Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition 38 (2005) 1407 – 1419

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Natural language grammar induction with a generative constituent-context model[☆]

Dan Klein, Christopher D. Manning*

Computer Science Department, Stanford University, 353 Serra Mall, Room 418, Stanford, CA 94305-9040, USA

Received 17 March 2004; accepted 17 March 2004

Abstract

We present a generative probabilistic model for the unsupervised learning of hierarchical natural language syntactic structure. Unlike most previous work, we do not learn a context-free grammar, but rather induce a distributional model of constituents which explicitly relates constituent yields and their linear contexts. Parameter search with EM produces higher quality analyses for human language data than those previously exhibited by unsupervised systems, giving the best published unsupervised parsing results on the ATIS corpus. Experiments on Penn treebank sentences of comparable length show an even higher constituent F_1 of 71% on non-trivial brackets. We compare distributionally induced and actual part-of-speech tags as input data, and examine extensions to the basic model. We discuss errors made by the system, compare the system to previous models, and discuss upper bounds, lower bounds, and stability for this task.
© 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Natural language; Structure learning; Grammar induction; Distributional clustering; Unsupervised learning

1. Introduction

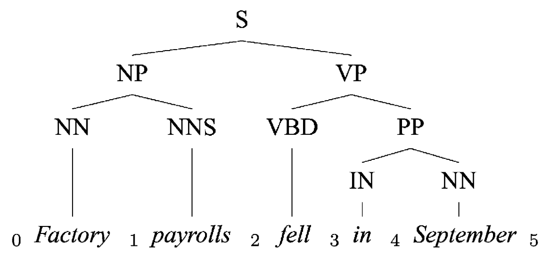
The task of inducing hierarchical natural language structure from observed yields alone has received a great deal of attention [1–8]. Researchers have explored this problem for a variety of reasons: to argue empirically against the poverty of the stimulus [9], to model child language acquisition [10], to use induction systems as a first stage in constructing large treebanks [11], or to build better language models [12,13].

The goal of a hierarchical structure induction system is to learn syntactic structures like the tree shown in Fig. 1(a). The underlying hypothesis is that there exists a natural and useful decomposition of natural language sentences into nested units, or *constituents*, whose reality rests on some kind of structural, semantic, or distributional coherence. For example, in the analysis of the sentence shown, *factory payrolls* is posited as a constituent, while *fell in* is not. Various arguments can be put forward to justify constituency (a full discussion is beyond the scope of this article, but see Radford [14] *inter alia* for a comprehensive presentation). For example, other units like *three stocks* can be substituted for *factory payrolls* without destroying the well-formedness of the sentence, and *factory payrolls* can appear as a unit in other contexts (e.g. *They cut factory payrolls by* 10%). Such distributional criteria were suggested as the basis of a grammar learning process in Harris [15] and, formalized in various ways, have formed the basis of many grammar learning systems since.

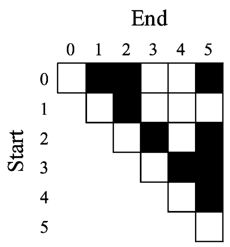
[☆] This paper is an expanded version of a paper delivered at the 40th meeting of the Association for Computational Linguistics. This paper is based on work supported in part by the National Science Foundation under Grant No. IIS-0085896 and by an IBM Faculty Partnership Award to the second author.

* Corresponding author. Tel.: +1 650 723 7683; fax: +1 650 725 2588.
E-mail address: manning@cs.stanford.edu (C.D. Manning).

0031-3203/$30.00 © 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2004.03.023



(a)



(b)

Span	Label	Yield	Context
⟨0,5⟩	S	NN NNS VBD IN NN	⋄ – ⋄
⟨0,2⟩	NP	NN NNS	⋄ – VBD
⟨2,5⟩	VP	VBD IN NN	NNS – ⋄
⟨3,5⟩	PP	IN NN	VBD – ⋄
⟨0,1⟩	NN	NN	⋄ – NNS
⟨1,2⟩	NNS	NNS	NN – VBD
⟨2,3⟩	VBD	VBD	NNS – IN
⟨3,4⟩	IN	IN	VBD – NN
⟨4,5⟩	NN	NNS	IN – ⋄

(c)

Fig. 1. (a) Example parse tree with (b) its associated bracketing and (c) the yields and contexts for each constituent span in that bracketing. Distituent yields and contexts are not shown, but *are* modeled.

While tree representations of natural language are not perfect, much of the broad syntactic structure of natural language can be captured by them. By far the most common formal model of such structures is the family of context-free grammars (CFGs) or their probabilistic counterparts, the probabilistic context-free grammars (PCFGs). With the exception of van Zaanen [11], all of the systems cited above attempt to induce a CFG or PCFG from observed well-formed sentences. A famous result of Gold [16] is that all superfinite languages, including in particular CFGs, are not identifiable in the limit, meaning that, even if given an infinite amount of data generated by a CFG, it is impossible to uniquely determine which CFG produced that data. The slightly less famous counter-result [17] is that PCFGs *can* be learned in the limit from positive data alone (although it did not present a practical procedure for doing so).

While some recent work [18] has further explored the theoretical issues in PCFG-learning as they might pertain to natural languages, even learning regular languages is a very difficult problem, both in practice [19] and in theory [20,21]. While people have nonetheless used tree-structured models in grammar induction, their complexity and amount of hidden structure has tended to cause results to be poor. As a consequence, most researchers in natural language processing have turned to supervised methods for sentence analysis, *inter alia* [22,23]. This paper introduces a way to get the benefits of tree-structured models while avoiding the direct induction of (P)CFGs. Rather, we learn (nested) bracketings of data sentences directly. As such, the success of this system is best measured by its empirical behavior on natural language examples.

