defined over word alignments explicitly, and defines evaluation metrics based on TEUs. In particular, Sogaard (Søgaard, 2010) writes that he employs "a more aggressive search" for TEUs than earlier work, thereby leading to far tighter upper bounds on hand aligned data. Our results seem to back this claim but, unfortunately, we could not pin down the formal details of his procedure.

More remotely related, the work described in (Huang et al., 2009) presents a binarization algorithm for productions of an SCFG instance (as opposed to formalism). Although somewhat related, this is different from checking whether there exists an NF-ITG instance (which has to be determined) that covers a word alignment.

In contrast with earlier work, we present the alignment coverage problem as an intersection of two partially ordered sets (graphs). The partial order over TEUs as well as the formal definition of parsing as intersection in this work are novel elements, making explicit the view of word alignments as automata generating partially order sets.

6.8 HATs: Exact reasoning about alignment complexity without explicit intersection

In this section we explain how Hierarchical Alignment Trees (HATs) provide an efficient way to reason exactly about the complexity and other properties of grammar formalisms that are required to cover word alignment. This is done without the need to implement the costly explicit intersection based approach discussed before.

The theoretical framework of determining coverage of word alignments by intersection with a specific grammar is theoretically appealing. Practically however it causes problems when implemented naively. Certainly theoretically we could make a program that has a main loop generating all possible grammars, and then for each of this grammars parses the sentence pair and performs the intersection with the set of TEUs. However, since the set of possible grammars is infinite, it follows that this approach will not work in practice. Fortunately we will see that taking inspiration from the analogous example of treebank grammars (Charniak and Charniak, 1996)

for monolingual parsing, we do not need to separately create grammars and then test their compatibility with the word alignments. Instead we can directly read out a set of compatible and minimal required structures from the set of hierarchical TEUs as compactly described by the HATs for word alignments. We will now clarify why this is a valid approach.

As discussed in chapter 4, HATs are by definition built to exactly capture the full set of contiguous TEUs induced by a an alignment triple (a triple of source, target and word alignment), as well as their full set of subsumption relations.[7] HATs are constructed in such a way as to always provide maximal decompositions of word alignments, which means for every n, an *n-ary* branching node will only be introduced in a HAT when some part of the alignment strictly requires a node of that branching factor to decompose it and (recursive) decomposition using nodes with smaller branching factors is not possible. This implies that for any n if a HAT has an *n-ary* branching node, it follows that in order to produce all the TEUs $\mathbf{TE}(\mathbf{f}, \mathbf{e}, \mathbf{a})$ and subsumption relations $<_{\mathbf{a}}$ induced by the alignment, and hence *cover the alignment* as defined in section 6.4, at least an n-ary branching grammar is required. Thus the HATs of word alignments allow us to directly predict the (minimal) branching factor that grammars require to cover the alignment, and the explicit intersection of sets of TEUs and subsumption relations induced by the alignment and producible by the grammar is not necessary.

This simple insight, shows that by parsing alignments and producing their induced HATs, using the algorithms described in chapter 4, section 4.7, it is possible to efficiently produce alignment coverage statistics for various grammar formalisms under the *alignment coverage by intersection* formalism of section 6.4. This is done without requiring the explicit intersection to be computed, which would be highly expensive if not intractable. In section 6.5, we essentially took the same approach, but limited ourselves beforehand to the simpler case of normal form ITG (NF-ITG), effectively parsing with a simplified HAT parsing algorithm that can produce only binary-branching HATs (BITTs). Here, by using a general HAT parsing algorithm in an analogous way, we generalize the approach of section 6.5, producing all TEUs and subsumption relations induced by word alignments under maximal decompositions and based on those determining the (minimal) branching factor and other properties required of grammar formalisms in order to cover the alignments.

6.8.1 Restriction to contiguous translation equivalence units

An important remark, requiring some explanation here, is that HAT-based analysis is based on the assumption of working with contiguous TEUs (phrase pairs). Previously, in our experiments for NF-ITG we wanted to show the influence of the choice the

[7] In fact, HATs are constructed in such a way to even capture the full set of *mapping relations*, a stronger concept than subsumption relations, but for this discussion set of subsumption relations, captured by implication, suffices.

type of TEUs used on the results. In what follows we will study the properties of the grammar formalisms required to cover word alignments. In this, we restrict ourselves to contiguous TEUs and do not allow discontiguous TEUs to exist independently, while discontiguous constructions are still allowed embedded inside TEUs. The motivation for this choice is that if we do not make this assumption, we end up with the much more complex class of grammatical systems called *synchronous linear context-free rewriting systems* (synchronous LCFRS), for which relevant analysis has been performed by Kaeshammer (2013). Kaeshammer (2015) has recently applied synchronous LCFRS to build an actual hierarchical SMT system, proving that using them for translation is already feasible. But a disadvantage of working with LCFRS we think is that they provide very little inductive bias when deducing TEUs from word alignments.[8]

Our main concern is to have a representation that allows learning a useful notion of hierarchical translation equivalence, particularly in such a way that the TEUs are properly compositional and the coherence of translations with respect to observed composition in training data, and associated word order, can be reinforced. That this is possible with HATs, making the assumption of restriction to contiguous TEUs[9] is shown in detail in chapter 5, where based on HATs soft bilingual reordering constraints are formulated, that effectively encourage coherence of word order and significantly improve the translations.

6.8.2 What are empirical word alignments?

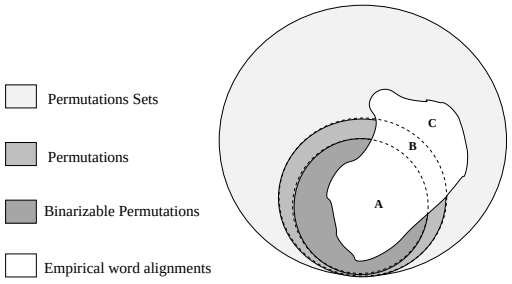


Figure 6.1: A sketch of a possible characterization of empirical word alignments

In the next section we will employ the generalized approach based on HATs (and restricted to contiguous TEUs), to efficiently produce an extensive range of exact

[8]The problem of finding the segmentation of a word alignment is a notoriously difficult problem, and the reason why most existing approaches to both phrase-based and hierarchical phrase-based translation resort to heuristic methods, with some exceptions (DeNero et al., 2006; Blunsom et al., 2009; Cohn and Blunsom, 2009; Mylonakis and Sima'an, 2010, 2011). A stronger grammar formalism only worsens this problem, by further reducing the amount of inductive bias provided by the grammar formalism.

[9]While we do allow hierarchical rules which have gaps/variables, these gaps need to be (recursively) filled by contiguous TEUs. This is why these lexicalized SCFG rules are considered (abstractions over) contiguous TEUs rather than discontiguous TEUs, the latter allowing discontiguity even at the atomic (terminal rule) level, which makes them no longer expressible by SCFGs but only by stronger formalisms such as Synchronous LCFRS.

alignment coverage and related statistics for different language pairs . But before we do this, we pause for a moment, and ask ourselves: what are empirical word alignments? Figure 6.1 shows a sketch of a likely situation: empirical word alignments are not a proper subset of binarizable permutations, nor are they a proper subset of permutations. However, by definition, word alignments are a subset of s-permutations. The figure shows that empirical word alignments overlap with binarizable permutations (area A), with permutations (area A+B) and with s-permutations (area A+B+C). What are the relative sizes of areas A, B and C? Clearly this is an empirical question that depends on the nature of the parallel corpus (language pair and language use) and on the kind of word alignments found in it. In the next section we aim to explore this question empirically on some standard data sets, with either manual alignments or automatic alignments with GIZA++ in combination with the grow-diag-final heuristic.

6.9 Empirical study of recursive reordering in word alignments

The question we want to answer in this section can be formulated as:

> What percentage of word alignments and TEUs is representable given different sets of formal constraints on the allowed subset of HATs (e.g., upper bound on branching factor, limits on the permitted complexity type of the reordering operator of the HAT nodes, bucketing of node transduction operators as a "distance measure" from pure permutations)?

We study this question on manually produced word alignments as well as automatically generated word alignments. Note that our empirical findings must be interpreted *in relation to the specific definition of sub-permutations, i.e., word-aligned phrase pairs*.

HATs are minimally branching STPs for the word alignments. Therefore a primary differentiating parameter for the performance measures is the *maximal branching factor*:

β_{max} the *maximal branching factor* of the linked nodes (above the pre-terminal level) in the HATs. The branching factor is measured on both sides and satisfying the branching factor constraint requires all linked nodes in the HATs for a word alignment to fall within the range $[1..\beta_{max}]$.[10]

For a given β_{max} value we report:

[10]Null aligned words are not counted in the branching factor, i.e, even unaligned words dominated directly by a node don't contribute to the branching factor of that node. The rationale behind this is that we do not want to discriminate between s-permutations differing only by NULL aligned words. This means that the reported results are rather conservative.

Alignment coverage This is the percentage of word alignments for which every node in any of the HATs induced by the word alignment has a branching factor of at most β_{max} Alignment coverage is not necessarily equivalent to *alignment reachability* (Søgaard and Wu, 2009; Søgaard, 2010) or the complementary measure of *parsing failure rate (PFR)* (Zens and Ney, 2003; Søgaard and Wu, 2009), which are both reported under subsets of ITG.

Alignment coverage is equivalent to alignment reachability under a given grammar formalism iff both are measured in under the same semantics with respect to the notion of TEUs and also the setting of β_{max} corresponds exactly to the formalism in question.[11] In this sense, alignment coverage for $\beta_{max} = 2$ is an exact measure (as opposed to upper/lower-bounds) of alignment reachability for NF-ITG.[12]

Translation equivalents coverage (TEC) This is the *percentage of (linked) phrase pairs* as represented by linked nodes in the HATs that have maximal branching factor β_{max}. The TEC measure is strongly related to Translation Units Error Rate (TUER) (Søgaard and Kuhn, 2009b; Søgaard and Wu, 2009). In fact, when using the same definition of TEUs, the same counting algorithm and the same representation for both (NF-ITG), we find that TUER = (1-TEC).

Binarizability score The ratio of the number of linked nodes in a HAT (subject to β_{max}) constructed for a given word alignment relative to the number of such nodes in a hypothetical, fully binary branching HAT (using the length of the shortest among the source and target sentences minus one). In contrast to TEC, the binarizability score provides a relatively objective measure of to what extent the word alignments in the corpus can be represented by deeply nested HATs. The lower the binarizability score, the less linked nodes exist in the HATs admitted (given β_{max}), if any at all. HATs that contain flat structures indicate complex word alignments involving unaligned words and many-to-many alignments that cannot be decomposed into smaller TEUs. This can be seen to imply idiomatic translation, as opposed to compositional translation (given the semantics defined here).

Because for some alignments our implementation can consume large memory in calculating the HAT forest we had to abort it in a small percentage of cases ($\leq 10^{-5}$), mostly very long alignments with many unaligned words. Later on we optimized the algorithm, taking out unaligned words during parsing, and putting them back when unpacking the generated compacted HAT forest. We limit the maximum number of inferences per linked node in the CYK chart implementing the HAT algorithm in

[11] Unlike for ITGs, the case $\beta_{max} = 3$ is not necessarily equivalent to $\beta_{max} = 2$ because the s-permutations (as opposed to permutations) over three positions can be non-binarizable, see e.g., Figure 4.13.

[12] Since our algorithm builds HATs over minimal phrase pairs, the case $\beta_{max} = 2$ is equivalent to the NF-ITG over these minimal phrase pairs, i.e., given the defined semantics of word alignments.

Algorithm 1 (chapter 4, section 4.7): once the algorithm generates more than 100k inferences for any node we skip the alignment. The number of skipped sentence pairs is reported for each experiment separately in what follows.

6.9.1 Manual word alignments

For our empirical study of recursive reordering in manual word alignments, we use the manually aligned part of the Hansards corpus (English–French), which was created and first used by Och and Ney (Och and Ney, 2000, 2003). This tiny corpus consists of 447 manually aligned sentence pairs, with alignment links labeled as *Sure* or *Possible*. Some basic statistics of this corpus are shown in Table 6.2. We report the results for the Sure+Possible links, i.e., all alignment links.[13]

Language Pair	**English–French**
Total number of sentence pairs	447
#skipped sentence pairs	0
#sentence pairs containing nulls	231
Mean source length	15.705±6.994
Mean target length	17.362±7.554
Mean and STD of ratio source to target lengths	0.928±0.221
Mean #links per word	2.52 ± 1.73

Table 6.2: Statistics of Hansards manually aligned corpus. The “mean #links per word" is calculated using the mean $\frac{\text{\# alignment links}}{\min_{x\in\{\mathbf{f},\mathbf{e}\}} length\ of\ x}$ over the corpus word alignments.

Tables 6.3a, 6.3b and 6.3c report respectively the break-down of coverage, TEC and binarizability score to β_{max}. In the range $\beta_{max} \in [2..10]$ all scores increase rapidly, after $\beta_{max} > 10$ their increase is slower. The alignment coverage and the TEC results start low in the seventies for $\beta_{max} = 2$ (NF-ITG), but quickly increase to the low nineties by $\beta_{max} = 6$. The increase continues at a lowering pace for higher values of β_{max}. The binarizability scores, Table 6.3c, show a different side of the story. It is made clear by these results that the HATs built for the word alignments contain only at most 62% of the number of linked nodes in a *hypothetical* fully binary HAT (binarizable permutation). This suggests that these word alignments induce HATs that are somewhat flat relative to the hypothetical fully binary HATs. This observation

[13] Slightly different results for these experiments were earlier reported in (Sima'an and Maillette de Buy Wenniger, 2013), the reason for the differences is that in the earlier experiment some alignments were skipped because otherwise the unaligned words lead to too many inferences, making the HAT parser fail. A later improved version of the code used in these experiments takes the unaligned words out before parsing, remembers their positions, and puts them back afterwards. This avoids unnecessary complexity issues, assuring all alignments can be parsed, and explaining the slight differences between the earlier results and the results reported here.

β_{max}	English–French
2	71.36%
3	76.96%
4	82.33%
5	87.03%
6	89.93%
7	92.62%
8	94.41%
9	96.42%
10	97.76%
15	99.33%
19	100.00%

(a) Alignment coverage.

β_{max}	English–French
2	73.42%
3	80.53%
4	87.09%
5	91.58%
6	93.65%
7	95.13%
8	96.97%
9	98.31%
10	98.90%
15	99.81%
19	100.00%

(b) Translation Equivalents coverage.

β_{max}	English–French
2	42.05%
3	46.65%
4	51.18%
5	54.62%
6	56.43%
7	57.95%
8	59.52%
9	60.75%
10	61.50%
15	62.39%
19	62.64%

(c) Binarizability scores.

Table 6.3: Scores for Sure+Possible manual alignments in the Hansards corpus as a function of β_{max}.

completes the picture drawn by the coverage and TEC results for $\beta_{max} > 2$: the word alignments are clearly far more complex than the basic cases of binarizable permutations.

Taking a different but at least as interesting look on matter, Table 6.4 reports the coverage of word alignments to the kind of s-permutation (HATs) involved: binarizable permutations (BITTs), permutations (PETs) and s-permutations (HATs). Remarkably, coverage is not much improved by merely moving away from binarizable to all permutations, whereas the general case of s-permutations provides full coverage. This result combined with the break-down of the statistics to β_{max} values in the other tables suggests that the cases of non-binarizable permutations typically co-occur with other complex forms of alignments, including discontinuous and many-to-many cases. Rather than implying that the full descriptive power of permutations is unnecessary, this means that on their own permutations are *almost as insufficient as their binarizable*

Kind of HATs (S-permutations)	English–French
BITTs (Binarizable permutations)	71.36%
PETs (All permutations)	72.04%
HATs (S-permutations)	100.00%

Table 6.4: The ratio of the different subsets of HATs in the manual Sure+Possible alignments in the Hansards corpus: BITTs, PETs and HATs

subset for capturing word alignments found in actual translation data.

6.9.2 Automatic word alignments

We report empirical results on English–Dutch, English–French and English–German corpora derived from Europarl (Koehn, 2005).[13] We also report results for a Chinese–English corpus based the combination of the *Hong Kong Parallel Text* parallel corpus and the *MultiUN* (Eisele and Chen, 2010) corpus. The Chinese side of this parallel corpus is segmented using the Stanford Segmenter (Chang et al., 2008), more details about the segmentation are given in A.6. All four corpora are derived from their input corpora by setting an upper bound of 40 words on the sentence length on the source and target sides. The sizes of the corpora are listed in Table 6.5. The corpora derived from Europarl are about half the size of the corresponding original corpora. Nonetheless we think that they are large enough to contain word alignments representative of the subsuming, full-size corpora. The English–German corpus is roughly equivalent to the corpus used in chapter 5 in the translation experiments for English–German, and in inverted direction for German–English.[14] The Chinese–English corpus used in this chapter is exactly equivalent to the one used for translation in chapter 5.

Following standard practice (e.g., (Koehn et al., 2007)), the sentences for the English, French and German sides of the corpora were lower-cased and tokenized using the relevant Moses scripts.[15] As mentioned before, the Chinese side of the Chinese–English corpus was segmented rather than tokenized. Arguably lowercasing is not the optimal strategy for all language pairs. Particularly in German all nouns are capitalized which means that some information will get certainly lost by lowercasing. However, the information that is lost is not too important for the IBM models, and thus is not expected to influence the word alignments much. The sentence pairs were word-aligned using GIZA++[16] with number of training iterations as follows: four for IBM model 1, three for IBM model 3, four for IBM model 4 and three for the HMM model.

[14]In the translation experiments the cased version of the source and target side was required for parsing by the SAMT (Zollmann and Venugopal, 2006) baseline, but this version had been lost in our version of the corpus. We therefore retrieved the cased version of the sentences from the original Europarl corpus, and removed these sentence pairs for which the cased version could not be retrieved, giving a slightly smaller corpus used for translation than the one used in this chapter.

[15]`http://www.statmt.org/moses/`

[16]`http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html`.

Language Pairs	English–Dutch	English–French	English–German	Chinese–English
Total number of sentence pairs	945167	949408	995909	7340000
#skipped sentence pairs	0	0	0	0
# sentence pairs containing nulls	802693	783624	839788	6323218
Mean source length	21.30±8.92	20.56±8.58	21.56±9.14	20.11±9.25
Mean target length	21.22±9.02	22.56±9.38	20.46±8.88	21.74±9.91
Mean and STD of ratio of source to target lengths	1.03±0.22	0.93±0.20	1.08±0.24	0.98±0.39
Mean & STD #links per word	1.12±0.14	1.15±0.15	1.14±0.16	1.22±0.35

Table 6.5: The corpora used in our analysis (sentence length ≤ 40 words). The "mean #links per word" is calculated using as the mean over the corpus alignments for the ratio $\frac{\text{\# alignment links}}{\min_{x\in\{\mathbf{f},\mathbf{e}\}} length\ of\ x}$.

The symmetrization of the alignments in the two translation directions was done using the grow-diag-final heuristic.

Tables 6.6 and 6.7 show the alignment coverage and TEUs coverage (TEC) results respectively. Compared to the manual word alignments, for the automatic alignments the alignment coverage and TEC results increase dramatically for β_{max} values in [2..6]. For $\beta_{max} = 2$ the results are in the mid forties for English–Dutch/German, low fifties for English–French and up to forty for Chinese–English, but by $\beta_{max} = 6$ all results are in the nineties or near. In fact the coverage score for the earlier mentioned manual alignments as reported in Tables 6.3a, 6.3b also have risen to scores above ninety for $\beta_{max} = 6$, but the difference is that they start already from much higher values even for $\beta_{max} = 2$. We will now discuss some properties of the manual versus automatic word alignments that can explain these differences.

Automatic alignments obtained by symmetrization heuristics (particularly grow-diag-final) are constructed in a way that allows the extraction of a large number of phrase pair equivalents. This could explain why binary branching HATs (BITTs) have lower coverage of such word alignments, and why averaged over language pairs for more than half the alignments a branching factor larger than two is needed. Table 6.8 supports this observation. On the one hand, for HATs that are at most binary branching ($\beta_{max} = 2$), the binarizability score is very low in the thirties (Dutch/German) or forties (French) suggesting that these word alignments are hard to capture with BITTs. On the other hand, by $\beta_{max} \geq 10$ the binarizability score is around the 83% for English–French suggesting HATs with many linked nodes, particularly when we contrast this with a score of approximately 62% for the Hansards manual alignment given the same constraint on β_{max}.

	English-Dutch	English-French	English-German	Chinese-English
$\beta_{max} = 2$	45.50%	52.82%	45.57%	40.00%
$\beta_{max} = 3$	55.75%	67.24%	56.76%	53.61%
$\beta_{max} = 4$	73.91%	82.57%	74.71%	72.27%
$\beta_{max} = 5$	83.47%	89.98%	84.59%	82.51%
$\beta_{max} = 6$	89.87%	94.37%	90.71%	89.19%
$\beta_{max} = 7$	93.52%	96.69%	94.31%	93.14%
$\beta_{max} = 8$	95.87%	98.03%	96.50%	95.60%
$\beta_{max} = 9$	97.33%	98.83%	97.84%	97.17%
$\beta_{max} = 10$	98.30%	99.29%	98.67%	98.18%
$\beta_{max} = 15$	99.85%	99.95%	99.89%	99.78%
$\beta_{max} = 22$	100.00%	100.00%	100.00%	100.00%

Table 6.6: Coverage of the corpus as a function of β_{max} for symmetrized (grow-diag-final) word alignments in Europarl parallel corpora (sentence length ≤ 40) for three language pairs

	English-Dutch	English-French	English-German	Chinese-English
$\beta_{max} = 2$	46.75%	53.57%	45.47%	37.21%
$\beta_{max} = 3$	58.61%	70.17%	58.30%	53.10%
$\beta_{max} = 4$	77.83%	85.84%	77.34%	74.00%
$\beta_{max} = 5$	87.15%	92.79%	87.29%	84.77%
$\beta_{max} = 6$	92.82%	96.39%	92.88%	91.25%
$\beta_{max} = 7$	95.72%	98.08%	95.93%	94.80%
$\beta_{max} = 8$	97.45%	98.95%	97.63%	96.87%
$\beta_{max} = 9$	98.45%	99.43%	98.61%	98.10%
$\beta_{max} = 10$	99.06%	99.68%	99.19%	98.84%
$\beta_{max} = 11$	99.43%	99.81%	99.52%	99.28%
$\beta_{max} = 12$	99.65%	99.89%	99.73%	99.54%
$\beta_{max} = 13$	99.80%	99.94%	99.84%	99.71%
$\beta_{max} = 14$	99.88%	99.96%	99.91%	99.82%
$\beta_{max} = 15$	99.93%	99.98%	99.94%	99.89%
$\beta_{max} = 16$	99.96%	99.99%	99.97%	99.93%
$\beta_{max} = 17$	99.98%	100.00%	99.98%	99.96%
$\beta_{max} = 18$	99.99%	100.00%	99.99%	99.97%
$\beta_{max} = 19$	99.99%	100.00%	100.00%	99.98%
$\beta_{max} = 20$	100.00%	100.00%	100.00%	99.99%
$\beta_{max} = 21$	100.00%	100.00%	100.00%	100.00%

Table 6.7: Translation Equivalents Coverage as a function of β_{max}

	English-Dutch	English-French	English-German	Chinese-English
$\beta_{max} = 2$	33.42%	41.41%	32.72%	24.90%
$\beta_{max} = 3$	42.37%	54.72%	42.42%	36.72%
$\beta_{max} = 4$	58.34%	68.89%	58.14%	53.28%
$\beta_{max} = 5$	66.43%	75.44%	66.54%	61.94%
$\beta_{max} = 6$	71.71%	79.17%	71.57%	67.38%
$\beta_{max} = 7$	74.58%	81.06%	74.42%	70.44%
$\beta_{max} = 8$	76.38%	82.12%	76.08%	72.26%
$\beta_{max} = 9$	77.48%	82.72%	77.09%	73.38%
$\beta_{max} = 10$	78.18%	83.07%	77.69%	74.05%
$\beta_{max} = 15$	79.23%	83.52%	78.53%	75.03%
$\beta_{max} = 21$	79.32%	83.55%	78.59%	75.12%

Table 6.8: Binarizability scores as a function of β_{max}. For any $\beta_{max} > 21$ the scores do not increase further.

	English-Dutch	English-French	English-German	Chinese-English
$BITTs$	45.50%	52.82%	45.57%	40.00%
$BITTs \cup PETs$	52.62%	56.55%	52.55%	46.96%
$BITTs \cup PETs \cup HATs$	100.00%	100.00%	100.00%	100.00%

Table 6.9: The ratio of the different subsets of HATs in the corpus: BITTs, PETs and HATs

It is unlikely that the latter difference can fully be explained by the difference in language use (Hansards vs Europarl), in fact he shorter average sentence length in the Hansards manually aligned corpus suggests the reverse situation should be true.

This supports the following hypothesis:

Hypothesis. *Symmetrized automatic alignments are built such a way that they can facilitate extracting a larger number of phrase pair equivalents, leading to many more nodes in the* HATs *than manual alignments.*

Table 6.9 shows that the coverage of BITTs is around 52% for French and around 45% for Dutch and German.

The coverage of PETs (permutations) increases by 4-7% only, which again suggests that neither BITTs nor PETs (as pure permutation-devices) can provide good coverage of phenomena in word alignments. If only about 50% of all such word alignments can be represented fully as a permutation, then the other 50% demands the notion of a s-permutation which can capture discontinuous alignments and complex many-to-many cases. Yet, s-permutations remain simple extensions of permutations and HATs can be considered conservative extensions of PETs and BITTs.

	$\beta_{max} = 2$	$\beta_{max} = 3$	$\beta_{max} = 6$	$\beta_{max} = 8$	$\beta_{max} = 18$
$\alpha_{max} = 1$	71.36%	76.96%	89.93%	94.41%	99.78%
$\alpha_{max} = 2$	71.36%	76.96%	89.93%	94.41%	99.78%
$\alpha_{max} = 3$	75.39%	76.96%	89.93%	94.41%	99.78%
$\alpha_{max} = 5$	81.66%	81.88%	89.93%	94.41%	99.78%
$\alpha_{max} = 10$	93.29%	93.51%	94.86%	95.97%	99.78%
$\alpha_{max} = 20$	98.66%	98.66%	98.88%	99.33%	99.78%
$\alpha_{max} = 30$	100.00%	100.00%	100.00%	100.00%	100.00%

Table 6.10: Manual word alignments (Hansards). Coverage of the corpus as a function of α_{max}, the *Maximum Source Length of Atomic Fragments* and β_{max}, the *Maximal Branching Factor*.

6.9.3 Is embedding into larger atomic units an effective solution for complex alignment patterns?

One hypothesis that is frequently posed with respect to the complexity of non-binarizable word alignments, is that the complexity of these word alignments would be mainly local. This would mean that the complexity could be effectively dealt with by embedding the complexity inside atomic phrase pairs. Here we test if this hypothesis is true. In table 6.11 we show how much coverage we can get for different values of β_{max} provided that we allow all phrase pairs with maximum source/target lengths up to a certain value α_{max} to be treated as atomic units and hence ignored in the determination of alignment coverage. Here we see, that while coverage initially quickly increases by increasing the value of α_{max}, very high values of α_{max} and/or β_{max} are required to get full or near full coverage of the word alignments. This means that just embedding complexity into big atomic units helps a bit, but is clearly not the complete solution for dealing with the complexity observed in empirical word alignments. This insight is important because other related work deals with the complexity of non-binarizable or non-bijective mappings by embedding the complexity inside large atomic units (Mylonakis and Sima'an, 2011; Stanojević and Sima'an, 2015). The results in Table 6.11 do suggest that this approach can work at least to an extend, and as such are consistent with the improvements in translation quality reported in work using this approach. But the results also suggests that still much can be gained by using formalisms that really support non-bijective mappings, instead of trying to avoid them by incorporation into large, poorly generalizing, atomic units. For the English-French manual word alignments of Hansards, see Table 6.10, embedding into larger atomic fragments to increase coverage yields similar results as for the same language pair with automatic word alignments. For Chinese–English, the percentage of word alignments that can be covered by BITTs is the lowest across the four automatically aligned language pairs, and this is also reflected in the lowest coverage results in Table 6.11 across all four language pairs.

	$\beta_{max} = 2$	$\beta_{max} = 3$	$\beta_{max} = 6$	$\beta_{max} = 8$	$\beta_{max} = 22$
English-Dutch					
$\alpha_{max} = 1$	45.50%	55.75%	89.87%	95.87%	100.00%
$\alpha_{max} = 2$	45.50%	55.75%	89.87%	95.87%	100.00%
$\alpha_{max} = 3$	52.41%	55.75%	89.87%	95.87%	100.00%
$\alpha_{max} = 5$	65.97%	67.16%	89.87%	95.87%	100.00%
$\alpha_{max} = 10$	84.02%	84.20%	92.74%	96.21%	100.00%
$\alpha_{max} = 20$	96.48%	96.49%	97.82%	98.53%	100.00%
$\alpha_{max} = 40$	100.00%	100.00%	100.00%	100.00%	100.00%
English-French					
$\alpha_{max} = 1$	52.82%	67.24%	94.37%	98.03%	100.00%
$\alpha_{max} = 2$	52.82%	67.24%	94.37%	98.03%	100.00%
$\alpha_{max} = 3$	64.00%	67.24%	94.37%	98.03%	100.00%
$\alpha_{max} = 5$	79.06%	79.61%	94.37%	98.03%	100.00%
$\alpha_{max} = 10$	91.21%	91.26%	96.48%	98.30%	100.00%
$\alpha_{max} = 20$	98.01%	98.02%	98.96%	99.36%	100.00%
$\alpha_{max} = 40$	100.00%	100.00%	100.00%	100.00%	100.00%
English-German					
$\alpha_{max} = 1$	45.57%	56.76%	90.71%	96.50%	100.00%
$\alpha_{max} = 2$	45.57%	56.76%	90.71%	96.50%	100.00%
$\alpha_{max} = 3$	52.57%	56.76%	90.71%	96.50%	100.00%
$\alpha_{max} = 5$	65.94%	67.83%	90.71%	96.50%	100.00%
$\alpha_{max} = 10$	84.04%	84.48%	93.61%	96.86%	100.00%
$\alpha_{max} = 20$	96.73%	96.77%	98.16%	98.83%	100.00%
$\alpha_{max} = 40$	100.00%	100.00%	100.00%	100.00%	100.00%
Chinese-English					
$\alpha_{max} = 1$	40.00%	53.61%	89.19%	95.60%	100.00%
$\alpha_{max} = 2$	40.00%	53.61%	89.19%	95.60%	100.00%
$\alpha_{max} = 3$	48.24%	53.61%	89.19%	95.60%	100.00%
$\alpha_{max} = 5$	61.70%	63.61%	89.19%	95.60%	100.00%
$\alpha_{max} = 10$	80.05%	80.42%	92.15%	96.05%	100.00%
$\alpha_{max} = 20$	95.27%	95.30%	97.52%	98.46%	100.00%
$\alpha_{max} = 40$	100.00%	100.00%	100.00%	100.00%	100.00%

Table 6.11: Automatic word alignments. Coverage of the corpus as a function of α_{max}, the *Maximum Source Length of Atomic Fragments* and β_{max}, the *Maximal Branching Factor*. Scores are given for four different language pairs.

6.9.4 Discussion and related empirical work on the analysis of alignments

There is a lot of debate concerning the representation power of (Normal-Form) ITG for translation data in the translation community. The empirical results presented in the preceding section relate to this debate. Existing reports have mostly concentrated on *upper bounds* for the representation power of ITG in terms of its ability to capture manual or automatic word alignments (Zens and Ney, 2003; Galley et al., 2004; Wellington et al., 2006; Søgaard and Wu, 2009; Søgaard, 2010). A small part of our empirical results can be seen to contribute to this debate, particularly those concerning BITTs. Søgaard and Wu (Søgaard and Wu, 2009) observe that the reports in the literature differ considerably along various dimensions which makes comparison of compare coverage results difficult.

As they note, approaches differ in at least four dimensions (i) Data: what data and which alignments are used, (ii) Metrics: the way the coverage is measured (sentence vs. translation unit levels), (iii) Semantics: how to interpret word alignments (disjunctively/conjunctively) and (iv) Algorithmics: the algorithm used for computing the upper bounds. Indeed, in studying the existing literature we found it particularly hard to pin down the exact choices made along these dimensions, which makes it difficult to interpret the reported results. There is, however, a good reason for the difficulty of exact description as we will explain next.

Conceptually, determining the ability of a grammar formalism to capture word alignments entails an intersection of the spaces of concepts that is representable by each of these formal systems. However, because these systems are of different types there is no a priori, objective, formal grounds on which word alignments and synchronous grammars can be formally intersected (or composed). Before intersecting these two completely different representations, it is necessary to specify a *shared meaning/semantics*. We think this is the main reason why measuring the exact coverage of word alignments by a synchronous grammar is so complicated. The solution we use in this work is to define for every word alignment a semantically equivalent set of HATs, and then to check that these HATs can be built by an instance of the grammar formalism, i.e., that there exists an instance at all that can generate these HATs.

The earlier defined sentence-level coverage is then measured by checking that there exists an instance of the grammar formalism that can generate all these HATs exactly. And similarly coverage/recall of TEUs/units (TEC) is computed by determining the percentage of linked nodes that can be generated by an instance of the grammar formalism. Our results with $\beta_{max} = 2$ are in fact coverage and TEC results for NF-ITG. Some earlier work has concentrated on measuring *upper bounds* on the coverage/TEC either by deducting the complex alignment cases that cannot be covered by NF-ITG, e.g., (Søgaard and Kuhn, 2009b; Søgaard and Wu, 2009), or by defining a “lighter" semantics of word alignments by using a disjunctive interpretation (as opposed to the more accepted conjunctive interpretation) (Wellington et al., 2006).

As far as we can see, the results presented in (Zens and Ney, 2003; Wu et al., 2006;

Søgaard, 2010), although based on a different approach and computed for different data sets and word alignments, are measured in ways that are close to implementing the intersection used here. Zens and Ney (Zens and Ney, 2003) use Viterbi alignments on Hansards data (sentence length up to 30 words) and obtain much higher coverage results for NF-ITG ($\approx$ 81% and 73% depending on direction) than our results for English–French Europarl data with symmetrized word alignments ($\approx$ 53%). Besides the differences in corpus data, symmetrized alignments, which are the basis for training state-of-the-art systems, are known to be distinctively different from their Viterbi uni-directional ancestors. The coverage result of (Søgaard, 2010) on the manual Hansards data (77%) and our coverage result (72%) lie very close together. Søgaard (Søgaard, 2010) is presented densely and somewhat informally, so that we miss certain details. We attribute the difference to various reasons, including sentence-length differences (in (Søgaard, 2010) the cutoff is 15 words) and choices concerning how to define TEUs with unaligned words on either side. We also mention the work of (Wu et al., 2006), but do not cover it further, as it concerns Arabic–English data, which is not studied here. Next, looking at a related yet quite distinct work, (Huang et al., 2009) reports measures of word alignment coverage under ITG constraints in combination with syntactic constraints. The work is based on the GHKM (Galley et al., 2004) method of extracting synchronous rules, which involves target-language syntax. The authors report statistics over what fraction of the extracted synchronous rules is binarizable. The results reported are incomparable to our results for NF-ITG because they are subject to the GHKM extraction method of synchronous rules. This means syntactic constraints are used as an argument to encapsulate very difficult word alignments as internal, lexical parts of a synchronous rule. By doing so, the coverage is measured with regards to a different semantics (the GHKM extraction method) of word alignments, which constitutes a very different (syntax constrained, not just alignment constrained) set of permissible TEUs than our choice of semantics. Our word alignments semantics is more exhaustive than the GHKM semantics in that we allow all phrase pairs to be extracted without constraints from monolingual syntax or other performance-driven limitations.

6.10 Stronger formalisms and other important trends in SMT

In this chapter we have mainly restricted ourselves to working with contiguous TEUs and explained why we believe this is appropriate. Before we end this chapter, in what follows we want to give an outlook on what it might mean to go beyond this restriction, and how this relates to other big trends in the field of machine translation and AI in general.

We think going beyond SCFGs can be very interesting in itself, but to be successful certain things have to be taken care of:

- We have to make the stronger formalisms fast enough to support real translation.
- We must assure sufficient inductive bias remains to eventually learn a form of compositional hierarchical translation equivalence.
- We must apply the power of the stronger formalism only where necessary, and stay within the extended upon weaker formalisms whenever possible.

Note how the third principle basically concerns the importance of adequate smoothing and backoff. This principle has shown its importance over and over in the history of the field of Computational Linguistics, in translation as much as parsing, language modelling and other applications.

It is likely that with increasing computation power, stronger formalisms such as Synchronous LCFRS might in the future play a role in improving the state of the art of statistical machine translation, similar to how approaches based on synchronous context free grammars have slowly overtaken, at least for certain language pairs, the simpler finite state machines that are sufficient for phrase-based translation.

But perhaps more important still than stronger formalisms, is the need to use stronger machine learning techniques that make better use of the data, and better model the compositional structure of translation as part of its parameters, instead of relying heavily on redundancy and excessive memorization. The recent popularity of neural networks and deep learning approaches highlights a renewed appreciation of the importance of machine learning in the field of machine translation. But note that neural networks or deep learning methods are not the only way to tackle stronger machine learning. More in spirit with the original work on the IBM models, that are foundational to contemporary machine translation, a publication (Ittycheriah and Roukos, 2007) from the time before neural networks became so popular in machine translation shows how based on discriminative learning with *Maximum Entropy Models* machine translation can be improved. This paper proposes a "minimalist" approach called "Direct Translation Model 2", working with a non-overlapping inventory of translation units and employing effective learning of many parameters to enforce structure, and achieve improvements over the state-of-the-art for Arabic–English (phrase-based) translation.

What many methods involving neural networks and stronger machine learning have in common, is that in learning (soft) constraints that transcend the rule level, they effectively decrease the power of the dependency assumptions and add more context. Our own approach in this matter, explained in detail in chapter 4, follows largely the same approach. But beyond coherent translation and sufficient context to make adequate translation decisions lies a richer goal, further on the horizon. This goal is to go beyond mere translation and learn the supposedly (largely) language independent meaning equivalence underlying translation equivalence.

6.10.1 On the value of learning interpretable meaning representations

While neural network based and deep learning based methods promise to create a hidden representation capturing a notion of this meaning automatically, their inherent black-box optimization design makes it often hard to interpret or directly use what is learned. In contrast, recent work by Titov and Klementiev (2012) proposes a new and inspiring approach towards the automated induction of semantic role labels. While this approach has some big restrictions, as it remains monolingual, and assumes successful parsing as a preprocessing step, it highlights a direction for a new kind of research. This direction is research that focuses on learning the meaning of sentences, which while inherently very challenging, is expected to be of great importance in the years to come, both in the field of machine translation and computational linguistics in general. While strong machine learning techniques will be important in this, good models, and effective forms of inductive bias may prove to be as important. It is possible that eventually black-box optimization methods will prove sufficient for many tasks. But ultimately, if we want humans to be able to give any interpretation and validation of what is learned, symbolic models of some form will keep their merit.

6.11 Conclusions

In the first part of this chapter we provide a formal characterization for the problem of determining the coverage of a word alignment by a given grammar formalism as the intersection of two partially ordered sets. These partially ordered set of TEUs can be formalized in terms of hypergraphs implementing forests (packed synchronous trees), and the coverage as the intersection between sets of synchronous trees generalizing the trees of Zhang et al. (2008a). This formal framework is then used to provide exact alignment coverage statistics, for ITG.

In the second part, the formal characterization is applied to general word alignments with the help of HATs. Prior to the extensive empirical study of this second part, it is shown that HATs can be used as a sufficient means to efficiently measure alignment coverage and other related metrics directly based on the induced HATs instead of explicitly performing actual intersection of TEUs. This is important, as the latter approach would be computationally very expensive. The empirical investigation gives some new insights, in particular the observation that at least for automatically aligned data, NF-ITG can only cover about 53% of the aligned sentence pairs for English–French and even less for the other language pairs. This goes against the common belief that NF-ITG would be sufficient for nearly all constructions occurring in actual, empirical, aligned translation data.

Another insight is that embedding complex alignment constructions up to some maximum source and target length inside atomic units, so that their complexity can

be ignored, provides no complete solution for the complexity observed in empirical word alignments. Complexity does reduce by such embedding, but when for example atomic fragments with a maximum length of 10 source/target words are allowed, a considerable fraction of complex non-binarizable alignments remains. For English–French this fraction is about 9%, while for English–German and English–Dutch this is about 16% and for Chinese–English even about 20%. This insight, combined with the fact embedding into long atomic units also leads to extreme sparsity, suggests that avoiding discontiguity by embedding into large atomic units may not be the best solution. This is an argument for preferring HIERO grammars or even stronger formalisms such as *synchronous linear context-free rewriting systems* (Kaeshammer, 2013) over formalisms that cannot model the inherent discontiguity of word alignments, whether used for translation or for preordering.[17]

A final insight given by this chapter is that the choice for the type of TEUs allowed, contiguous or discontiguous (general) TEUs, is theoretically important to make measurement of alignment coverage and related metrics well defined. It is also shown for ITG, that the type of TEUs used can make a real difference for alignment coverage results.

This concludes our quantitative empirical analysis of hierarchical translation equivalence. In work outside this thesis we have also looked at visualization of hierarchical translation equivalence (Maillette de Buy Wenniger and Sima'an, 2014b), which can play an important role in getting a better qualitative understanding of empirical translation equivalence in big parallel corpora.

[17] HIERO can deal with discontiguity only through lexicalization which is also restricted, nevertheless this is still much stronger than unlexicalized SCFGs or unlexicalized ITGs, which have no way at all to generalize in case of discontiguous mappings in the word alignment.

7

Conclusions

I get irritated, I get upset.
Especially when I'm in a hurry.
But I see it all as part of our training.
To get irritated is to lose our way in life.
– Haruki Murakami, *A Wild Sheep Chase*

In this thesis we introduced a new framework to represent hierarchical translation equivalence, called hierarchical alignment trees (HATs) (Sima'an and Maillette de Buy Wenniger, 2013). First, this framework is applied to produce bilingual reordering labels that help to significantly improve hierarchical statistical machine translation (SMT). Second, it is used to perform exact measurement of the coverage of word alignments by certain grammar formalisms as well as measure the complexity of word alignments.

HATs extend the existing frameworks of permutation trees (PETs) and normalized decomposition trees (NDTs). The set-permutation labels decorating every node in a HAT make the recursive reordering taking place at these HAT nodes explicit. This difference with NDTs, while arguably small, turns out to be crucial in facilitating straightforward extraction of reordering labels on the one hand and measurement of alignment complexity on the other hand. From a more theoretical point of view, this addition is also essential for producing a compact representation of hierarchical translation equivalence that retains all information that is present in the original word alignments.

The main contribution of this thesis is the application of HATs to improve hierarchical SMT, in particular with respect to word order. The reordering information present in the hierarchical translation structure induced by word alignments, is a rich resource currently not exploited by hierarchical SMT (Hiero). Hiero uses rule lexicalization to guarantee that the reordering decisions within its rules are based on at least some lexical context. However, this information is only of limited use within the

scope of rules with variables. It does not contribute sufficiently to the global coherence of reordering across rules, nor does it help to inform reordering for fully lexicalized rules (phrase pairs). In contrast, lexicalized orientation models have been successful in improving word order in phrase-based SMT and hierarchical SMT. These models show that adding reordering information to HIERO can be helpful.

In this thesis we enrich HIERO with reordering labels based on the hierarchical translation equivalence structure represented by HATs, induced by the same word alignments that are used to extract the rules. This means that the approach works without any additional resources. In particular, it requires no syntactic parsers or taggers, which are not available for all languages. HATs contain reordering labels (set permutations) for every phrase pair, that denote the recursive reordering within that phrase pair. Producing effective reordering labels based on the rich labels decorating HAT nodes turns out to be a non-trivial engineering effort. One reason for this is that there are many different complex labels in a HAT, so the number needs to be reduced to avoid issues with sparsity. Second of all, the raw HAT labels denote reordering of child phrases, from the perspective of the parent phrase. But another perspective, where reordering of the current phrase with respect to its embedding parent could be more effective. As it turns out, a sensible heuristic bucketing of HAT labels, yields a successful approach to produce effective reordering labels. Using such an approach we produce two types of labels:

- 0^{th}*-order* labels represent the reordering of the child phrases with respect to the current phrase.

- 1^{st}*-order* labels represent the reordering of the current phrase with respect to its embedding parent(s).

Since both types of labels yield information that could be useful, which type is better is an empirical question. In our translation experiments we evaluated the success of these labels for German–English, English–German and Chinese–English translation. Reordering labeled systems were compared against a HIERO baseline and a syntactically labeled SAMT baseline. The performance was evaluated using five different metrics: BLEU, METEOR, BEER, TER, KRS. The evaluation showed large, significant improvements for Chinese–English translation of about 1 BLEU point, and proportional improvements for all other metrics except TER. For English–German and German–English translation, smaller, but still significant improvements of about 0.2 BLEU were made. The reordering labeled systems also systematically outperformed SAMT. One factor that turned out to be crucial to the success of reordering labels, was the usage of elastic-substitution decoding. In a strict matching setting, reordering labels still yielded improvements for Chinese–English translation, but these improvements were much smaller (only about 0.3 BLEU) than when using elastic-substitution decoding. For English–German and German–English translation no clear improvements were made in a strict matching setting. Apart from the effect

of using elastic-substitution decoding versus strict matching, we also looked at the influence of three other dimensions on the performance:

- Type of labels: comparing 0^{th}-*order* with 1^{st}-*order* labels.
- Granularity of the labels: comparing the original labels with a further coarsened variant
- The set of features used to learn matching preferences (label substitution features): basic feature set or sparse feature set.

From these experiments we can draw the following conclusions. In general, 1^{st}-*order* labels seem to perform slightly better than 0^{th}-*order* labels, while there is some variation across languages and evaluation metrics. Furthermore, the original versions of the label sets perform better than their coarsened variants. The exception to this is when strict matching is used: in this scenario the coarse (ITG) variant of the 0^{th}-*order* labels gives the best performance for English–German and Chinese–English translation. This can probably be attributed to the fact that in a strict-matching setting, labels can block valid translations, and the chance of this increases as labels become more fine-grained. This means that it can be better to have coarser labels in this setting, because the advantage of having more descriptive labels does not weigh up against the loss caused by blocked translations. Lastly, with respect to the label substitution features used, we observed that in general the sparse feature set works better. Again there was an exception to this: English–German translation. We attribute the fact that the sparse feature set does not always work best, to the fact that it also creates a more difficult learning problem with a larger change of overfitting. Indeed, looking at scores for the development rather than test set, we observed that with the normal (uncoarsened) labels in the elastic-substitution decoding setting, the sparse feature set gave better performance for all language pairs. But because of overfitting, this did not carry over to performance on the test set in case of English–German.

Finally we did analysis experiments where we replaced the strong 4-gram language model with a weak unigram language model, which gives no information about the correct word order. In this setting all translation scores went down enormously, although the reordering labels still yielded a relative improvement in all settings. Interestingly, the relative improvement of using reordering labels went down for Chinese–English in the unigram language model setting, while it went up for German–English and English–German. But a clear difference between these language pairs is that in case of Chinese–English overall performance was really decimated when using the unigram language model, dropping by about 12 BLEU points. For German–English and English–German translation the performance loss was smaller, only about 5 BLEU points. Clearly this means that for Chinese–English, the language model was more crucial for moving towards an initial subset of reasonable translations, enabling the reordering labels to make a better choice from this already reasonable set.

Summarizing from these observations we conclude that:

- Reordering labels help improving performance in hierarchical SMT for the settings explored in this thesis.
- Using elastic-substitution decoding is crucial for getting good results with reordering labels.
- Reordering labels are complementary with the benefits for performance from using a strong language model.
- Reordering labels are competitive with syntactic labeling schemes, outperforming strict-matching SAMT in our experiments, while not requiring any additional linguistic or syntactic resources.

In chapter 6 we looked at measuring the coverage of word alignments by synchronous grammars. But what it means for a grammar to cover a word alignment is not obvious. Results for existing work that measure alignment coverage for grammars make different assumptions, and consequently results diverge. Furthermore most work has focused on ITG using lower bounds rather than exact measurement. To overcome these problems we proposed a well defined theoretical framework based on the intersection of the sets of translation equivalence units (TEUs) induced by word alignments and the grammar. To cover a word alignment, all TEUs induced by the word alignment must be induced in the set of TEUs derivable from the grammar, and furthermore these TEUs must have the same recursive subsumption structure in the word alignment and the grammar. This yields a sufficient framework to facilitate exact, well-defined measurement of word alignment coverage for different types of synchronous context-free grammars. In the second part of this chapter we showed that by using HATs, albeit not implementing an explicit intersection of sets of TEUs, exact measurement of word alignment coverage by grammars can be efficiently computed. This is possible, since HATs by construction are made to form the simplest possible structure that captures the hierarchical translation equivalence structure induced by word alignments. Hence, in order to be able to cover a word alignment, a grammar formalism must allow synchronous productions at least as complex as the most complex production observed in the set of HATs induced by a word alignment. Therefore, from the complexity of induced HATs, the type of grammar formalism sufficient to cover a word alignment can be induced. One remarkable finding from the empirical analysis of hierarchical translation equivalence, is that a large fraction of empirical word alignments constitutes non-binarizable permutations that cannot be covered by ITG grammars. For manual word alignments, between about 71% and 91% of the word alignments was coverable, depending on the language pair and data set. For automatic word alignments this went down even further, and between only about 45% and 52% of the alignments was coverable by binarizable permutations, depending on the language pair. Also in all these cases, the coverage was only marginally increased by moving from binarizable permutations to general permutations; and for the main jump to full coverage an upgrade to general non-bijective mappings (general set-permutations) is necessary. These findings are important because much current work

on translation as well as pre-ordering before translation, relies on either a form of ITGs or a form of PETs. But while computationally and conceptually hard, these finding suggest that much can still be gained from considering the full set of set-permutations. The success of reordering labels that go beyond the well known cases from ITG, are a first proof that such efforts can be fruitful.

This thesis is titled "Aligning the Foundations of Hierarchical Statistical Machine Translation". In it, we saw that hierarchical translation equivalence, as induced by word alignments and represented by HATs forms a foundation for a new direction of research in SMT that makes the most of the full set of hierarchical translation equivalence relations induced by word alignments. This is in contrast with standard Hiero, which uses word alignments only to extract phrase pairs and hierarchical phrase pairs, and then discard them, leaving most of their valuable information unused. We note that phrase-based translation models typically do use orientation models (Tillmann, 2004; Galley and Manning, 2008) which also exploit reordering information obtained from the word alignments, and that these models can also be adapted for Hiero (Nguyen and Vogel, 2013b; Huck et al., 2013). But in comparison to orientation models adapted for use in Hiero, our approach with reordering labels has the advantage of being more general[1] and requiring less radical changes to the decoder.[2] A last conceptual advantage of reordering labels over orientation models, is that reordering labels provide a form of Markovization, which orientation models do not provide. Consequently, it provides a direct way of promoting the coherence of the produced translations, which orientation models can only promote indirectly.

The creation of HATs constitutes an effort with similarity to the creation of treebanks for parsing. Currently these HATs have been used to extract simple but effective reordering labels, and perform exact measurement of word alignment coverage, as well as visualize hierarchical translation equivalence. But the vision behind HATs is not limited to only these applications. Forming richer labels, as well as possibly learning of labels using HATs are possibilities opened up by this framework. While existing work with PETs has shown that these structures are also effective for pre-ordering and evaluation. This thesis has shown that word alignments are a main foundation for hierarchical SMT and that using it only to extract rules means doing little with an information source that is much richer. By improving word order and global coherence through reordering labels we have given a first strong evidence for the power of HATs and its applications. But we think this is only the beginning.

[1] More and less specific reordering labels have been explored, and reordering can be approached with respect to the embedding parent or with respect to reordered children. Additionally, sparse label substitution constraints can be used to further refine the reordering contexts. In contrast, phrase orientations have not been tested for orientations other than the three cases based on ITG.

[2] In practice the changes made to the decoder to effectively implement soft matching of labels are also considerable. Theoretically however, our approach can also be implemented by using specialized label switch rules that implement soft matching and soft constraints without any changes to the decoder. This is important, because it means that our approach remains within the standard SCFG formalism, while the other approaches effectively changes the decoding formalism to something beyond this formalism.

Acronyms

AI artificial intelligence.

BEER better evaluation as ranking.

BITT binary inversion-transduction tree.

CCG combinatory categorial grammar.

CFG context-free grammar.

EM expectation maximization.

HAT hierarchical alignment tree.

HIERO hierarchical phrase-based translation, as first proposed by (Chiang, 2005).

HMM hidden Markov model.

ITG inversion transduction grammar.

MERT minimum error rate training.

MIRA margin infused relaxed algorithm.

MT machine translation.

NDT normalized decomposition tree.

NF-ITG normal form inversion transduction grammar.

PET permutation tree.

POS part-of-speech.

SAMT syntax-augmented machine translation.

SCFG synchronous context-free grammar.

SMT statistical machine translation.

TEU translation equivalence unit.

Appendix A

Appendix

A.1 Word Alignment Symmetrization

Symmetrization of word alignments is an important step to get good many-to-many word alignments based on one-to-many alignments in two directions. If this step is not done well, the translation system may suffer unnecessarily from low accuracy or low coverage. Going beyond the very basic symmetrization methods of intersection and union, Koehn et al. (2003) introduces a series of alternative heuristic schemes for merging source-to-target and target-to-source word alignments into better many-to-many word alignments. These schemes build further on the intersection-based *refined method* first introduced by (Och et al., 1999).

Let A_1 be the set of source-to-target alignments and A_2 be the set of target-to-source alignments. Furthermore let $A_{union} = A_1 \cup A_1$ and $A_{intersect} = A_1 \cap A_1$. Following the community conventions, an alignment link from source word i to target word j will be called an *alignment point* in our discussion and denoted as (i, j). The heuristic methods start from an initial alignment set A which is initialized as the intersection $A \leftarrow A_{intersect}$. It then extends A by iteratively adding more alignment points from the union in two stages. In the required first stage the initial alignment is extended by extra alignment points within the neighborhood of the current points. In the optional second stage additional alignment points are added that are not in the neighborhood.

Differences in the details of how the extra alignments are included in the two stages yield alternative schemes known as[1]:

- grow
- grow-final

[1]There actually seems to be a lot of confusion in the naming of these different schemes. What in (Koehn et al., 2003) is referred to as "diag-and" is more commonly referred to as "grow-diag-final-and" while what it refers to as "diag" is more commonly referred to as "grow-diag-final". In combination with the fact that the description of these methods in the literature is quite dense and scattered, it can be hard to understand really what is going on, without diving into the source code of the Moses scripts. This was one of the motivations to add this appendix.

- grow-final-and
- grow-diag
- grow-diag-final
- grow-diag-final-and

In the required first stage, all methods iteratively add additional alignment points (i, j) from A_{union} to A under the condition that

- *Neighborhood condition*: These are neighbors of existing alignment points already in A.
- *Partially excluded condition*: Either i or j is not part of any other existing alignment point (X, i) or (j, Y) in A for all X and Y.

The methods differ in how they define what neighbors are, i.e. their *neighborhood function*, and there are two alternatives. First the *grow* schemes use only block-neighboring points, second the *grow-diag* schemes include diagonal neighbors as well (Koehn et al., 2003). In the optional second stage more alignment points are added by the schemes with the *-final* and *-final-and* in their suffixes. These schemes differ in the condition they use to include the additional alignment points, which is one of the following:

- *Partially excluded condition* (as before)
- *Fully excluded condition*: Both i and j are not part of any other existing alignment point (X, i) or (j, Y) in A for all X and Y.

The schemes with the *-final* suffix use the partial excluded condition, while those with the *final-and* suffix use the stricter fully excluded condition.

We note that the original method discussed by (Och et al., 1999; Och and Ney, 2003) is equal to *grow-final*. We also note that *grow-diag-final* and *grow-diag-final-and* seem to be the most popular for phrase based and hierarchical phrase-based translation. The reason is that adding the diagonal neighbors further increases coverage (recall) at the cost of precision, which generally turns out to be a good tradeoff in phrase based and hierarchical phrase based translation. Which alignment scheme is best for translation turns out to depend both on the language pair as well as whether in-domain or out-domain data needs to be translated by the system. Initial comparisons were given by (Och and Ney, 2003) and a more detailed study is provided by (Wu and Wang, 2007). As a final remark, the *grow-final-and* scheme, while being a valid option, seems to be rarely used in practice. The reason might be that it lowers coverage even below that of the original *grow-final* scheme, which probably in most cases means a loss in the quality of the tradeoff between recall and precision, meaning worse translation results.

A.2 Feature Weights Training

So far we discussed how in the original source-channel approach the language model and the translation model are combined to produce the probability of translations. As we saw, the translation model is typically subdivided into various subcomponents, including phrase weights, lexical weights, reordering (distortion) scores, word and phrase penalties. Each of these components has a weight that needs to be trained. In the log-linear model setup many more features can be combined. These features need not be probabilities but can be numeric or binary as well. Finding good feature weights is essential in order to build a system that produces high quality translations. We will now review various discriminative training methods for learning feature weights. We start the discussion with the well known Minimum Error Rate Training (MERT) (Och, 2003). MERT cannot scale to many different features, and our description will help to understand why this is the case. But in the context of this thesis, a feature weights training method that does scale to many features is essential. We need such a method to enable the learning of soft reordering constraints, as discussed in chapter 5, which forms the core of the experimental work in this thesis. After discussing MERT, we therefore give a detailed description of some recent feature weight training methods that scale to thousands or even millions of features. We focus on the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003), and a particular efficient and easy to adopt version of it called Batch MIRA (Cherry and Foster, 2012).

MERT MERT was proposed as a method to directly optimize the score on the final the evaluation metric, such as BLEU (Papineni et al., 2002) or TER (Snover et al., 2006). It does so by repeatedly finding the optimal parameters along a line in the N-dimensional search space of feature weight values. MERT performs its weight optimization using N-best lists of best translations produced by the decoder on a development (dev) set. This causes a risk that the changes to the weights improve the choice of translations within these lists but yield different and possibly worse translations when the updated parameters are used to again decode the dev set. MERT accounts for this problem by decoding the dev set repeatedly and keeping all the translations that are produced over previous rounds of decoding. Before every iteration of parameter optimization, for every sentence the newly produced N-best list is merged with the collected previous translations for that sentence. Parameter optimization is then done on the merged sets of translations. This "outer loop" of decoding, N-best list merging, parameter optimization, (re-)decoding was introduced for MERT but has been followed also by more advanced tuning methods that we will discuss after MERT. But first we will discuss MERT's line search optimization, the core of MERT, in more detail.

Line Search Optimization for MERT Let $\mathbf{f}_1^S$ be a set of dev source sentences with associated reference translations $\mathbf{r}_1^S$. Furthermore assume a set of K different candidate translations $C_s = \{\boldsymbol{e}_{s,1}, \ldots, \boldsymbol{e}_{s,K}\}$ for every sentence $\boldsymbol{f}_s$ in $\boldsymbol{f}_1^S$. Then the best translation

for a sentence $\boldsymbol{f}_s$ in the dev set given the weight vector λ_1^M is defined as:

$$\hat{\boldsymbol{f}}(\boldsymbol{f}_s, \lambda_1^M) = \arg\max_{e \in \boldsymbol{C}_s} \left\{ \sum_{m=1}^{M} \lambda_m \phi_m(\boldsymbol{e}|\boldsymbol{f}_s) \right\} \tag{A.1}$$

Let $E(r, e)$ be an evaluation function that gives the sentence error count resulting from selection of e when r is the reference. The goal of MERT is then to minimize the total error count on the dev set:[2]

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^{S} E(\boldsymbol{r}_s, \hat{\boldsymbol{f}}(\boldsymbol{f}_s, \lambda_1^M)) \right\} \tag{A.2}$$

Because of the argmax operator in A.1 it is not possible to solve A.2 analytically, nor can a gradient be computed to permit gradient descent methods. Therefore an approximate solution is proposed which optimizes the weights by keeping most of them constant, and then finding the best new parameters along a line in the search space. Searching along a line is done by keeping the weights λ_1^M constant except for adding a constant weight vector d_1^m scaled by a parameter γ which determines the movement along the line. This yields an optimization problem of the following form:

$$\hat{\boldsymbol{e}}(\hat{\boldsymbol{f}}; \gamma) \;=\; \arg\min_{\boldsymbol{e} \in \boldsymbol{C}} \{t(\boldsymbol{e}, \boldsymbol{f}) + \gamma \cdot m(\boldsymbol{e}, \boldsymbol{f})\} \tag{A.3}$$

where $t(\cdot)$ and $m(\cdot)$ are constants with respect to λ_1^M. Thus every candidate solution in the optimization has a score that is a linear function of γ (a line). Since the function

$$f(\gamma; \boldsymbol{f}) \;=\; \min_{\boldsymbol{e} \in \boldsymbol{C}} \{t(\boldsymbol{e}, \boldsymbol{f}) + \gamma \cdot m(\boldsymbol{e}, \boldsymbol{f})\} \tag{A.4}$$

is piecewise-linear, the output of A.3 as γ changes consists of a finite sequence of intervals for which the best translation is stable. These intervals with a constant best translation plus the error score for those translations can be efficiently computed for every sentence. Combining the intervals with values over all sentences, an aggregated total error score function is computed, that consists again of a finite amount of intervals with a constant value. The optimal value for γ that maximizes the total error over all sentences along the line is then easily read out from this aggregated total score function.

Within one iteration, MERT executes line-search based parameter optimization for multiple lines. In the simplest setup where the direction of lines correspond to changing single parameters, every parameter is normally changed once. The order of the parameters can be chosen greedily based on trial changes whereby the parameter changes that give the most improvements are executed first.[3] One important detail in the implementation of MERT is how the direction for the search lines is to be

[2] Assuming the dev set is chosen to be a representative corpus for the sentences that need to be translated after tuning i.e. the test set.

[3] This is the default strategy used in Moses's MERT implementation according to (Cer et al., 2008).

chosen. In the simplest case (Och, 2003), search is done along the directions of single parameters by just changing one of these and keeping the rest constant. But searching along different lines, in particular using random directions, gives better results (Cer et al., 2008). Interestingly the random directions work even better than Powell's search method (Powell, 1964; Press et al., 2007), which aims to find a set of mutually non-interfering search directions.

But even with such improvements, MERT does not scale to many features. As the number of dimensions of the search space increases with more features, the line search slows, and suffers increasingly from local optima, failing to find a globally good solution. Furthermore, BLEU does not consider model scores but is only determined by the ranking of translations. This means that λ_1^M can be arbitrarily scaled, without changing the BLEU scores. Since MERT's direct optimization considers only the BLEU[4] score, it suffers from the extra instability of this scale invariance, which increases with more features. In such a complex search space, MERT's simple greedy and direct optimization strategy simply gets lost. So we need a feature weight training method that applies a more holistic optimization object, leading to more stability and better scaling to many features. We will discuss such a method next.

MIRA The Margin Infused Relaxed Algorithm (MIRA) was first used by (Watanabe et al., 2007) and later refined by (Chiang et al., 2008, 2009; Chiang, 2012). MIRA builds upon on the two fundamental concepts of *cost* (or loss) and *margin*. The cost of choosing a translation e_j given some oracle translation[5] e_i^* and a reference is defined as the difference between their scores on some evaluation metric, given the reference. In the typical case that BLEU is used as the evaluation metric of choice we have:

$$cost_i(e_j, e_i^*) = BLEU_i(e_i^*) - BLEU_i(e_j) \tag{A.5}$$

Let $\vec{\lambda} = \lambda_1^M$ be the feature weight vector. We will use $\vec{\lambda}_t = \lambda_1^M(t)$ to indicate the feature weight vector at a specific iteration t, when necessary. The margin, or distance in model scores, between e_j and e_i^* is defined as:

$$\text{margin}(e_j, e_i^*) = \vec{\lambda} \cdot (\vec{\phi}_i(e_i^*) - \vec{\phi}_i(e_j)) \tag{A.6}$$

In words, the summed differences of feature values multiplied by feature weights. Let ε be the set of all reachable hypotheses. Conceptually, the aim of MIRA is to separate all pairs of reachable hypotheses (candidate translations[6]) $\langle e_\alpha, e_\beta \rangle$ with $e_\alpha, e_\beta \in \mathcal{E}$

[4]Or some other evaluation metric that is independent of hypothesis model score.

[5]In principle we want to compute the cost with respect to some *correct* translation. But the reference translation might not have a derivation in the model, therefore a set of surrogate references called *oracles* that can be produced by the decoder is used instead. When using BLEU as evaluation metric, these oracles are typically also be used to augment the actual references, to provide desirable smoothing when computing BLEU at the sentence level.

[6]To be more precise, the aim of MIRA is to separate all pairs of reachable derivations, whereby the same translation can have possibly many derivations, see (Chiang, 2012). Here to simplify the notation, we gloss over this detail, and following (Cherry and Foster, 2012) use translations in the rest of the discussion, while actually referring to derivations and translations yielded by these derivations.

in such a way that the margin margin(e_j, e_i^*) between them is at least as big as the cost $cost_i(e_\alpha, e_\beta)$ of choosing e_α instead of e_β. Such a separation guarantees that not just one best hypothesis get the highest weight, as in MERT, but also globally the hypothesis space is properly structured such that lower quality hypotheses will have correspondingly lower model scores. This leads to a much higher stability of the optimization, and allows this method to scale to millions of features. This is in strong contrast with MERT, which does not scale well beyond 15 features, while failing badly beyond 30 features.

Formally the weight update rule of MIRA for an example i is usually given as:

$$\begin{aligned} \vec{\lambda}_{t+1} &= \arg\min_{\vec{\lambda}'} \frac{1}{2\eta} \|\vec{\lambda}' - \vec{\lambda}_t\| i^2 + \xi_i \\ \text{subject to } & \vec{\lambda}' \cdot (\vec{\phi}_i(e_i^*) - \vec{\phi}_i(e_j)) \geq cost_i(e_j, e_i^*) - \xi_i \qquad \forall e_j \in \mathcal{E} \end{aligned} \tag{A.7}$$

In this ξ_i is a slack variable, introduced to account for inseparable instances caused by noise in the data; and η is a step size.

The structured hinge loss employed by MIRA is:

$$\begin{aligned} L_i(\vec{\lambda}) &= \max_{e \in \mathcal{E}_i} \left[cost_i(e_j, e_i^*) - \text{margin}(e_j, e_i^*) \right] \\ &= \max_{e \in \mathcal{E}_i} \left[cost_i(e_j, e_i^*) - \vec{\lambda} \cdot (\vec{\phi}_i(e_i^*) - \vec{\phi}_i(e_j)) \right] \\ &= -\vec{\lambda} \cdot \vec{\phi}_i(e_i^*) + \max_{e \in \mathcal{E}_i} \left[cost_i(e_j, e_i^*) + \vec{\lambda} \cdot \vec{\phi}_i(e_j) \right] \end{aligned} \tag{A.8}$$

When optimizing (A.8), in theory we have to maximize over exponentially many translations, or derivations to be more exact[6]. What is more, we want in fact to optimize the average loss over all translations in the training set, not just the loss for one translation as in (A.8). But this is computationally infeasible, and in practice not necessary either. Instead it suffices to perform incremental optimization on the training set sentences separately, using a restricted set of pairs of undesired translations e^- and oracle translations e^* for each sentence. There are many alternatives for choosing these undesired and oracle translations. As part of this selection, the weights of the model score and cost component for these translations in the loss function can also be changed. Varying these properties and the details of the optimization method, many different discriminative training algorithms similar to MIRA (Tillmann and Zhang, 2006; Liang et al., 2006a) and variants of MIRA (Crammer et al., 2006; Chiang et al., 2008; Chiang, 2012; Cherry and Foster, 2012) have been proposed. Choosing for e^* translations that maximize a combination of high model score and a low cost ("hope" hypotheses) and for e^- translations with a high model score and a high cost ("fear translations") is particularly successful (Chiang et al., 2008).

Working with multiple pairs $\langle e^*, e^- \rangle$ complicates the optimization problem, yielding a quadratic program (QP) with a set of linear constraints that cannot be solved analytically, but require Hildreth's algorithm (Hildreth, 1957) or optimization in the style of Sequential Minimal Programming (Platt, 1998). Chiang (2012) work

with multiple pairs of hope and fear hypotheses, focusing on the worst constraint violating translations, and but introduce an efficient cutting-plane algorithm based on (Tsochantaridis et al., 2004) to optimize this restricted set of active constraints.

But when working with only a single pair $\langle e^*, e^- \rangle$, an analytical solution is possible, yielding a simple closed-form update that performs well[7] (Crammer et al., 2006; Eidelman, 2012). Let C be a regularization parameter determining the maximum allowed size of a weight change. The update can be performed in two steps:

$$\eta_t = min\left[C, \frac{cost_i(e_j, e_i^*) + \text{margin}(e_j, e_i^*)}{||(\vec{\phi}_i(e_i^*) - \vec{\phi}_i(e_j))||^2}\right] \tag{A.9}$$

$$\vec{\lambda}_{t+1} = \vec{\lambda}_t + \eta_t(\vec{\phi}_i(e_i^*) - \vec{\phi}_i(e_j)) \tag{A.10}$$

At the end of some maximum number of epochs J or when the weights no longer change, MIRA terminates. At this point the weight vectors over all iterations are averaged to get the final weight vector used for decoding the test set.

This standard *online*[8] version of MIRA is relatively slow, but parallelization can make it faster, and (Chiang, 2012) describes and compares various ways to implement this.

Batch MIRA The online version of MIRA as presented by (Chiang et al., 2008; Chiang, 2012) has several disadvantages, which are : 1) Being hard to parallelize, 2) being complex and still relatively slow 3) requiring a tight coupling between decoding and tuning, which severely hampers modularity and prohibits reuse with different decoders. To deal with these limitations (Cherry and Foster, 2012) proposes new batch-versions of MIRA using either K-best lists or lattices. These algorithms first decode the sentences in the dev set to produce sets of translations for each of them $[\tilde{\mathcal{E}}']_1^n$. This collection is then combined with the previous translations produced by the decoder $[\tilde{\mathcal{E}}]_1^n] = [\tilde{\mathcal{E}}]_1^n \cup [\tilde{\mathcal{E}}']_1^n$. The combined collection of translations $[\tilde{\mathcal{E}}]_1^n]$ is then used to run MIRA, pseudo-decoding (effectively reranking) the sentences in the dev set in random order, restricting the options to the set of translations present in $[\tilde{\mathcal{E}}]_1^n]$ and performing the MIRA weight update based on that set. These three steps are repeated, in total for J epochs. At the end of each epoch j, the aggregate epoch weight vector $\vec{\lambda}_j^{avg}$ is computed by averaging the weight vectors obtained after decoding each of the n sentences for that epoch. The aggregate epoch weight vector amongst the J epochs that yields the maximum evaluation score on the dev set is used as the final weight vector.

The K-best version of Batch MIRA is particularly easy to use out of the box, requiring only K-best lists that can be produced by any phrase-based or hierarchical phrase-based decoder. At the same time (Cherry and Foster, 2012) shows that this

[7] A particularly clear and easy to understand description of how to implement this simplified version of MIRA is given in (Eidelman, 2012).

[8] Online means that the algorithm interleaves decoding and weight updating, decoding one sentence, then updating the weights and then decoding again with the updated weights.

version performs consistently at the top of state-of-the-art discriminative feature weight training algorithms. It loses only a bit of performance in comparison to the lattice version of Batch MIRA. Later in this thesis in chapter 5, we will use this algorithm in all of our experiments for feature weights training.

Other Large-Scale Discriminative Training Methods We will now very shortly mention some other popular and recent discriminative feature weights learning methods that scale to large feature sets.

Minimum Bayes Risk (MBR) is a framework that can be used both to improve decoding (Kumar and Byrne, 2004; Tromble et al., 2008) as well as to perform discriminative feature weight training (Zens et al., 2007; Li and Eisner, 2009). MBR builds on the idea that rather looking for the **maximum a posteriori** (MAP) translation, we want a translation that minimizes the expected loss. Conceptually, this is achieved by choosing a translation that it is highly similar to most other very likely translations, so that choosing it causes minimal expected loss, even though it may itself not be exactly the most likely (MAP) translation. Let $\boldsymbol{e}$, $\boldsymbol{e}'$ be translations and $\boldsymbol{a}$, $\boldsymbol{a}'$ be word alignments. The MBR objective can then be formally described by the following decision rule:

$$\boldsymbol{e}_{\text{best}}^{\text{MBR}} = \arg\max_{\boldsymbol{e}} \sum_{\boldsymbol{e}',\boldsymbol{a}'} L(\langle \boldsymbol{e}, \boldsymbol{a}\rangle, \langle \boldsymbol{e}', \boldsymbol{a}'\rangle) p(\boldsymbol{e}', \boldsymbol{a}' | \boldsymbol{f}) \tag{A.11}$$

in this $L(\langle \boldsymbol{e}, \boldsymbol{a}\rangle, \langle \boldsymbol{e}', \boldsymbol{a}'\rangle)$ is a loss function that expresses the loss achieved when $\langle \boldsymbol{e}, \boldsymbol{a}\rangle$ is chosen but $\langle \boldsymbol{e}', \boldsymbol{a}'\rangle$ is the actual translation-alignment pair. Summing over all combinations of $\boldsymbol{e}'$ and $\boldsymbol{a}'$, and multiplying the loss obtained when each of these combinations is correct with the probability of that being the case, the expected loss for a certain choice of $\boldsymbol{e}$ is obtained. Minimizing this expected loss is the MBR objective.

When using the MBR framework for learning, a loss function expressing the MBR objective needs to be defined and then minimized. Here we summarize the approach taken by (Cherry and Foster, 2012). Define the cost function for the choice of translation $\boldsymbol{e}$ as its negative BLEU score: $cost_i(\boldsymbol{e}) = -\text{BLEU}_i(\boldsymbol{e})$. As in Batch MIRA, let $\tilde{\mathcal{E}}$ be the set of translations (for a particular sentence) collected in batch decoding of the dev set in the current and all previous iterations. The loss function is then defined as the expected cost over translations in $\tilde{\mathcal{E}}$,in this case the expected BLEU score:

$$L_i(\vec{\lambda}) = \mathbb{E}[cost_i(\boldsymbol{e})](\vec{\lambda}) = \frac{\sum_{e \in \tilde{\mathcal{E}}} \left[\exp(\vec{\lambda} \cdot \vec{\phi}_i(e)) cost_i(e)\right]}{\sum_{e' \in \tilde{\mathcal{E}}} \exp(\vec{\lambda} \cdot \vec{\phi}_i(e'))} \tag{A.12}$$

This expected cost function is decreased by assigning more probability mass to derivations with high BLEU score. The objective is smooth and non-convex, and with gradient-based optimizers a local minimum can be found. Cherry and Foster (2012) use stochastic gradient descent (Bottou, 2010) to optimize it.

The work by (Zens et al., 2007) suggests that using the MBR framework for training and decoding are complementary: using expected BLEU as the training criterion and

MBR as the decision rule during decoding gave the best results in their experiments. Related to MBR is the work by (Li et al., 2009) on *variational decoding*. This work introduces a variational method to approximate actual summation of the probabilities for alternative derivations of the same translation when searching the MAP translation during decoding, which is normally just ignored and replaced by the simpler goal of finding the Viterbi derivation.

Pairwise Ranking Optimization (PRO) (Hopkins and May, 2011) like MERT and Batch MIRA is another algorithm that fits in the MERT structure of decoding followed by a separate feature weight optimization phase. As in MERT and Batch MIRA optimization is done on the incrementally grown set of translations collected by combining the K-best lists (or lattices) produced by the decoder over all finished (decoding) iterations. Let S_i be a sample of translation pairs $\langle e_g, e_b \rangle$ such that $BLEU_i(e_g) > BLEU_i(e_b)$. PRO optimizes a loss function that is determined by the sum of wrong rankings within the translation pairs of S_i:

$$L_i(\vec{\lambda}) = \sum_{\langle e_g, e_b \rangle \in S_i} 2 \cdot \max\left((1 + \lambda_1^M \cdot (\vec{\phi}_i(e_b) - \vec{\phi}_i(e_g))),\ 0\right) \tag{A.13}$$

This loss is zero only if for every $\langle e_g, e_b \rangle \in S_i$ the models core of the higher evaluation score example e_g is at least 1 higher than that of the example with the lower evaluation score e_b. This particular form of the loss function, that ignores the actual BLEU scores, and effectively only cares about getting the *ranking* correct for the sampled pairs of translations, facilitates very simple and flexible optimization. The loss function used by PRO can be optimized by using any of-the-shelf binary classifier, such as Support Vector Machines (Cortes and Vapnik, 1995; Platt, 1998) or Perceptron learning (Freund and Schapire, 1999). The classifier learns to predict if the ranking of pairs is correct or not, by training it with one positive and one negative example for each sampled pair. As a result of this training, the classifier learns the weights $\vec{\lambda}$ necessary to obtain correct classifications.

Like MIRA, PRO learns to separate better from worse translations by a margin. But unlike MIRA, PRO uses a fixed margin of 1 for all its separations, thereby completely ignoring the magnitude of the quality- and model-score differences between translations. Furthermore PRO uses a sum in place of MIRA's max, with the result of assigning equal credit for every correctly ordered pair of translations in the sample. But this can be misleading, since not all correctly ordered pairs are equally difficult to separate nor equally important for the final BLEU score. This may be one reason why MIRA, which focuses directly on resolving the worst constraint violations, is still more successful than PRO to find good feature weights, especially for medium and big feature sets (Cherry and Foster, 2012).

Other Feature Weight Training Algorithms We gave an overview of some of the most popular classic and more recent feature weight training algorithms. Without

aiming for completeness, we mention here three more related approaches, while not going into depth.

AROW (Crammer et al., 2009) might be seen as a second generation version of MIRA. It introduces an adaptive regularization of weights. It replaces the weight vector $\vec{\lambda}$ of MIRA with a Gaussian distribution over weight vectors. This allows AROW to model the fact that changes to $\vec{\lambda}$ can pose different levels of risk for different directions. More details and a thorough comparison between ARROW and MIRA for usage in Machine Translation is given in (Chiang, 2012).

Cherry and Foster (2012) apply yet another method to the feature weights learning task called **Structured SVM** (Tsochantaridis et al., 2004) which is also highly similar to MIRA. In contrast to MIRA this method directly minimizes the *sum* of hinge-losses over all sentences, using a batch algorithm. Another difference with MIRA is that this method uses an explicit regularization term.

Finally we mention **RAMPION** (Gimpel and Smith, 2012) as another algorithm that is highly similar to MIRA, if not another form of MIRA. This work was published around the same time as (Cherry and Foster, 2012) and simplifies the version of MIRA (Chiang et al., 2008) in a very similar way, by proposing an algorithm that optimizes the loss function using just one pair of oracle (hope) and undesirable (fear) translations at a time. A contribution of this paper is that it shows that most versions of MIRA, optimize loss functions that are actually variants of *structured ramp loss* rather than *structured hinge loss*. The reason for this is that these algorithms are working with surrogate references (oracles) instead of real references, in the optimization.

findSAMTLabelForSpan
Input :
parse : the parse used for labeling.
{*i*, *j*} : the first and last positions of the span to be labeled.

Output: The SAMT label for the span [*i*, *j*].

```
// Extract an n × n table containing the constituents
// found in the parse used for labeling.
constituentTable ← findConstituentTable(parse);
if constituentTable[i][j] ≠ NULL then        // Find proper constituent
  | return constituentTable[i][j];
end
for splitPoint ← (i + 1) to (j) do                // Find single plus label
  | leftLabel ← constituentTable[i][splitPoint − 1];
  | rightLabel ← constituentTable[splitPoint][j];
  | if (leftLabel ≠ NULL) ∧ (rightLabel ≠ NULL) then
  |   | return leftLabel + "+" + rightLabel;
  | end
end
spanLength ← (j − i) + 1;
for extraSpanLenght ← 0 to (n − spanLength) do    // NT1/NT2 or NT2\NT1
label
  | if (j + extraSpanLength) ≤ (n - 1) then           // Find NT1/NT2 label
  |   | rightGrownLabel ← constituentTable[i][j + extraSpanLength];
  |   | substractedLabel ← constituentTable[j+1][j + extraSpanLength]; if
  |   | (rightGrownLabel ≠ NULL) ∧ (substractedLabel ≠ NULL then
  |   |   | return rightGrownLabel + "/" + substractedLabel;
  |   | end
  | end
  | if (i - extraSpanLength) ≥ 0 then                 // Find NT2\NT1 label
  |   | leftGrownLabel ← constituentTable[i - extraSpanLength][j];
  |   | substractedLabel ← constituentTable[i - extraSpanLength][i - 1]; if
  |   | (leftGrownLabel ≠ NULL) ∧ (substractedLabel ≠ NULL then
  |   |   | return substractedLabel + "\" + leftGrownLabel;
  |   | end
  | end
end
if allowlowDoublePlus then                  // Double-plus label allowed?
  | label ← findDoublePlusLabel(i,j);          // Find double-plus label
  | if Label ≠ NULL then
  |   | return label;
  | end
end
return "X";                                        // Return default label
```

Algorithm 11: Algorithm that finds the SAMT label for a span given a table with parse constituents.

$$X \quad \rightarrow \quad \text{koks kunnen } X^{\boxed{1}} \quad \| \quad \text{cooks can } X^{\boxed{1}}$$

$$\left\{ \begin{array}{l} p(H_0 = S, H_1 = VB \mid r) = 0.5 \\ p(H_0 = S, H_1 = VP \mid r) = 0.3 \\ p(H_0 = S/PP, H_1 = VB \mid r) = 0.2 \end{array} \right\} \tag{A.14}$$

$$X \quad \rightarrow \quad \text{bakken} \quad \| \quad \text{bake}$$

$$\left\{ \begin{array}{l} p(H_0 = VP \mid r) = 0.8 \\ p(H_0 = VB \mid r) = 0.2 \end{array} \right\} \tag{A.15}$$

$$X \quad \rightarrow \quad \text{bakken} \quad \| \quad \text{bins}$$

$$\left\{ p(H_0 = NNS \mid r) = 1.0 \right\} \tag{A.16}$$

Figure A.1: Preference Grammar Rules for Dutch-English. This minimal working example contains three Hiero rules with associated distributions over implicit SAMT labels.

A.3 SAMT Labeling Algorithm

The pseudocode for the SAMT labeling procedure is shown in algorithm 11. The pseudocode describes the procedure for producing the label for a single span, starting with the generation of a table of constituents. In practice, this algorithm would be wrapped by a double loop to produce a label chart with labels for all spans. The table of constituents only needs to be produced once.

A.4 Soft Constraints

Preference Grammars

Preference grammars (Venugopal et al., 2009) were proposed as a method to implement labels as soft constraints, overcoming both the weaknesses of unlabeled Hiero and methods that strictly enforce labels such as SAMT. Hiero rules use only the label X (plus the start symbol S) and consequently ignore the context in the substitution of rules. SAMT uses syntactic target side labels which strictly enforces a form of syntactic well formedness on the target side. SAMT prohibits certain invalid substitutions but at the price of also blocking valid substitutions. It also further worsens the problem of spurious ambiguity by creating many competing different labeled variants of unlabeled Hiero derivations. Preference grammars add joint probabilities over implicit label assignments to unlabeled Hiero rules. These distributions capture the empirical joint distributions of rule labels, as estimated by heuristic estimation (Chiang, 2005; Zollmann and Venugopal, 2006) in a packed way. Figure A.1 shows a minimal working example of a preference grammar for Dutch-English translation, with SAMT labels, which we will use in the discussion. This grammar supports

the combination of the two rules to correctly translate the Dutch sentence "koks kunnen bakken" as "cooks can bake". This translation can be formed with different consistent assignments of labels for the right-hand-side nonterminal in the HIERO rule "$X \rightarrow$ koks kunnen $X_{\boxed{1}}$ || cooks can $X_{\boxed{1}}$ " and the left-hand-side nonterminal in the lexical rule "$X \rightarrow$ bakken || bake". The probabilities for the consistent label assignments for pairs of rules can be multiplied and summed over all consistent pairs to yield a total probability for consistent label assignments. In contrast, there exists no label assignment that is consistent with the HIERO rule for the second lexical rule "$X \rightarrow$ bakken || bins". This means that the invalid translation "cooks can bins" has no latent labeled derivations supporting it. We will next give a more formal description of preference grammars, using our working example to illustrate its application.

The computation of feature $P_{syn}(d)$ Preference grammars build derivations like standard (unlabeled) HIERO grammars, but they add a feature $p_{syn}(d)$ that determines the chance that these are build up from rules with consistent latent labels. This feature approximates the marginalization of probabilities over all valid, hidden labeled variations for the unlabeled derivations. The feature is computed as a product of factors[9], one factor per nonterminal n_j in a derivation:

$$p_{syn}(d) = \prod_{j=1}^{|d|} \phi_j \tag{A.17}$$

Conceptually these factors measure the level of consistency between the preference distributions of substituting and substituted rules. Naively, every factor ϕ_j could be determined by the label distributions for two rules that participate in n_j: the rule $\underline{r}_j$ substituting to n_j, in which it is a right-hand-side nonterminal and the rule $\overline{r}_j$ in which n_j is on the left-hand-side. The authors then remark that the consistency can be measured roughly as the inner product of preference distributions for $\underline{r}_j$ and $\overline{r}_j$. But they immediately add that this actually inaccurate since rules can have multiple children, and the implicit labels for the left-hand-side and all child nonterminals are all dependent so that the computation cannot simply be factored like that over the child nonterminals. We want to formally capture the dependence between the label preferences of a rule and the preferences for all its child nonterminals. The computation of consistency and propagation of preferences might be better described by considering a substituting rule as an operator and the nonterminal children to which it substitutes as arguments of that operator. The mathematical operations that take place in the propagation of preferences are linear, yet generally more complex than simple vector or matrix products since multiple children can be involved. But it

[9]Note that this description and the descriptions that follow in this subsection work in probability space as opposed to log-probability space, as used for features in most other parts of the thesis. This is done to stay close to the original description of (Venugopal et al., 2009) and avoid unnecessary complication. In actual implementation conversion to log-probability space is necessary, but it is straightforward.

turns out that the more general framework of tensor products is adequate. Using this framework, every rule in a derivation has an associated tensor that models its preferences in a tensor-vector product interaction with the preference vectors of its nonterminal children. Binary rules have associated 3rd-order tensors, unary rules associated matrices, and lexical rules associated vectors. This formal framework was introduced for translation by the work on Latent-Variable SCFGs (Saluja et al., 2014) and is discussed in our overview of related work on labeling methods.[10] While this framework was not yet known to be useful for translation at the time of publication of (Venugopal et al., 2009), we believe it might in retrospect facilitate a more formal and high level description of what cannot be grasped at a high level as an inner products between vectors. While we try to make this connection where applicable, for simplicity we mostly stay close to the original description of (Venugopal et al., 2009) in our description.

Let $\mathcal{H}$ be the set of all hidden labels that can be assigned to nonterminals in unlabeled Hiero rules. Let $h \in \mathcal{H}$ be one specific hidden label. Furthermore let hargs(r) be the set of valid label assignments[11] of rules with nonzero preference probability. Every chart item that contains an explicit nonterminal symbol X (not S)) has a preference distribution over hidden labels:

$$u : \{h \in \mathcal{H}\} \rightarrow [0, 1] \qquad \text{(with} \sum_h u(h) = 1\text{)}$$

For chart items that have no nonterminal children the preference distribution is simply defined as the left-hand-side label preference distribution of its associated lexical rule:

$$u(h) = p_{pref}(h \mid \overline{r}_j) \tag{A.18}$$

In other cases the preference distribution is recursively defined based on the preference distribution of the added unary or binary rule and the preference distributions for the nonterminal children to which it substitutes. The rule's probabilities for the joint assignment of right-hand-side nonterminals are multiplied with the earlier computed

[10] Because in preference grammars, for every rule probabilities over different label assignments must sum to one, this puts restrictions on what tensors are permissible. Saluja et al. (2014) in contrast learn the tensors, and need not express probability distributions over label assignments for rules. This makes them more flexible to express for example that certain labels can be interchanged without penalty.

[11] A label assignment jointly assigns the labels for the left-hand-side nonterminal and all right-hand-side nonterminals.

$$
\begin{aligned}
\tilde{u}_{\mathrm{L}}(S) &= p_{pref}(\langle h = S, h' = VB\rangle \mid \underline{r})\, u(VB) + p_{pref}(\langle h = S, h' = VP\rangle \mid \underline{r})\, u(VP) \\
&= (0.5 \times 0.8) + (0.3 \times 0.2) = 0.46 \\
\tilde{u}_{\mathrm{L}}(S/PP) &= p_{pref}(\langle h = S/PP, h' = VB\rangle \mid \underline{r})\, u(VB) \\
&= (0.2 \times 0.2) = 0.04 \\
u_{\mathrm{L}} &= \langle u_{\mathrm{L}}(S) = 0.46/\tilde{u}_{\mathrm{L}}(S) + \tilde{u}_{\mathrm{L}}(S/PP), u_{\mathrm{L}}(S/PP) = 0.04/\tilde{u}_{\mathrm{L}}(S) + \tilde{u}_{\mathrm{L}}(S/PP)\rangle \\
&= \langle u_{\mathrm{L}}(S) = 0.46/50, u_{\mathrm{L}}(S/PP) = 0.04/0.50\rangle \\
\phi_2 &= u(VB) + u(VP) = 0.8 + 0.2 = 1
\end{aligned}
$$

Figure A.2: The calculation of u_{L} and ϕ_2 for the Dutch-English working example.

preferences for these nonterminals:

$$
\begin{aligned}
u_{\mathrm{L}}(h) &= \frac{\tilde{u}(h)}{\sum_{h'} \tilde{u}(h')} && \text{where} \\
\tilde{u}_{\mathrm{L}}(h) &= \sum_{h' \in \mathcal{H}: \langle h, h'\rangle \in \mathrm{hargs}(r)} p_{pref}(\langle h, h'\rangle | r) \times u(h') && \text{for unary rules} \\
\tilde{u}_{\mathrm{L}}(h) &= \sum_{\langle h', h''\rangle \in \mathcal{H}: \langle h, h', h''\rangle \in \mathrm{hargs}(r)} p_{pref}(\langle h, h', h''\rangle | r) \times u(h') \times u(h'') && \text{for binary rules}
\end{aligned}
\tag{A.19}
$$

Here we write the normalized/un-normalized left-hand-side preference values for new items as $u_{\mathrm{L}}/\tilde{u}_{\mathrm{L}}$, to avoid confusion with the preference distributions of the right-hand-side nonterminal children. These computations could be described using the more general framework of tensor products mentioned earlier.

Ultimately we need to compute the factors ϕ measuring the overlap in label preference between substituting rules and substituted for child nonterminals. This is done using the recursively defined chart item preferences $v(h)$. In case of lexical rules the consistency factor is simply defined to be 1. For unary rules the value is computed by determining the (hidden) child nonterminal labels with nonzero rule preference and summing the nonterminal preferences for these labels. In case of binary rules the computation is analogous, only now the nonterminal preferences for each pair of child nonterminal labels with nonzero rule preference is multiplied.

$$
\begin{aligned}
\phi &= \sum_{h' \in \mathcal{H}: \langle h, h'\rangle \in \mathrm{hargs}(r)} u(h') && \text{for unary rules} \\
\phi &= \sum_{\langle h', h''\rangle \in \mathcal{H}: \langle h, h', h''\rangle \in \mathrm{hargs}(r)} u(h') \times u(h'') && \text{for binary rules}
\end{aligned}
\tag{A.20}
$$

Applying this to the translation of "koks kunnen bakken" for our working example in Figure A.1, first and item is generated using the lexical rule producing *bake*. This

item gets a label preference distribution as given by the lexical rules preference for the LHS label, and a consistency factor of 1:

$$u = \langle u(VB) = 0.8, u(VP) = 0.1 \rangle \,, \qquad \phi_1 \;=\; 1$$

Next this lexical item is combined with the rule $X \;\rightarrow\;$ koks kunnen $X^{\boxed{1}}$ || cooks can $X^{\boxed{1}}$. This generates a new item with u_{L} and ϕ_2 whose values are computed with the recursive preference computation equations (A.19) and the consistency factor computation equations (A.20). These computations are shown in Figure A.2. We see that for this derivation producing the correct translation we get $\phi_2 = 1$. For the wrong derivation that produces "cooks can bins" in contrast $\phi_2 = 0$.

Implementation of preference grammars in the decoder

There are some important details concerning the actual implementation of preference grammars as discussed by (Venugopal et al., 2009). The first detail concerns the implementation in the decoder. A decoder with simple Hiero grammars has items/states in the chart of the form $[N, i, j, \tilde{e}]$ with $N \in \{X, S\}$ a nonterminal, $[i, j]$ the source span of the item and $\tilde{e}$ the language model state. But implementing preference grammars exactly as discussed in the previous subsection, adding preference distributions to chart items, effectively splits these states. This happens because left-hand-side preference distributions depend on the rule that was used to form an item. Such a state splitting is undesirable, as it increases the search space further, compromising the feasibility of efficient decoding. Therefore, rather splitting items by adding different label preferences v' for different derivations, all different derivations of a chart item $[N, i, j, \tilde{e}]$ share the value v for the *highest scoring* derivation of that chart item, and items need not be split further. The second important detail is that decoding is implemented in (Venugopal et al., 2009) as a two step process. In the first pass a hypergraph is computed without the preference-grammar feature $P_{syn}(d)$. In the second pass this hypergraph is refined by computing this feature and effectively rescoring the hypergraph. This is very similar to multi-pass parsing (Goodman, 1997) also known as coarse-to-fine parsing (Charniak, 2000; Charniak and Johnson, 2005) applied to translation. The last detail with regards to implementation is the preference distributions of both rules and items are pruned. The threshold β_R restricts the number of label assignments of rules to a limited number with the highest preference values. The threshold β_L is similar but applies only to lexical rules, and is typically stricter. Finally the threshold β_P limits the number of alternative label assignments per *item*. (Venugopal et al., 2009) experiments with various different values for these threshold: $\beta_L = 5, \beta_R \in \{10, 100\}, \beta_P \in \{1, 2, 5\}$.

Soft Matching Constraints

The work on preference grammars (Venugopal et al., 2009) softens the strict matching constraint of labeled hierarchical translation systems by performing a form of approximate summation over derivations that only differ in their labels. This leads to

finding the derivation class with the highest probability.[12] But Chiang (2010) notes that this approach still only includes derivations that satisfy the matching constraint, and proposes instead to soften the matching constraint itself. This is called in the literature decoding with *soft matching constraints*, or often just *soft constraints* or *fuzzy matching*.

Historical overview of soft constraint explorations The idea to use a form of soft constraints goes back at least to (Chiang, 2005). There a binary feature was added which fired if the source span of a rule matched a constituent in source parse of the input, allowing the decoder to learn to prefer translations that better match the source syntax. While the single feature used by Chiang (2005) yielded no improvements on the test set, the idea to use features marking rule application compatibility with source syntax was picked up by Marton and Resnik (2008) who refined it. They created more specific features for different types of source constituents. They used both features that marked the matching of specific constituent types by rule spans and also other features that marked their crossing by rule spans. We will refer to such features as *source syntax compatibility features*. The authors used minimum error rate training as tuning method, which restricted them in how much features they could successfully add at a time. They solved this problem by first testing many features separately and then combining only the most successful ones. Using this approach they obtained significant improvements over Hiero for Chinese–English and Arabic–English translation.

The work on source syntax compatibility features was further extended upon by (Chiang et al., 2008). The latter upgraded the tuning method to MIRA (Crammer and Singer, 2003), which allowed the successful training of the weights of much more features. It also added structural distortion features which mark reordering or no reordering by rules within the context of specific source span lengths for rule applications. Continuing in this line (Chiang et al., 2009), further extended the feature set. It kept internal word alignments associated with rules and used these to add sparse features marking translations of frequent words within the context of other frequent words on the left or right side. Additionally, it used soft constraints on the target side in string-to-tree translation. In this setting features were added to mark specific left-hand-side labels on the target side, and the occurrence of a combination of nonterminals in a rule. Other features counted the frequency of nonterminals of different types or marked the usage of rules that insert words on the target side without having words in the source. Both the source syntax compatibility features and target side features have similar roles in learning and encouraging the type of syntactic structure that improves translation quality.

Inter-rule soft matching constraints A crucial limitation of the soft constraints as mentioned in the previous discussion is that they are confined to single rules and their

[12] Just as in Hiero, the same translations can still be formed by alternative derivations that segment the input in different ways. Therefore just as in Hiero the translation with highest probability is not found, only the most likely derivation class.

context. They do not mark (inter-rule) substitutions of nonterminal X for nonterminal Y. Features marking specific substitutions were only first explored in Chiang (2010), and amongst the soft constraints we mentioned only this type has much similarity with preference grammars. This approach is also most similar to the approach we use in chapter 5.

The soft constraints translation approach proposed by (Chiang, 2010) which we will refer to as *soft source and target syntactic constraints* (SSTSC) uses a grammar with rules that are labeled on source and target side. Let $X = \langle X_f, X_e \rangle$ be a linked label pair (nonterminal pair) with source side label X_f and target side label X_e. Similarly let $Y = \langle Y_f, Y_e \rangle$ be another linked nonterminal pair with source side label Y_f and target side label Y_e. SSTSC allows any label pair[13] X to substitute for any other nonterminal pair Y. But rather than enforcing strict matching of substituting to substituted for source and target sides of these nonterminal pairs, features are added that mark matching and mismatching substitutions of for the source and target side of these label pairs. Additionally features are added that mark specific substitutions X_f to Y_f on the source side and X_e to Y_e on the target side. The complete set of soft syntactic constraint features that is used is as follows:

- $match^f$: counts the rule applications where substituting labels and substituted for labels match on the source side.
- $\neg match^f$: similarly counts the number of mismatching source side substitutions.
- $subst^f_{X \to Y}$ counts the number of specific substitutions of label X to label Y on the source side.
- $match^e$, $\neg match^e$, $subst^e_{X \to Y}$: these are the analogous features for the target side.
- $root_{X,X'}$: counts the number of rules with root label X on the source side and root label X' on the target side.

Bilingual syntactic labels The labels used in (Chiang, 2010) deserve some attention. These labels are a generalization of both SAMT labels (Zollmann and Venugopal, 2006) and synchronous tree-sequence-substitution grammar (STSSG) (Zhang et al., 2008b), the latter combining an (arbitrary) sequence of trees. The resulting labeling method allows sequences $X_1 + \cdots + X_n$ of any length, and allows the slash operator to take any number of arguments on either side.[14] The gain of this approach is that any

[13] With the exception of the start symbol $\langle S, S \rangle$.

[14] This description as given in (Chiang, 2010) still leaves ambiguity with respect to the exact implementation, as especially allowing both combination of labels with the + operator and "subtraction" of labels with the CCG "/" and "\" operator can yield different alternative labels for the same span. What is missing in the description is how the final label is chosen in case of multiple possible alternatives. We might assume that a strategy like the one in SAMT is used, preferring the "simplest" possible label under some definition of simplicity. One sensible criterion for simplicity could be the amount of sub-labels/parts are involved in the label. In the same way SAMT prefers the label A/B over $A + B + C$, since the latter has more parts and is therefore more complex and also likely more sparse.

span on the source or target side can be assigned a label. The price is that some of these labels will be very complex and rare.

Canonical labeled rules Decoding with soft constraints is conceptually simple and with the help of MIRA (Crammer and Singer, 2003) training label substitutions for even big label sets is feasible. One concern is that if there are many different labeled copies of the same HIERO rule, this will lead to additional complexity problems and more spurious ambiguity in combination with soft constraints decoding. The reason is that all these copies of the same rule with different labels can substitute anywhere, independent of the labels, which is not true of course in a normal strict matching setup. For this reason (Chiang, 2010) proposes the measure of using just one labeled version per HIERO rule type. This "*canonically labeled*" rule is chosen as the labeled version that occurs most frequent in the training data. Using this measure assures that the number of rules remains equal the number in original HIERO.

Relative decoding complexity Chiang (2010) points out that if canonical labeled rules are used this, assures that using soft matching will not substantially increase decoding times in comparison to HIERO, provided the cube pruning pop-limit is kept the same in both settings. This might not seem intuitive at first. While clearly the number of rules does not change, in a setting with fuzzy matching every rule can match to about ever other label.[15] And since these matchings for different labels are different they need to be distinguished. It would seem then that considerably more different distinguishable rule matchings take place than in standard HIERO. But looking deeper into the matter it becomes clear that rules in fact combine with *items*, which include language model state in addition to labels. This number of items, not the number of different labels, determines the complexity.[16]

And keeping the cube pruning pop-limit the same, assures this number will not change, at least not in cases where the number of items per chart entry reaches the pop limit for both systems (which is relatively common). Of course having more labels, does increase the search space, so with the same pop-limit the relative chance for search

[15]Except for the start/goal symbol and the glue rule label (X).

[16]This is the case at least in an optimal implementation of fuzzy matching in the decoder. One tricky part is that many decoders, including Joshua (Ganitkevitch et al., 2012) were not build with fuzzy matching in mind. They must be adapted in a way that guarantees that soft matching does not slow down performance unnecessarily. Most decoders (including Joshua) use CYK+ parsing (Chappelier and Rajman, 1998), which uses a so called *DotChart* to keep track of matching, partially completed rules. With strict matching, for a certain span, there is only one possible matching of a rule containing nonterminals to already proven labels. But with soft constraints there are many alternative labels that can be matched to a single rule. When using a DotChart, to avoid an explosion in memory usage, and to a lesser extend computation, it is crucial to match just one of the alternative labels rather than matching all of them separately, and representing their matchings separately. Instead a compact representation of a matchable rule and its alternatives for substituted-for nonterminal labels should be created, the latter which can be trivially created from the available labels. As a last remark on this issue, we note that it may be best to avoid using a DotChart altogether, which is possible with the proposal of (Sennrich, 2014).

errors increases in comparison to HIERO for the soft matching system. Nevertheless, even taking into account this increased risk of search errors, as Chiang (2010) remarks, the net advantage of the soft constraints—even with the same pop-limit—is substantial.

Further complexity reduction by restriction to nesting phrases

In order to further reduce computational complexity, (Chiang, 2010) proposes a method to select a limited collection of *nesting* phrases per sentence pair, giving preference to those phrases which are most in agreement with source and target syntax. Two phrases, with source words $w_i \cdots w_j$ and $w_{i'} \cdots w_{j'}$ are defined to be nesting, as opposed to *crossing*, if one is properly contained inside the other. Formally $i \leq i' < j' \leq j$ or $i' \leq i < j \leq j'$. For every sentence pair, different sets of nesting phrases are possible. To determine which set to use, (Chiang, 2010) proposes a heuristic, greedy method that at each iteration adds the most syntax-consistent phrase that is nesting with the phrases that were already added before.

The level of "agreement with syntax" is itself implemented as a heuristic (full) ordering over phrases. This order first sorts on how many of the two labels, one for the source and one for the target side, correspond to syntactic constituents (both, then one, then zero). Ties are further sorted by how many syntactic constituents $\overline{f}$ and $\overline{e}$ cross. There are two more such criteria to further distinguish in case of ties, we refer to (Chiang, 2010) for the complete scheme.

Here we notice that although on a sentence level nesting phrases are selected, this does not imply that the rules extracted over the entire training set will nest when recombined for the translation of new sentences. In fact it is likely that the derivations produced by the decoder using these rules still include many alternative segmentations of the input. Nevertheless a substantial reduction of the amount of rules is achieved by this technique, e.g. going doing from 440M rules without nesting to 180M rules with matching on Chinese–English translation. This reduces memory requirements and speeds up decoding, while interestingly not leading to a loss in performance.[17]

A.5 Latent Variable Synchronous CCFGs for Hierarchical SMT

Learning latent variable tensors The dense latent variable tensors are learned from sparse features that mark the context of rules in the training set. The features that are

[17] One might believe that the reduced search-space as a result of using nesting gives an advantage that weights off against the other advantage of using more different phrases. The former advantage is only valid if approximate search using pruning is done. It is therefore plausible that a system allowing all phrases would still be superior if exhaustive search were done. But that is practically infeasible as it would take forever to translate even a single sentence. And essentially automated translation nearly always involves a tradeoff between affordable time (and memory) costs and quality. Hence if we accept the implicit assumption of (Chiang, 2010) that a pop-limit assuring reasonable decoding times is required, the argument that no performance is lost is valid.

used mark the inside context of rules as the identity of their right-hand-side children. The outside context is also marked as the identity of the parent rule and sibling rule, if it exists. Using these rule indicator features as well as more features encoding context by means of lexical and length descriptions, a rich description of rule contexts is formed represented as high dimensional and sparse feature vectors.

Learning of the latent variable tensors is based on the covariances between the sparse feature vectors for the inside and outside trees of rules in the training corpus, encoded as an empirical covariance matrix $\hat{\Omega}$. This matrix encodes how the features of inside trees predict the features of outside trees and vice versa. Next, $\hat{\Omega}$ is mapped to a lower dimensional space that is chosen so as to retain maximum ability to accurately predict inside trees from outside trees, as these were represented by the original sparse feature representation. Let O be the set of pairs of all inside-outside trees $\langle o, t \rangle$ is collected from the corpus. Let $\phi(t) \in \mathbb{R}^{\mathrm{d}\times 1}, \psi(\mathrm{o}) \in \mathbb{R}^{\mathrm{d}'\times 1}$ be the inside and outside features for these pairs. Then for each vector pair the outer product is computed, and aggregated to obtain an inside-outside covariance matrix:

$$\hat{\Omega} = \frac{1}{|O|} \sum_{\langle o,t\rangle \in O} \phi(t)(\psi(o))^{\top} \tag{A.21}$$

This empirical covariance matrix can be factored by means of a singular value decomposition (SVD), and then compressed by keeping only the m largest singular values, projecting all feature vectors to a much smaller m-dimensional space. This can be formally expressed as $\hat{\Omega} = U\Sigma V^{\top} \approx U'\Sigma' V'^{\top}$, with $U' \in \mathbb{R}^{\mathrm{d}\times \mathrm{m}}$ and $V' \in \mathbb{R}^{\mathrm{d}'\times \mathrm{m}}$ being trunctuated matrices containing the most significant left and right singular vectors, and $\Sigma \in \mathbb{R}^{\mathrm{m}\times \mathrm{m}}$ being a diagonal matrix with the m largest singular values on the diagonal. This compression by projection to an m-dimensional space with much lower dimensionality is the basis of the computation of latent variables by means of the spectral method, as presented in (Cohen et al., 2013) for PCFGs and generalized in this work to SCFGs. The remaining steps proceed as follows:

1. The base vectors computed by the covariance matrix compression are used to project the high dimensional vector representations of inside and outside trees to m-dimensional (dense) vector representations.

2. Rule-specific covariances are computed between the compressed vector representations of outside trees and inside trees, for rules and their gaps. These covariances are computed using the *tensor product*, which generalizes the outer product to allow computation of covariances between more than two random vectors. For each rule, the covariance tensor is computed by averaging over all covariance tensors for different occurrences of the rule.

3. For the start rule the final tensor is computed as the average dense inside tree representation over all root rules in the training set. For every other rule, its final tensor is computed by scaling its average tensor from step 2 with the relative frequency of the rule over the entire training corpus.

Instead of using the spectral method for learning, Expectation Maximization (EM) (Dempster et al., 1977) can be applied as well. But in combination with the tensor representation of latent variables, EM yields a more complicated and slower learning procedure and also produces worse end results than the spectral method. We refer the interested reader to the original paper (Saluja et al., 2014) for more details about the learning with the spectral method and the EM algorithm.

Results The work is evaluated on Chinese–English translation and achieves big and significant improvements over baselines with minimal rules but without the latent-variable based feature re-scoring. A much smaller, yet still significant improvement of +0.5 BLEU over the HIERO baseline is also made. This is explained by the initial relative loss of 2.5 BLEU points performance by working with minimal rules, in line with what was reported earlier by (M.Galley et al., 2006) on this matter.

A.6 Chinese Data Preparation

One complication of the *HongKong Parallel Text* data is that the Chinese side is written in the *traditional Chinese* script which is used in Hong Kong in contrast to the *simplified Chinese* script which is the default in mainland China and which is used for all other datasets.[18]

The *simplified Chinese* script, which has about 8K characters is as the name suggests a simplification of the *traditional Chinese* script, which has more than 80K characters[19]. In principal, a one-to-one mapping from *traditional* to *simplified Chinese* is possible, and we perform this mapping using a simple conversion program based on a character lookup table[20].

The Chinese data needs to be segmented, and the quality of the segmentation can have a considerable effect on the translation quality. We use the Sanford Segmenter(Chang et al., 2008) which is a common choice in the field. The segmenter has in fact two models available; one trained using data and conventions from the Chinese Penn Treebank, the other using data and conventions of the Peking University Standard (Beijing University). We use the Chinese Treebank (ctb) model, the difference between the two segmentation models for the purpose of translation seems small both according to their resulting vocabulary sizes and also based on large scale empirical studies where for some data sets the first model (ctb) performs better and for others the latter (pku), see (Wang et al., 2014).

Finally empirical analysis revealed that both the *MultiUn*(Eisele and Chen, 2010; Tiedemann, 2012) data and the *HongKong Parallel Text* data from the Linguistic Data

[18] Another practical complication is that *HongKong Parallel Text* uses all kinds of different encodings, Big5 for most of the Chinese text being one of them, so everything must first be converted to UTF8 to work with it.

[19] It is hard to give exact numbers, as these are subject to change.

[20] Downloaded from http://www.mandarintools.com/zhcode.html

Consortium contained significant amounts of duplication, in which identical triples of source, sentence and alignment appeared multiple times. Removing these duplicates gave a considerable speed-up to grammar extraction while having no negative effects on performance.

A.7 Grammar Merging

For certain language pairs and training sets, labeled grammars with a big label set—in particular SAMT— can become too big to fit in memory during grammar extraction, even while filtering rules based on the sentences in the test set. In the case of Chinese–English translation, with a multiple million sentence pair training set, and a test set of approximately 2K sentences, extracting the SAMT grammar as a whole gives problems. This is a difficult technical problem, that can only be solved completely by implementing a new grammar extractor that uses disk space instead of memory to store the grammar rules. But this solution is not so desirable as it would require a lot of extra re-implementation work, just to solve a purely technical issue that only affects grammar extraction in rare scenarios. This motivated us to use a much simpler solution, which is to split the test set into parts, extract a *partial* filtered grammar for each of these parts, and then merge these partial grammars to get the end results. In the case of Chinese–English translation with SAMT, splitting the test set into 4 parts is enough to solve the memory problems.

Merging the filtered grammars When merging the grammars from the different parts, it must be noted that many rules will occur in the filtered grammars for multiple parts. To deal with this situation, we concatenate all the filtered grammars and then sort the resulting file which now contains many duplicates. Finally, going over the sorted merged file, we simply take the first occurrence of every grammar rule type with its feature values and use that occurrence in the final merged grammar. A consequence of this approach is that the feature values of the merged grammar are not exactly the same as would be the feature values for the filtered grammar when it would be extracted as a whole.

The question is if the changes in feature values as a result of the merging scheme are big and likely to have an impact on translation performance. To answer this question, remark that arguably the most important features, the phrase translation probabilities also known as phrase weights, are not affected at all. These features are $p(s|t)$ and $p(t|s)$, plus optionally all their smoothed variants in the case of labeled grammars. They are computed using relative frequency estimation based on counts that are "local" to the rule type. This means that for every rule type these counts are the same in every filtered grammar where it occurs. Therefore it makes no difference for these features whether they are taken from the grammar as computed as a whole or from one of the partial filtered grammars in our merging scheme.

This situation is different for the generative probability of a rule r:

$$p(r|LHS(r)) = \frac{count(r)}{\sum_{r`\in Rules \wedge (LHS(r')=LHS(r)} count(rule))} \tag{A.22}$$

The reason is that in this feature the normalization factor, which is the sum of counts of all rules with the same left-hand-side, will be different in partial grammars and in grammars that are computed in one go. The same argument also holds for the rarity penalty feature, which also depends on the total rule count, see section 5.4.1.

There are at least three reasons to believe that the differences will be neither very big nor have a real influence on translation performance.

The first reason is the form of he distribution over rule counts. This is a Zipfian distribution, similar to the type of Zipfian distributions over tree fragments used in parsing with tree-substitution grammars. This means that as the size of the set of sentences used to filter the grammar grows, the amount of new rule types grows steadily (at most linearly) with it, but the rate of increase of rule tokens quickly drops of. The reason for this is that the most common rules make up most of the total rule count, so that once all common rules have been included the total rule count will only increase slowly.

The second reason is that any differences in absolute feature values can be compensated for by the tuner, given that the relative values are mostly stable. The absolute magnitude of the generative probabilities and rarity penalty features may vary depending on the size of the set of sentences used for filtering during grammar extraction (*filter set*). But the relative magnitude or ranking of these feature values will remain mostly stable.[21] This means that adapting the weights of these features during tuning should give sufficient compensation of any scaling effects as a result of differently sized filter sets. Therefore the tuner can compensate sufficiently to cancel out any relative differences in absolute feature values that result from varying total feature counts. The only requirement is that the same grammar extraction method is used for both development and test grammar creation, and that the filter sets in both cases are of similar size.[22]

A final reason to think that the absolute values of certain features should not have a big effect on final translation performance we mention the work by Lopez (2008).

[21] Between different partial grammars there will be marginally different total rule counts for different left-hand-sides. These total rule counts will also typically differ from the total rule count when extracting the filtered grammar as a whole. But the total rule counts are mostly determined by the most common rule types, which are shared across the different partial grammars. Therefore there should be only very small differences in the values of total rule counts between the partial grammars. This means that the feature values are more or less scaled uniformly across the partial grammars, and relative feature values will not change much in comparison to extraction of the entire filtered grammar as a whole.

[22] Remark that using a very differently sized filter set for development and testing is generally a bad idea, since this will lead to a suboptimal weighting for those features whose absolute values depend on total rule counts, which vary with the size of the filter sets.

His work shows that even when grammar features are computed by sampling, when producing input-filtered grammars on the fly for fast online hierarchical statistical machine translation, the translation results do not seam to suffer much. In personal correspondence with Adam Lopez, discussing the difference between his grammar extraction and feature estimation method by (Chiang, 2005) he remarked that:

"

1. Which estimate is most effective is an empirical question.
2. Most of the estimates used in the literature, heuristic or otherwise, yield very similar performance because of (3)
3. The estimators (even the sound ones) are used with models that make a lot of assumptions that simply aren't true in the data, which are in any case very noisy."

A similar way to look at this is that almost all the features that are used in hierarchical SMT are heuristic and somewhat ad-hoc to begin with. Therefore, adding one more level of approximation or variation to the computation of these features will not change the result a lot, provided that it leaves the general characteristics of the features intact, so that tuning can effectively compensate for the absolute differences in feature values.

To summarize we noted that:

1. Only the generative rule probability and rarity penalty change as a result of varying the filter set, and for these only the absolute and not the relative values are changed.
2. Tuning can compensate for any differences in the absolute values of the affected features by adapting the feature weights.
3. Earlier work by Lopez (2008) showed that the influence on the details of the exact computation of the feature estimates is small to begin with.

For these reasons we conclude that although for certain features our grammar merging scheme leads to differences in the absolute feature values, on the final translation task it can be expected to give very similar performance to grammar extraction as a whole in one step.

Bibliography

Aho, A. V. and Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci.*, 3(1):37–56.

Albert, M. H., Atkinson, M. D., and Klazar, M. (2005). The enumeration of simple permutations. *Journal of Integer Sequences*, 6(03.4.4).

Almaghout, H., Jiang, J., and Way, A. (2010). Ccg augmented hierarchical phrase-based machine translation. In Federico, M., Lane, I., Paul, M., and Yvon, F., editors, *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 211–218.

Almaghout, H., Jiang, J., and Way, A. (2011). Ccg contextual labels in hierarchical phrase-based smt. In *Proceedings of the 15th Annual Conference of the European Association for Machine Translation (EAMT-2011)*.

Almaghout, H., Jiang, J., and Way, A. (2012). Extending ccg-based syntactic constraints in hierarchical phrase-based SMT. In *Proceedings of the Annual Conference of the European Association for Machine Translation (EAMT)*.

Ambati, V., Lavie, A., and Carbonell, J. (2009). Extraction of syntactic translation models from parallel data using syntax from source and target languages. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Arnold, D. and Tombe, L. D. (1987). Basic theory and methodology in EUROTRA. In Nirenburg, S., editor, *Machine Translation: Theoretical and Methodological Issues*. Cambridge University Press, Cambridge, UK.

Auli, M., Galley, M., Quirk, C., and Zweig, G. (2013). Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA. Association for Computational Linguistics.

Ayan, N. and Dorr, B. (2006). Going beyond AER: an extensive analysis of word alignments and their impact on MT. In *Proc. of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 9–16, Morristown, NJ, USA.

Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(2):179–190.

Baker, J. K. (1979). Trainable grammars for speech recognition. In Klatt, D. H. and Wolf, J. J., editors, *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550.

Bangalore, S., Haffner, P., and Kanthak, S. (2007). Statistical machine translation through global lexical selection and sentence reconstruction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 152–159, Prague, Czech Republic. Association for Computational Linguistics.

Bangalore, S. and Joshi, A. K. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Bar-Hillel, Y., Perles, M., and Shamir, E. (1961). On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bennett, W. S. and Slocum, J. (1985). The lrc machine translation system. *Comput. Linguist.*, 11(2-3):111–121.

Berger, A. L., Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Gillett, J. R., Lafferty, J. D., Mercer, R. L., Printz, H., and Ureš, L. (1994). The candide system for machine translation. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 157–162, Stroudsburg, PA, USA. Association for Computational Linguistics.

Berger, A. L., Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Kehler, A. S., and Mercer, R. L. (1996a). Language translation apparatus and methods using context-based translation models. US Patent 5,510,981.

Berger, A. L., Della Pietra, S. A., and Della Pietra, V. J. (1996b). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–72.

Birch, A. and Osborne, M. (2010). LRscore for evaluating lexical and reordering quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332, Uppsala, Sweden. Association for Computational Linguistics.

Birch, A., Osborne, M., and Blunsom, P. (2010). Metrics for MT evaluation: Evaluating reordering. *Machine Translation*, 24(1):15–26.

Blunsom, P., Cohn, T., Dyer, C., and Osborne, M. (2009). A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 782–790.

Blunsom, P. and Osborne, M. (2008). Probabilistic inference for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 215–223, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bod, R. (1992). A computational model of language performance: Data oriented parsing. In *COLING 1992 Volume 3: The 15th International Conference on Computational Linguistics*, pages 855–859.

Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. CLSI Publications, Stanford University, CA.

Bod, R. (2001). What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 66–73.

Bod, R. (2003). An efficient implementation of a new dop model. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 19–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bod, R., Scha, R., and Sima'an, K. (2003). *Data-oriented Parsing*. Center for the Study of Language and Information - Lecture Notes Series. CSLI Publications.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Lechevallier, Y. and Saporta, G., editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France. Springer.

Brown, P., Cocke, J., Pietra, S. D., Pietra, V. D., Jelinek, F., Mercer, R., and Roossin, P. (1988). A statistical approach to language translation. In *Proceedings of the 12th Conference on Computational Linguistics - Volume 1*, COLING '88, pages 71–76, Stroudsburg, PA, USA. Association for Computational Linguistics.

Brown, P., Pietra, V. D., Pietra, S. D., and Mercer, R. (1993). The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

Burkett, D., Blitzer, J., , and Klein, D. (2010). Joint parsing and alignment with weakly synchronized grammars. In *Proc. of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 127–135.

Cer, D., Jurafsky, D., and Manning, C. D. (2008). Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 26–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chang, P.-C., Galley, M., and Manning, C. D. (2008). Optimizing chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 224–232, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chappelier, J.-C. and Rajman, M. (1998). A generalized cyk algorithm for parsing stochastic cfg. In *TAPD*, pages 133–137.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 132–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Charniak, E. (2001). Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 124–131, Stroudsburg, PA, USA. Association for Computational Linguistics.

Charniak, E. and Charniak, E. (1996). Tree-bank grammars. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chelba, C. and Jelinek, F. (2000). Structured language modeling. *Computer Speech and Language*, 14:283–332.

Chen, S. F. and Goodman, J. T. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University.

Cherry, C. (2008). Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 72–80, Columbus, Ohio. Association for Computational Linguistics.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *HLT-NAACL*, pages 427–436.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 263–270.

Chiang, D. (2006). An introduction to synchronous grammars. Tutorials at the Annual Meeting of the Association for Compuational Linguistics (ACL).

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chiang, D. (2010). Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1443–1452, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chiang, D. (2012). Hope and fear for discriminative training of statistical translation models. *J. Mach. Learn. Res.*, 13(1):1159–1187.

Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 218–226, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 224–233, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Clark, S. and Curran, J. R. (2004). The importance of supertagging for wide-coverage ccg parsing. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cocke, J. (1969). *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2013). Experiments with spectral learning of latent-variable pcfgs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157, Atlanta, Georgia. Association for Computational Linguistics.

Cohn, T. and Blunsom, P. (2009). A bayesian model of syntax-directed tree to string grammar induction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 352–361.

Cohn, T., Blunsom, P., and Goldwater, S. (2010). Inducing tree-substitution grammars. *Journal of Machine Learning Research*, pages 3053–3096.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Crammer, K., Kulesza, A., and Dredze, M. (2009). Adaptive regularization of weight vectors. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 414–422. Curran Associates, Inc.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38.

DeNeefe, S., Knight, K., Wang, W., and Marcu, D. (2007). What can syntax-based MT learn from phrase-based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 755–763.

DeNero, J., Gillick, D., Zhang, J., and Klein, D. (2006). Why generative phrase models underperform surface heuristics. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 31–38, Stroudsburg, PA, USA. Association for Computational Linguistics.

Denkowski, M. and Lavie, A. (2011). Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 85–91, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271.

Dixon, W. J. and Mood, A. M. (1946). The statistical sign test. *Journal of the American Statistical Association*, 41(236):557–566.

Dorna, M., Frank, A., van Genabith, J., and Emele, M. C. (1998). Syntactic and semantic ransf er with f-structures. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 341–347, Montreal, Quebec, Canada. Association for Computational Linguistics.

Dorr, B. J. (1994). Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4):597–633.

Duh, K. and Kirchhoff, K. (2008). Beyond log-linear models: Boosted minimum error rate training for n-best re-ranking. In *Proceedings of ACL-08: HLT, Short Papers*, pages 37–40, Columbus, Ohio. Association for Computational Linguistics.

Duong, L., Cohn, T., Bird, S., and Cook, P. (2015). A neural network model for low-resource universal dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 339–348, Lisbon, Portugal. Association for Computational Linguistics.

Dyer, C. (2010). Two monolingual parses are better than one (synchronous parse). In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 263–266, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dyer, C., Weese, J., Setiawan, H., Lopez, A., Ture, F., Eidelman, V., Ganitkevitch, J., Blunsom, P., and Resnik, P. (2010). Cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*.

Eidelman, V. (2012). Optimization strategies for online large-margin learning in machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine*

Translation, pages 480–489, Montréal, Canada. Association for Computational Linguistics.

Eisele, A. and Chen, Y. (2010). MultiUN: A multilingual corpus from United Nation documents. In Tapias, D., Rosner, M., Piperidis, S., Odjik, J., Mariani, J., Maegaard, B., Choukri, K., and Calzolari, N., editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA).

Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, pages 205–208.

Fox, H. J. (2002). Phrasal cohesion and statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, Proceedings of EMNLP, pages 304–311, Stroudsburg, PA, USA. Association for Computational Linguistics.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296.

Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proc. of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 273–280.

Galley, M. and Manning, C. D. (2008). A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 847–855, Honolulu, Hawaii. Association for Computational Linguistics.

Ganitkevitch, J., Cao, Y., Weese, J., Post, M., and Callison-Burch, C. (2012). Joshua 4.0: Packing, pro, and paraphrases. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 283–291, Montréal, Canada. Association for Computational Linguistics.

García-Varea, I., Casacuberta, F., and Ney, H. (1998). An iterative, dp-based search algorithm for statistical machine translation.

Garmash, E. and Monz, C. (2014). Dependency-based bilingual language models for reordering in statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1689–1700, Doha, Qatar. Association for Computational Linguistics.

Garmash, E. and Monz, C. (2015). Bilingual structured language models for statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods*

in Natural Language Processing, pages 2398–2408, Lisbon, Portugal. Association for Computational Linguistics.

Garrette, D., Mielens, J., and Baldridge, J. (2013). Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 583–592, Sofia, Bulgaria. Association for Computational Linguistics.

Germann, U., Jahr, M., Knight, K., Marcu, D., and Yamada, K. (2001). Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 228–235, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gildea, D. (2003). Loosely tree-based alignment for machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 80–87.

Gildea, D., Satta, G., and Zhang, H. (2006). Factoring synchronous grammars by sorting. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 279–286, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gimpel, K. and Smith, N. A. (2012). Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 221–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Goodman, J. (1997). Global thresholding and multiple-pass parsing. In *Second Conference on Empirical Methods in Natural Language Processing*, pages 11–25.

Graca, J., Pardal, J., Coheur, L., and Caseiro, D. (2008). Building a golden collection of parallel multi-language word alignment. In *LREC'08*, Marrakech, Morocco. European Language Resources Association (ELRA).

Haghighi, A., Blitzer, J., DeNero, J., and Klein, D. (2009). Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 923–931.

Hanneman, G. and Lavie, A. (2011). Automatic category label coarsening for syntax-based machine translation. In *Proceedings of the Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST-5, pages 98–106, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hanneman, G. and Lavie, A. (2013). Improving syntax-augmented machine translation by coarsening the label set. In *HLT-NAACL*, pages 288–297.

Hasan, S., Ganitkevitch, J., Ney, H., and Andrés-Ferrer, J. (2008). Triplet lexicon models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 372–381, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hassan, H., Hearne, M., Sima'an, K., and Way, A. (2006). Syntactic phrase-based statistical machine translation. In *Proceedings of IEEE/ACL first International Workshop on Spoken Language Technology (SLT)*, pages 238–241.

Hassan, H., Sima'an, K., and Way, A. (2007). Supertagged phrase-based statistical machine translation. In *Proceedings of ACL 2007*, page 288–295.

Hassan, H., Sima'an, K., and Way, A. (2009). A syntactified direct translation model with linear-time decoding. In *In Proceedings of the Conference on Empirircal Methos in NLP (EMNLP'09)*, page 1182–1191.

Heafield, K. (2011). KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Heafield, K. (2013). *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation*. PhD thesis, Carnegie Mellon University.

Hildreth, C. (1957). A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85.

Hillel, Y. B. (1953). A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 29(1):47–58.

Hopkins, M. and Kuhn, J. (2007). Machine translation as tree labeling. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 41–48, Rochester, New York. Association for Computational Linguistics.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Stroudsburg, PA, USA. Association for Computational Linguistics.

Huang, B. and Knight, K. (2006). Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 240–247, New York City, USA. Association for Computational Linguistics.

Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In McKeown, K., Moore, J. D., Teufel, S., Allan, J., and Furui, S., editors, *ACL*, pages 586–594. The Association for Computer Linguistics.

Huang, L. and Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151. Association for Computational Linguistics.

Huang, L., Knight, K., and Joshi, A. (2006). A syntax-directed translator with extended domain of locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 1–8, New York City, New York. Association for Computational Linguistics.

Huang, L., Zhang, H., and Gildea, D. (2005). Machine translation as lexicalized parsing with hooks. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 65–73.

Huang, L., Zhang, H., Gildea, D., and Knight, K. (2009). Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

Huck, M., Hoang, H., and Koehn, P. (2014). Preference grammars and soft syntactic constraints for ghkm syntax-based statistical machine translation. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 148–156, Doha, Qatar. Association for Computational Linguistics.

Huck, M., Peitz, S., Freitag, M., and Ney, H. (2012). Discriminative reordering extensions for hierarchical phrase-based machine translation. In *In Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT*, pages 313–320.

Huck, M., Wuebker, J., Rietig, F., and Ney, H. (2013). A phrase orientation model for hierarchical machine translation. In *ACL 2013 Eighth Workshop on Statistical Machine Translation*, pages 452–463, Sofia, Bulgaria.

Imamura, K., Okuma, H., and Sumita, E. (2005). Practical approach to syntax-based statistical machine translation. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, Phuket, Thailand.

Irvine, A. and Callison-Burch, C. (2013). Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 262–270, Sofia, Bulgaria. Association for Computational Linguistics.

Isozaki, H., Sudoh, K., Tsukada, H., and Duh, K. (2010). Head finalization: A simple reordering rule for sov languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 244–251, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ittycheriah, A. and Roukos, S. (2007). Direct translation model 2. In *In HLT-NAACL 2007: Main Conference*, pages 57–64.

Johnson, R., King, M., and des Tombe, L. (1985). Eurotra: A multilingual system under development. *Comput. Linguist.*, 11(2-3):155–169.

Joshi, A. K. (1985). Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In Dowty, D. R., Karttunen, L., and Zwicky, A. M., editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 206–250. Cambridge University Press, Cambridge.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163.

Joshi, A. K. and Schabes, Y. (1997). Tree-adjoining grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer-Verlag New York, Inc., New York, NY, USA.

Kaeshammer, M. (2013). Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77.

Kaeshammer, M. (2015). Hierarchical machine translation with discontinuous phrases. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 228–238, Lisbon, Portugal. Association for Computational Linguistics.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Kaplan, R. M., Netter, K., Wedekind, J., and Zaenen, A. (1989). Translation by structural correspondences. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 272–281.

Kasami, T. (1965). An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, Bedford, MA.

Kim, Y.-B., Snyder, B., and Sarikaya, R. (2015). Part-of-speech taggers for low-resource languages using cca features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302, Lisbon, Portugal. Association for Computational Linguistics.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*, pages 79–86. AAMT, AAMT.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P., Axelrod, A., Birch Mayne, A., Callison-Burch, C., Osborne, M., and Talbot, D. (2005). Edinburgh system description for the 2005 iwslt speech translation evaluation. In *In Proceedings of the International Workshop on Spoken Language Translation (IWSLT*, pages 68–75.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kumar, S. and Byrne, W. (2004). Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL*, page 169–17.

Lang, B. (1988). Parsing incomplete sentences. In *In Proc. of the 12th International Conference on Computational Linguistics*, pages 365–371.

Langlais, P. and Gotti, F. (2006). Phrase-based SMT with shallow tree-phrases. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 39–46, New York City. Association for Computational Linguistics.

Lari, K. and Young, S. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

Lewis, II, P. M. and Stearns, R. E. (1968). Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488.

Li, J., Tu, Z., Zhou, G., and van Genabith, J. (2012a). Head-driven hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 33–37, Stroudsburg, PA, USA. Association for Computational Linguistics.

Li, J., Tu, Z., Zhou, G., and van Genabith, J. (2012b). Using syntactic head information in hierarchical phrase-based translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 232–242, Montréal, Canada. Association for Computational Linguistics.

Li, Z. and Eisner, J. (2009). First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Singapore. Association for Computational Linguistics.

Li, Z., Eisner, J., and Khudanpur, S. (2009). Variational decoding for statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 593–601, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. (2006a). An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 761–768, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liang, P., Taskar, B., and Klein, D. (2006b). Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 104–111.

Liu, D. and Gildea, D. (2008). Improved tree-to-string transducer for machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 62–69, Columbus, Ohio. Association for Computational Linguistics.

Liu, Y., Liu, Q., and Lin, S. (2006). Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 609–616, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liu, Y., Lü, Y., and Liu, Q. (2009). Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*

of the AFNLP, pages 558–566, Suntec, Singapore. Association for Computational Linguistics.

Locke, W. N. (1955). *Machine Translation of Languages: fourteen essays*. Technology Press of the Massachusetts Institute of Technology and Wiley.

Lopez, A. D. (2008). *Machine Translation by Pattern Matching*. PhD thesis, University of Maryland at College Park.

Macháček, M. and Bojar, O. (2013). Results of the WMT13 metrics shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria. Association for Computational Linguistics.

Maillette de Buy Wenniger, G. and Sima'an, K. (2013a). A formal characterization of parsing word alignments by synchronous grammars with empirical evidence to the itg hypothesis. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 58–67. Association for Computational Linguistics.

Maillette de Buy Wenniger, G. and Sima'an, K. (2013b). Hierarchical alignment decomposition labels for hiero grammar rules. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 19–28.

Maillette de Buy Wenniger, G. and Sima'an, K. (2014a). Bilingual markov reordering labels for hierarchical SMT. In *Proceedings of the Eight Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 11–21.

Maillette de Buy Wenniger, G. and Sima'an, K. (2014b). Visualization, search and analysis of hierarchical translation equivalence in machine translation data. *The Prague Bulletin of Mathematical Linguistics*, (101):43–54.

Marcu, D., Wang, W., Echihabi, A., and Knight, K. (2006). Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 44–52, Sydney, Australia. Association for Computational Linguistics.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 133–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marton, Y., Chiang, D., and Resnik, P. (2012). Soft syntactic constraints for arabic—english hierarchical phrase-based translation. *Machine Translation*, 26(1-2):137–157.

Marton, Y. and Resnik, P. (2008). Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011, Columbus, Ohio. Association for Computational Linguistics.

Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005). Probabilistic cfg with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mauser, A., Hasan, S., and Ney, H. (2009). Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 210–218, Stroudsburg, PA, USA. Association for Computational Linguistics.

McCarley, J. S., Ittycheriah, A., Roukos, S., Xiang, B., and Xu, J.-m. (2011). A correction model for word alignments. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 889–898.

Melamed, D. (1998). Annotation style guide for the blinker project, version 1.0. Technical Report IRCS TR #98-06, University of Pennsylvania.

Melamed, I. D. (2003). Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 79–86.

M.Galley, J.Graehl, Knight, K., Marcu, D., DeNeefe, S., W.Wang, and Thayer, I. (2006). Scalable inference and training of context-rich syntactic models. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mi, H. and Huang, L. (2008). Forest-based translation rule extraction. In *Proceedings of EMNLP*.

Mi, H., Huang, L., and Liu, Q. (2008a). Forest-based translation. In *Proceedings of ACL: HLT*.

Mi, H., Huang, L., and Liu, Q. (2008b). Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199, Columbus, Ohio. Association for Computational Linguistics.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of Interspeech 2010*, pages 1045–1048.

Mino, H., Watanabe, T., and Sumita, E. (2014). Syntax-augmented machine translation using syntax-label clustering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 165–171, Doha, Qatar. Association for Computational Linguistics.

Mitchell, T. M. (1980). The need for biases in learning generalizations. In Shavlik, J. W. and Dietterich, T. G., editors, *Readings in Machine Learning*, pages 184–191. Morgan Kauffman. Book published in 1990.

Mylonakis, M. (2012). *Learning the Latent Structure of Translation*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam.

Mylonakis, M. and Sima'an, K. (2008). Phrase translation probabilities with itg priors and smoothing as learning objective. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 630–639, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mylonakis, M. and Sima'an, K. (2010). Learning probabilistic synchronous cfgs for phrase-based translation. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 117–125, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mylonakis, M. and Sima'an, K. (2011). Learning hierarchical translation structure with linguistic annotations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 642–652, Portland, Oregon, USA. Association for Computational Linguistics.

Nagao, M. (1984). A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, New York, NY, USA. Elsevier North-Holland, Inc.

Nederhof, M.-J. and Satta, G. (2004). The language intersection problem for non-recursive context-free grammars. *Information and Computation*, pages 172–184.

Nesson, R., Shieber, S. M., and Rush, A. (2006). Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *5th Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, Massachusetts.

Neubig, G., Watanabe, T., and Mori, S. (2012). Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 843–853, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nguyen, T. and Vogel, S. (2013a). Integrating phrase-based reordering features into a chart-based decoder for machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1587–1596, Sofia, Bulgaria. Association for Computational Linguistics.

Nguyen, T. and Vogel, S. (2013b). Integrating phrase-based reordering features into a chart-based decoder for machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1587–1596.

Nguyen, T. P., Shimazu, A., Ho, T. B., Nguyen, M. L., and Nguyen, V. V. (2008). A tree-to-string phrase-based model for statistical machine translation. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 143–150, Manchester, England. Coling 2008 Organizing Committee.

Niehues, J., Herrmann, T., Vogel, S., and Waibel, A. (2011). Wider context by using bilingual language models in machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 198–206, Edinburgh, Scotland. Association for Computational Linguistics.

Nießen, S., Vogel, S., Ney, H., and Tillmann, C. (1998). A dp based search algorithm for statistical machine translation. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 960–967, Stroudsburg, PA, USA. Association for Computational Linguistics.

Och, F. and Ney, H. (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, 3:417–449.

Och, F. J. (1999). An efficient method for determining bilingual word classes. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 71–76, Stroudsburg, PA, USA. Association for Computational Linguistics.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2000). Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 440–447.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 160–167.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1)(1):19–51.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. In *Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Och, F. J. and Weber, H. (1998). Improving statistical natural language translation with categories and rules. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 985–989, Stroudsburg, PA, USA. Association for Computational Linguistics.

Padó, S. and Lapata, M. (2006). Optimal constituent alignment with edge covers for semantic projection. In *ACL-COLING'06*, ACL-44, pages 1161–1168, Stroudsburg, PA, USA. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pauls, A. and Klein, D. (2011). Faster and smaller n-gram language models. In *Proceedings of ACL*, Portland, Oregon. Association for Computational Linguistics.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA. Association for Computational Linguistics.

Petrov, S. and Klein, D. (2007). Learning and inference for hierarchically split pcfgs. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, pages 1663–1666.

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING.

Post, M. and Gildea, D. (2009). Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48.

Poutsma, A. (2000). Data-oriented translation. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 635–641.

Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162.

Prescher, D. (2005). Inducing head-driven pcfgs with latent heads: Refining a tree-bank grammar for parsing. In *Proceedings of the 16th European Conference on Machine Learning*, ECML'05, pages 292–304, Berlin, Heidelberg. Springer-Verlag.

Prescher, D., Scha, R., Sima'an, K., and Zollmann, A. (2004). On the statistical consistency of dop estimators. In *In Proceedings of the 14th Meeting of Computational Linguistics in the Netherlands*, pages 63–77.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition.

Quirk, C. and Menezes, A. (2006). Dependency treelet translation: The convergence of statistical and example-based machine translation? *Machine Translation*, 20:43–65.

Rayner, J. and Best, D. J. (1999). Modeling ties in the sign test. *Biometrics*, pages 663–665.

Riesa, J. and Marcu, D. (2010). Hierarchical search for word alignment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 157–166.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

Saluja, A., Dyer, C., and Cohen, S. B. (2014). Latent-variable synchronous cfgs for hierarchical translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1964, Doha, Qatar. Association for Computational Linguistics.

Sangati, F. and Zuidema, W. (2011). Accurate parsing with compact tree-substitution grammars: Double-dop. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 84–95.

Sato, S. and Nagao, M. (1990). Toward memory-based translation. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*, COLING '90, pages 247–252, Stroudsburg, PA, USA. Association for Computational Linguistics.

Satta, G. and Peserico, E. (2005). Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 803–810, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Scha, R. (1990). Taaltheorie en taaltechnologie; competence en performance. In *Computertoepassingen in de Neerlandistiek.*, pages 7–22. Landelijke Vereniging van Neerlandici.

Scherrer, Y. and Sagot, B. (2013). Lexicon induction and part-of-speech tagging of non-resourced languages without any bilingual resources. In *Proceedings of the Workshop on Adaptation of Language Resources and Tools for Closely Related Languages and Language Variants*, pages 30–39, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Schwenk, H., Dchelotte, D., and Gauvain, J.-L. (2006). Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 723–730, Stroudsburg, PA, USA. Association for Computational Linguistics.

Senellart, J., Yang, J., and Rebollo, A. (2003). Systran intuitive coding technology. In *MT Summit IX: Proceedings of the ninth machine transaltion summit*, pages 346–35.

Sennrich, R. (2014). A cyk+ variant for scfg decoding without a dot chart. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 94–102, Doha, Qatar. Association for Computational Linguistics.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:623–656.

Shen, L., Sarkar, A., and Och, F. J. (2004). Discriminative reranking for machine translation. In Susan Dumais, D. M. and Roukos, S., editors, *HLT-NAACL 2004: Main Proceedings*, pages 177–184, Boston, Massachusetts, USA. Association for Computational Linguistics.

Shieber, S. M. (2007). Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 88–95, Rochester, New York. Association for Computational Linguistics.

Shieber, S. M. and Schabes, Y. (1990). Synchronous tree-adjoining grammars. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*, pages 253–258.

Sima'an, K., Bod, R., Krauwer, S., and Scha, R. (1994). Efficient disambiguation by means of stochastic tree substitution grammars. In *Proceedings International Conference on New Methods in Language Processing*, pages 50–58. UMIST.

Sima'an, K. and Maillette de Buy Wenniger, G. (2011a). Hierarchical translation equivalence over word alignments. Internal Report.

Sima'an, K. and Maillette de Buy Wenniger, G. (2011b). Hierarchical translation equivalence over word alignments. Technical Report PP-2011-38, Institute for Logic, Language and Computation.

Sima'an, K. and Maillette de Buy Wenniger, G. (2013). Hierarchical alignment trees: A recursive factorization of reordering in word alignments with empirical results. Internal Report.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the Association for Machine Translation in the Americas*, pages 223–231.

Søgaard, A. (2010). Can inversion transduction grammars generate hand alignments? In *Proccedings of the 14th Annual Conference of the European Association for Machine Translation (EAMT).*

Søgaard, A. and Kuhn, J. (2009a). Empirical lower bounds on alignment error rates in syntax-based machine translation. In *SSST '09*, pages 19–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

Søgaard, A. and Kuhn, J. (2009b). Empirical lower bounds on alignment error rates in syntax-based machine translation. In *SSST '09*, pages 19–27.

Søgaard, A. and Wu, D. (2009). Empirical lower bounds on translation unit error rate for the full class of inversion transduction grammars. In *Proceedings of the 11th International Workshop on Parsing Technologies (IWPT-2009), 7-9 October 2009, Paris, France*, pages 33–36. The Association for Computational Linguistics.

Sokolov, A., Wisniewski, G., and Yvon, F. (2012). Non-linear n-best list reranking with few features. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*.

Stanojević, M. and Sima'an, K. (2014a). BEER: BEtter evaluation as ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA. Association for Computational Linguistics.

Stanojević, M. and Sima'an, K. (2014b). Beer: Better evaluation by ranking. In *Proceedings of the Workshop on Statistical Machine Translation (WMT)*.

Stanojević, M. and Sima'an, K. (2014c). Fitting sentence level translation evaluation with many dense features. In *Proceedings of EMNLP 2014*.

Stanojević, M. and Sima'an, K. (2015). Reordering grammar induction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 44–54, Lisbon, Portugal. Association for Computational Linguistics.

Steedman, M. (1987). Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403–439.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Steinbiss, V., Tran, B.-H., and Ney, H. (1994). Improvements in beam search. In *Proceedings of the International Conference on Spoken Language Processing, (ICSLP'94)*, pages 2143–2146.

Sumita, E. and Iida, H. (1991). Experiments and prospects of example-based machine translation. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, ACL '91, pages 185–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In Calzolari, N., Choukri, K., Declerck, T., Uğur Doğan, M., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Tillmann, C. (2004). A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tillmann, C. and Ney, H. (2000). Word re-ordering and dp-based search in statistical machine translation. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 850–856, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tillmann, C. and Ney, H. (2003). Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Comput. Linguist.*, 29(1):97–133.

Tillmann, C., Vogel, S., Ney, H., and Zubiaga, A. (1997a). A dp based search using monotone alignments in statistical translation. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, pages 289–296, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tillmann, C., Vogel, S., Ney, H., Zubiaga, A., and Sawaf, H. (1997b). Accelerated dp based search for statistical translation. In *Proceedings of the Fifth European Conference on Speech Communication and Technology (Eurospeech 97)*, pages 2667–2670.

Tillmann, C. and Zhang, T. (2006). A discriminative global training algorithm for statistical mt. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 721–728, Stroudsburg, PA, USA. Association for Computational Linguistics.

Titov, I. and Klementiev, A. (2012). Crosslingual induction of semantic roles. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 647–656.

Toma, P. (1977). Systran as a multilingual machine translation system. In *Overcoming the language barrier: Third European congress on information systems and networks*, pages 569–581. Verlag Dokumentation, München.

Tran, K. M., Bisazza, A., and Monz, C. (2014). Word translation prediction for morphologically rich languages with bilingual neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1676–1688, Doha, Qatar. Association for Computational Linguistics.

Tromble, R. and Eisner, J. (2009). Learning linear ordering problems for better translation. In *Proceedings of EMNLP'09*, pages 1007–1016.

Tromble, R. W., Kumar, S., Och, F., and Macherey, W. (2008). Lattice minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 620–629, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 104–, New York, NY, USA. ACM.

Uno, T. and Yagiura, M. (2000). Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, pages 290–309.

Uszkoreit, J. and Brants, T. (2008). Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-08: HLT*, pages 755–762, Columbus, Ohio. Association for Computational Linguistics.

van Cranenburgh, A. and Bod, R. (2013). Discontinuous parsing with an efficient and accurate dop model. In *Proceedings of the International Conference on Parsing Technologies*, pages 27–29.

van Noord, G. (1995). The intersection of finite state automata and definite clause grammars. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 159–165, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vaswani, A., Mi, H., Huang, L., and Chiang, D. (2011). Rule markov models for fast tree-to-string translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*

- *Volume 1*, HLT '11, pages 856–864, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vauquois, B. (1968). A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *IFIP Congress-68*, pages 254–260.

Vauquois, B. (1975). *La traduction automatique à Grenoble*. Documents de linguistique quantitative. Dunod.

Vauquois, B. and Boitet, C. (1985). Automated translation at grenoble university. *Comput. Linguist.*, 11(1):28–36.

Venugopal, A., Zollmann, A., Smith, N. A., and Vogel, S. (2009). Preference grammars: softening syntactic constraints to improve statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 236–244, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vijay-Shanker, K., J. Weir, D., and K. Joshi, A. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Viterbi, A. J. (1969). Error bounds for the white gaussian and other very noisy memoryless channels with generalized decision regions. *IEEE Transactions on Information Theory*, 15(2):279–287.

Vogel, S., Ney, H., and Tillmann, C. (1996). Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, pages 836–841.

Wang, W., Knight, K., and Marcy, D. (2007). Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754.

Wang, W., May, J., Knight, K., and Marcu, D. (2010). Re-structuring, re-labeling and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36:247–277.

Wang, X., Utiyama, M., Finch, A., and Sumita, E. (2014). Empirical study of unsupervised chinese word segmentation methods for smt on large-scale corpora. In *"Proceedings of ACL-2014 Volume 2: Short Papers"*, pages 752–758.

Wang, Y.-Y. and Waibel, A. (1997). Decoding algorithm in statistical machine translation. In *Proceedings of the 35thAnnual Conference of the Association forComputational Linguistics (ACL)*, pages 366–372.

Wang, Y.-Y. and Waibel, A. (1998a). Fast decoding for statistical machine translation. In *In Proceeings of the Fifth International Conference Spoken Language Processing (ICSLP 98)*, pages 2775–2778.

Wang, Y.-Y. and Waibel, A. (1998b). Modeling with structures in statistical machine translation. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 1357–1363, Stroudsburg, PA, USA. Association for Computational Linguistics.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic. Association for Computational Linguistics.

Wellington, B., Waxmonsky, S., and Melamed, I. D. (2006). Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Williams, P. and Koehn, P. (2011). Agreement constraints for statistical machine translation into german. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 217–226, Edinburgh, Scotland. Association for Computational Linguistics.

Williams, P., Sennrich, R., Nadejde, M., Huck, M., Hasler, E., and Koehn, P. (2014). Edinburgh's syntax-based systems at wmt 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 207–214, Baltimore, Maryland, USA. Association for Computational Linguistics.

Wu, D. (1995). Stochastic inversion transduction grammars with application to segmentation, bracketing, and alignment of parallel corpora. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1328–1335, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.

Wu, D. (2005). MT model space: statistical versus compositional versus example-based machine translation. *Machine Translation*, 19(3-4):213–227.

Wu, D., Carpuat, M., and Shen, Y. (2006). Inversion transduction grammar coverage of arabic-english word alignment for tree-structured statistical machine translation.

In *Proceedings of the IEEE/ACL 2006 Workshop on Spoken Language Technology (SLT 2006)*.

Wu, D. and Wong, H. (1998). Machine translation with a stochastic grammatical channel. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 1408–1415, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wu, H. and Wang, H. (2007). Comparative study of word alignment heuristics and phrase-based smt. In *Proc. of Machine Translation Summit XI*, pages 507–514.

Wu, M. C. D. (2007). Improving statistical machine translation using word sense disambiguation. In *Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, page 61–72.

Xiao, T. and Zhu, J. (2013). Unsupervised sub-tree alignment for tree-to-tree translation. *Journal of Artificial Intelligence Research*, 48(1):733–782.

Xiao, X., Su, J., Liu, Y., Liu, Q., and Lin, S. (2011). An orientation model for hierarchical phrase-based translation. In *Proceedings of the 2011 International Conference on Asian Language Processing*, IALP '11, pages 165–168, Washington, DC, USA. IEEE Computer Society.

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Zens, R., Hasan, S., and Ney, H. (2007). A systematic comparison of training criteria for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 524–532, Prague, Czech Republic. Association for Computational Linguistics.

Zens, R. and Ney, H. (2003). A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the Annual Meeting of the ACL*, pages 144–151.

Zens, R., Och, F. J., , and Ney, H. (2002). Phrase-based statistical machine translation. In *KI 2002: Advances in Artificial Intelligence. 25. Annual German Conference on AI (KI 2002)*, volume 2479, pages 18–32. Springer Verlag.

Zhang, D., Li, M., Li, C.-H., and Zhou, M. (2007a). Phrase reordering model integrating syntactic knowledge for SMT. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 533–540.

Zhang, H. and Gildea, D. (2005). Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 475–482, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, H. and Gildea, D. (2007). Factorization of synchronous context-free grammars in linear time. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, SSST '07, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, H., Gildea, D., and Chiang, D. (2008a). Extracting synchronous grammar rules from word-level alignments in linear time. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 1081–1088, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, M., Jiang, H., Aw, A., Li, H., Tan, C. L., and Li, S. (2008b). A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567, Columbus, Ohio. Association for Computational Linguistics.

Zhang, M., Jiang, H., Aw, A., Sun, J., Li, S., and Tan, C. L. (2007b). A tree-to-tree alignment-based model for statistical machine translation. In *Proceedings of the MT Summit XI*.

Zhang, Y., Lei, T., Barzilay, R., and Jaakkola, T. (2014). Greed is good if randomized: New inference for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1013–1024, Doha, Qatar. Association for Computational Linguistics.

Zhou, B., Xiang, B., Zhu, X., and Gao, Y. (2008). Prior derivation models for formally syntax-based translation using linguistically syntactic parsing and tree kernels. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, pages 19–27, Columbus, Ohio. Association for Computational Linguistics.

Zollmann, A. (2011). *Learning Multiple-Nonterminal Synchronous Grammars for Statistical Machine Translation*. PhD thesis, Carnegie Mellon University.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 138–141, Stroudsburg, PA, USA. Association for Computational Linguistics.

Index

Samenvatting

Globalisering is een van de kenmerken van onze tijd. Documenten en geschreven teksten zijn wereldwijd beschikbaar maar ze zijn niet altijd toegankelijk vanwege taalbarrières. Het werk van menselijke vertalers is tijdrovend en kostbaar. De introductie van machinevertaling heeft nieuwe mogelijkheden gegeven, maar de resultaten van machinevertaling schieten vaak nog tekort op verscheidene vlakken met inbegrip van de woordvolgorde.

Deze dissertatie stelt manieren voor om hiërarchische vertaal equivalentie relaties afgeleid van *woord alignments* te gebruiken om hiërarchische statistische machinevertaling te verbeteren. Deze verbetering betreft in het bijzonder de woordvolgorde en coherentie van de geproduceerde vertalingen, maar tevens de keuze van woorden.

Het belangrijkste probleem dat in deze dissertatie wordt behandeld is dat hiërarchische statistische machinevertaling weinig context gebruikt in het combineren van regels tot vertalingen. Dit leidt tot separaat gemaakte en daarmee slecht gecoördineerde herordenings beslissingen. Met name HIERO (Chiang, 2005) grammatica's zijn niet van voorzien van nonterminal labels, wat er toe leidt dat de decoder de context van andere regels negeert wanneer regels worden toegepast.

Het fundament van deze dissertatie is een expliciete representatie van de hiërarchische vertaal equivalentie structuur afgeleid uit woord alignments, gebruikmakend van het nieuw voorgestelde kader hierarchical alignment trees (HATs) (Sima'an and Maillette de Buy Wenniger, 2013). Dit maakt het mogelijk om te modelleren hoe vertaal equivalentie is opgebouwd in geobserveerde data, en hieruit te generaliseren om contextgevoelige regels te leren die kunnen worden gecombineerd voor de vertaling van ongeziene data. Het belangrijkste specifieke geval van dit algemene schema dat wordt behandeld in deze dissertatie, is het gebruik van HATs voor het leveren van een om een herordenings context aan regels in de vorm van labels.

Het tekortschietende gebruik van context door HIERO is eerder aangepakt in ander werk door het toevoegen van syntactische labels aan HIERO. Het populaire systeem syntax-augmented machine translation (SAMT) (Zollmann and Venugopal, 2006) is het standaard voorbeeld van deze aanpak. Maar er zijn twee problemen met het

gebruik van syntax. Het eerste probleem is dat syntax en alignment structuur vaak niet verenigbaar zijn. Het tweede probleem is dat betrouwbare parsers vaak niet beschikbaar zijn voor alle talen. Deze problemen motiveren onze nieuwe aanpak, die niet afhankelijk is van syntax, maar alleen van de rijke informatie die in word alignments aanwezig is. Op basis van deze word alignments worden bilinguale herordenings labels gevormd die het mogelijk maken om in de combinatie van regels betere, contextgevoelige herordenings beslissingen te nemen. Herordenings labels worden toegepast in combinatie met een flexibele vorm van label matching, die het mogelijk maakt om zachte preferenties voor specifieke label substituties te leren tijdens de tuning. Deze labels leveren significante verbetering op ten opzichte van zowel HIERO en SAMT voor drie verschillende talen paren, met de sterkste verbetering voor de vertaling van Chinees naar Engels.

Waar komen herordenings labels vandaan? Herordenings labels komen van HATs. HATs zijn bilinguale bomen die de hiërarchische vertaal equivalentie structuur afgeleid uit woord alignments representeren. HATs representeren alle doorlopende translation equivalence units (TEUs) die kunnen worden afgeleid uit woord alignments.

Hoe verschillen HATs van bestaande representaties voor TEUs en hiërarchische herordenings structuur? HATs bouwen verder op permutation trees (PETs) (Gildea et al., 2006) en normalized decomposition trees (NDTs) (Zhang et al., 2008a). Belangrijk is dat HATs alle informatie representeren die aanwezig is in originele woord alignments, wat ze onderscheidt van NDTs die alleen de decompositie structuur van de TEUs representeren. Cruciaal is dat HATs zowel PETs en NDTs generaliseren door willekeurige woord alignments te representeren en tegelijkertijd een representatie te geven van de recursieve bilinguale overeenkomstigheidsrelaties voor alle op uit woord alignments afgeleide TEUs.

Welke nieuwe bijdragen worden geleverd in deze dissertatie op basis van HATs? HATs leveren net als NDTs een basis voor het extraheren van bilinguale regels, maar anders dan NDTs leveren zij ook de informatie die nodig is om herordenings labels te vormen voor deze regels. Verder zijn HATs toegepast om hiërarchische vertaal equivalentie te visualiseren en daarmee een beter kwalitatief begrip van empirische hiërarchische vertaal equivalentie te faciliteren (Maillette de Buy Wenniger and Sima'an, 2014b). Daarnaast worden HATs gebruikt om de complexiteit van empirische vertaal equivalentie te bepalen, zoals we hierna in meer detail zullen bespreken.

Het laatste deel van deze dissertatie onderzoekt hoe de complexiteit van empirische vertaal equivalentie afgeleid uit woord alignments kan worden gekarakteriseerd. Specifiek, gegeven een woord alignment en een grammatica, probeert de dissertatie de formele vraag te beantwoorden wat het betekent voor de grammatica om het woord alignment te ondersteunen. Een exacte manier om deze vraag te beantwoorden wordt voorgesteld, gebaseerd op de intersectie van 1) de set van vertaalequivalenten afgeleid uit het woord alignment en 2) de set vertaalequivalenten afleidbaar uit de grammatica. Vervolgens wordt aangetoond dat HATs kunnen worden toegepast om deze procedure efficiënt te implementeren en te gebruiken om de dekking van woord alignments door een grammatica exact te kunnen meten, zonder een expliciete intersectie van sets van

vertaalequivalenten te hoeven uitvoeren. Dit maakt het mogelijk om de meting niet alleen exact te doen maar ook efficient te maken.

Een grote empirische studie van zowel handmatig- als automatisch gegenereerde word alignments laat zien dat: 1) Empirische hiërarchische vertaal equivalentie veel complexer is dan doorgaans wordt aangenomen, 2) voor alle talenparen een grote fractie van de woord alignments niet is te binariseren noch is te beschrijven door uitsluitend permutaties (bijectieve projecties), 3) complexe alignment configuraties tot een bepaalde lengte inbedden in atomaire units en deze negeren bij de bepaling van complexiteit, lost slechts een deel van de complexe alignment configuraties op en is op zichzelf niet voldoende om volledige dekking van woord alignments te bereiken.

Deze dissertatie laat zien dat het mogelijk is om significante verbeteringen van woordvolgorde en coherentie in statistische machinevertaling te bewerkstelligen alleen op basis van de informatie in woord alignments en zonder gebruik van syntax. De dissertatie levert het nieuwe kader van HATs als bijdrage aan machinevertaling. Daarnaast wordt het nut daarvan beschreven bij verscheidene toepassingen inclusief regel extractie, labeling alsook de empirische studie van hiërarchische vertaal equivalentie.

Abstract

Globalization is one of the characteristics of our time. Documents and written texts are available from all over the world but they are not always accessible because of language barriers. The work of human translators is time consuming and costly. The introduction of machine translation has given new opportunities, but the results of machine translation are often still lacking in various aspects including word order.

This dissertation contributes methods to improve hierarchical statistical machine translation (SMT) using the hierarchical translation equivalence relations induced from word alignments. The information obtained from these relations improves word order and coherence of the produced translations in particular, but lexical choice is affected as well.

The core problem addressed in this dissertation is that hierarchical SMT uses little context when composing rules into translations, leading to independently made and poorly coordinated reordering decisions. In particular, Hiero (Chiang, 2005) grammars lack labels for nonterminals, causing the decoder to ignore the context of other rules when rules are applied.

The foundational idea in this thesis is to explicitly represent the hierarchical translation equivalence structure induced by word alignments, using a newly proposed framework called hierarchical alignment trees (HATs) (Sima'an and Maillette de Buy Wenniger, 2013). This allows us to model how translation equivalence is composed in observed data, and generalize from this to learn context-aware rules that can be composed for the translation of unseen data. The main specific case of this general scheme addressed in this thesis, is the use of this representation to provide reordering context to hierarchical rules in the form of labels.

The poor use of context by Hiero has been addressed before in the literature by adding syntactic labels to Hiero. The popular system syntax-augmented machine translation (SAMT) (Zollmann and Venugopal, 2006) is the standard example of this approach. But there are two main problems with using syntax. The first problem is that syntax and alignment structure are not necessarily compatible. The second problem is that reliable parsers are not available for all languages. These problems motivate

our new approach, which does not rely on syntax, but instead on the rich information from the word alignments. These word alignments serve to construct bilingual reordering labels that allow rules to support better, context-aware reordering decisions. Reordering labels are applied in combination with a loosely constrained approach to label matching, which allow the grammar to learn soft preferences for particular label substitutions during tuning. These labels yield significant improvements over both HIERO and SAMT for three different language pairs, with the strongest improvements obtained for Chinese–English translation.

Where do reordering labels come from? Reordering labels come from HATs. HATs are bilingual trees that represent the hierarchical translation equivalence structure induced from word alignments. HATs compactly represent all contiguous translation equivalence units (TEUs) that can be induced from word alignments.

How do HATs differ from existing representations for TEUs and hierarchical reordering structure? HATs build further upon permutation trees (PETs) (Gildea et al., 2006) and normalized decomposition trees (NDTs) (Zhang et al., 2008a). Importantly, HATs preserve all information present in the original word alignments, distinguishing them from NDTs which present only the decomposition structure of the TEUs. Crucially HATs generalize both PETs and NDTs by representing arbitrary discontiguous word alignments while at the same time representing the recursive bilingual correspondence relations for all TEUs induced from word alignments.

What new contributions are made based on HATs in this thesis? HATs like NDTs provide a basis for the extraction of bilingual rules, but unlike NDTs they also provide the information required to form reordering labels for those rules. Furthermore HATs have been applied to visualize hierarchical translation equivalence, accommodating a better qualitative understanding of empirical hierarchical translation equivalence (Maillette de Buy Wenniger and Sima'an, 2014b). Additionally, HATs have been used to study the complexity of empirical translation equivalence quantitatively, as discussed in more detail next.

The last part of this thesis examines how to characterize the complexity of empirical translation equivalence as induced from word alignments. In particular, given a word alignment and a grammar, we try to give a formal answer to the question what it means for the grammar to cover the word alignment. Based on the intersection of the sets of TEUs induced from the word alignment and inferable from the grammar, we contribute a method to answer this question exactly. This contrasts with other work providing only upper bounds on alignment coverage. It is then shown how HATs can be applied to implement our method, while avoiding explicit intersection of sets of translation equivalents. This enables exact measurement of alignment coverage that is also efficient.

A large empirical study of both manually and automatically produced word alignments shows that: 1) Empirical hierarchical translation equivalence is much more complex than commonly believed, 2) for all language pairs, a large fraction of the word alignments is neither binarizable, nor coverable by only permutations (bijective mappings), 3) embedding complex alignment configurations up to a limited

maximum length in atomic units that are ignored for complexity only eliminates part of the complex alignment configurations and is by itself not sufficient to achieve full alignment coverage.

This thesis shows that word order and coherence of hierarchical statistical machine translation can be significantly improved without syntax, by using just the information present in word alignments. The thesis contributes the framework of HATs and demonstrates its usefulness for a variety of applications including rule extraction, labeling as well as analysis of empirical hierarchical translation equivalence.

Titles in the ILLC Dissertation Series:

ILLC DS-2009-01: **Jakub Szymanik**
Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language

ILLC DS-2009-02: **Hartmut Fitz**
Neural Syntax

ILLC DS-2009-03: **Brian Thomas Semmes**
A Game for the Borel Functions

ILLC DS-2009-04: **Sara L. Uckelman**
Modalities in Medieval Logic

ILLC DS-2009-05: **Andreas Witzel**
Knowledge and Games: Theory and Implementation

ILLC DS-2009-06: **Chantal Bax**
Subjectivity after Wittgenstein. Wittgenstein's embodied and embedded subject and the debate about the death of man.

ILLC DS-2009-07: **Kata Balogh**
Theme with Variations. A Context-based Analysis of Focus

ILLC DS-2009-08: **Tomohiro Hoshi**
Epistemic Dynamics and Protocol Information

ILLC DS-2009-09: **Olivia Ladinig**
Temporal expectations and their violations

ILLC DS-2009-10: **Tikitu de Jager**
"Now that you mention it, I wonder…": Awareness, Attention, Assumption

ILLC DS-2009-11: **Michael Franke**
Signal to Act: Game Theory in Pragmatics

ILLC DS-2009-12: **Joel Uckelman**
More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains

ILLC DS-2009-13: **Stefan Bold**
Cardinals as Ultrapowers. A Canonical Measure Analysis under the Axiom of Determinacy.

ILLC DS-2010-01: **Reut Tsarfaty**
Relational-Realizational Parsing

ILLC DS-2010-02: **Jonathan Zvesper**
Playing with Information

ILLC DS-2010-03: **Cédric Dégremont**
The Temporal Mind. Observations on the logic of belief change in interactive systems

ILLC DS-2010-04: **Daisuke Ikegami**
Games in Set Theory and Logic

ILLC DS-2010-05: **Jarmo Kontinen**
Coherence and Complexity in Fragments of Dependence Logic

ILLC DS-2010-06: **Yanjing Wang**
Epistemic Modelling and Protocol Dynamics

ILLC DS-2010-07: **Marc Staudacher**
Use theories of meaning between conventions and social norms

ILLC DS-2010-08: **Amélie Gheerbrant**
Fixed-Point Logics on Trees

ILLC DS-2010-09: **Gaëlle Fontaine**
Modal Fixpoint Logic: Some Model Theoretic Questions

ILLC DS-2010-10: **Jacob Vosmaer**
Logic, Algebra and Topology. Investigations into canonical extensions, duality theory and point-free topology.

ILLC DS-2010-11: **Nina Gierasimczuk**
Knowing One's Limits. Logical Analysis of Inductive Inference

ILLC DS-2010-12: **Martin Mose Bentzen**
Stit, Iit, and Deontic Logic for Action Types

ILLC DS-2011-01: **Wouter M. Koolen**
Combining Strategies Efficiently: High-Quality Decisions from Conflicting Advice

ILLC DS-2011-02: **Fernando Raymundo Velazquez-Quesada**
Small steps in dynamics of information

ILLC DS-2011-03: **Marijn Koolen**
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

ILLC DS-2011-04: **Junte Zhang**
System Evaluation of Archival Description and Access

ILLC DS-2011-05: **Lauri Keskinen**
Characterizing All Models in Infinite Cardinalities

ILLC DS-2011-06: **Rianne Kaptein**
Effective Focused Retrieval by Exploiting Query Context and Document Structure

ILLC DS-2011-07: **Jop Briët**
Grothendieck Inequalities, Nonlocal Games and Optimization

ILLC DS-2011-08: **Stefan Minica**
Dynamic Logic of Questions

ILLC DS-2011-09: **Raul Andres Leal**
Modalities Through the Looking Glass: A study on coalgebraic modal logic and their applications

ILLC DS-2011-10: **Lena Kurzen**
Complexity in Interaction

ILLC DS-2011-11: **Gideon Borensztajn**
The neural basis of structure in language

ILLC DS-2012-01: **Federico Sangati**
Decomposing and Regenerating Syntactic Trees

ILLC DS-2012-02: **Markos Mylonakis**
Learning the Latent Structure of Translation

ILLC DS-2012-03: **Edgar José Andrade Lotero**
Models of Language: Towards a practice-based account of information in natural language

ILLC DS-2012-04: **Yurii Khomskii**
Regularity Properties and Definability in the Real Number Continuum: idealized forcing, polarized partitions, Hausdorff gaps and mad families in the projective hierarchy.

ILLC DS-2012-05: **David García Soriano**
Query-Efficient Computation in Property Testing and Learning Theory

ILLC DS-2012-06: **Dimitris Gakis**
Contextual Metaphilosophy - The Case of Wittgenstein

ILLC DS-2012-07: **Pietro Galliani**
The Dynamics of Imperfect Information

ILLC DS-2012-08: **Umberto Grandi**
Binary Aggregation with Integrity Constraints

ILLC DS-2012-09: **Wesley Halcrow Holliday**
Knowing What Follows: Epistemic Closure and Epistemic Logic

ILLC DS-2012-10: **Jeremy Meyers**
Locations, Bodies, and Sets: A model theoretic investigation into nominalistic mereologies

ILLC DS-2012-11: **Floor Sietsma**
Logics of Communication and Knowledge

ILLC DS-2012-12: **Joris Dormans**
Engineering emergence: applied theory for game design

ILLC DS-2013-01: **Simon Pauw**
Size Matters: Grounding Quantifiers in Spatial Perception

ILLC DS-2013-02: **Virginie Fiutek**
Playing with Knowledge and Belief

ILLC DS-2013-03: **Giannicola Scarpa**
Quantum entanglement in non-local games, graph parameters and zero-error information theory

ILLC DS-2014-01: **Machiel Keestra**
Sculpting the Space of Actions. Explaining Human Action by Integrating Intentions and Mechanisms

ILLC DS-2014-02: **Thomas Icard**
The Algorithmic Mind: A Study of Inference in Action

ILLC DS-2014-03: **Harald A. Bastiaanse**
Very, Many, Small, Penguins

ILLC DS-2014-04: **Ben Rodenhäuser**
A Matter of Trust: Dynamic Attitudes in Epistemic Logic

ILLC DS-2015-01: **María Inés Crespo**
Affecting Meaning. Subjectivity and evaluativity in gradable adjectives.

ILLC DS-2015-02: **Mathias Winther Madsen**
The Kid, the Clerk, and the Gambler - Critical Studies in Statistics and Cognitive Science

ILLC DS-2015-03: **Shengyang Zhong**
Orthogonality and Quantum Geometry: Towards a Relational Reconstruction of Quantum Theory

ILLC DS-2015-04: **Sumit Sourabh**
Correspondence and Canonicity in Non-Classical Logic

ILLC DS-2015-05: **Facundo Carreiro**
Fragments of Fixpoint Logics: Automata and Expressiveness

ILLC DS-2016-01: **Ivano A. Ciardelli**
Questions in Logic

ILLC DS-2016-02: **Zoé Christoff**
Dynamic Logics of Networks: Information Flow and the Spread of Opinion