2. Previous work

Some work on learning syntax has assumed that semantic representations are already specified [10,24], and this assumption is probably correct for child language acquisition, but many grammar induction systems, including the present work, look at the problem of learning formal constituency, in the absence of such prior constraint.

Early work on grammar induction emphasized heuristic structure search, where the primary induction is done by incrementally adding new productions to an initially empty grammar [25,4,3]. For example, given an initially empty grammar, and assuming we already knew part-of-speech distinctions, we might notice an abundance of adjective-noun pairs and decide to add a production of the form X → ADJECTIVE NOUN to our grammar. Later, another production using X might be added. As linguists observing the process, we might hope that this X category would represent a $\overline{\mathrm{N}}$ (i.e., a modified noun unit).

In the early 1990s, attempts were made to do (probabilistic) grammar induction by parameter search, where the broad structure of the grammar is fixed in advance and only parameters are induced [26,5]. For example, the same rule X→ ADJECTIVE NOUN would have been around from the beginning, but so would other rules like X → VERB DETERMINER. On this approach, the question of which rules are included or excluded becomes the question of which parameters are zero. Classic experiments [26] showed that the structure induction task is quite difficult for even simple PCFGs. By placing a complexity-based prior on grammars, Stolcke and Omohundro [8] extended such techniques to maximum a posteriori estimation of grammars. Their grammar search procedure started with maximally specific grammar rules read off of trees, and then did greedy state merging. At the level of rewrites of an individual nonterminal, such a procedure is similar to the state merging procedures most commonly used for finite state automata induction [27,19]. However, for large natural language grammars, parameter search methods seemed fragile, and performed poorly.

Partially because of disappointing results from parameter search methods, and partially because of the conceptual appeal of modeling the actual incremental search process, most recent work has returned to structure search. Recently, several researchers have taken new approaches: van Zaanen [11] uses alignment-based learning to directly reason about constituency from minimal pairs of sentences; Adriaans and Haas [18] discuss a theoretically sound method for identifying a certain restricted class of PCFGs, but it does not seem to perform well in practice on natural language data; Clark [9] uses a distributional clustering technique along with a constituent-filtering heuristic to achieve substantially more success. Structure search methods are local; one must have a way of making local decisions about which merges will produce a coherent, globally good grammar. To the extent that such incremental approaches work, they work because good local heuristics have been explicitly engineered [28,9].

Parameter search is also local, but in another way—parameters which are locally optimal may be globally poor. A concrete example is the experiments from [5]. They restricted the space of grammars to the PCFGs which are isomorphic to a dependency grammar over the part-of-speech (POS) symbols in the Penn treebank (a collection of 1 million words of hand-parsed text [29], which gives traditional parses such as the one shown in Fig. 1(a)). Such a grammar has productions like VERB → NOUN VERB indicating that a verb-headed unit (like a verb phrase) can take as a left argument a noun-headed unit (like a noun phrase) and become a new verb-headed unit (like a sentence).[1] There is one category for each part-of-speech type (henceforth referred to as *tags*), of which there are 45 in the Treebank tag set. Carroll and Charniak then searched for parameters with the inside-outside algorithm (a special case of the EM algorithm for PCFGs, [12]) starting with 300 random production-weight vectors. Each seed converged to a different locally optimal grammar, none of them nearly as good as the treebank grammar, measured either by parsing performance or data-likelihood.

Parameter search methods do have a potential advantage over structure search methods. By aggregating over only valid, complete parses of each sentence, they naturally incorporate the constraint that constituents cannot cross—the bracketing decisions made by the grammar must be coherent. However, the Carroll and Charniak experiments had two primary causes for failure which nullified this benefit. First, random initialization is not always good, nor even necessary. The parameter space for a PCFG is riddled with local likelihood maxima, and starting with a very specific, but random, grammar should not be expected to work well. To give a concrete example, when a verb has a determiner-noun direct object (*call the number*), the natural unit is the determiner-noun pair (*the number*), not the verb-determiner pair (*call the*). However, if a grammar has a strong initial preference for the latter analysis, no amount of data will cause it to switch; that would mean both taking away mass from one rule (like VERB → VERB DETERMINER) and adding mass to another rule (like NOUN → DETERMINER NOUN). This kind of ridge operation is generally difficult for search procedures, and EM empirically will not explore across such ridges.

The reason grammars are generally given random initialization is because in many situations it is necessary to break model symmetry. For example, in the K-means algorithm (which is an instance of hard-assignment EM for a mixture of spherical gaussians), one cannot begin with all the means at the same point. This configuration is a saddle point, like a balanced needle, because the problem is symmetric in the means—each data point pulls equally on each mean in that configuration. This symmetry is not present in the case of dependency grammars: it is not the case that any grammar symbol can generate any sequence. For a sequence like *adjective noun*, only the non-terminals corresponding to *noun* and *adjective* can be parsed over it, regardless of the initial parameters. We duplicated the Carroll and Charniak experiments, but used a uniform parameter initialization where all productions were equally likely. This allowed the interaction between the grammar and data to break the initial symmetry, and resulted in an induced grammar of higher quality than Carroll and Charniak reported. This grammar, which we refer to as DEP-PCFG will be evaluated in more detail in Section 4.

Viewed with hindsight, the second way in which their experiment was guaranteed to be somewhat unencouraging is that a delexicalized dependency grammar is a very poor model of human language, even in a supervised setting. By the F_1 measure used in the experiments in Section 4, an induced dependency PCFG scores 48.2, compared to a score of 82.1 for a supervised PCFG read from local trees of the treebank. However, a supervised *dependency* PCFG scores only 53.5, not much better than the unsupervised version, and worse than the right-branching baseline (of 60.0) discussed in Section 4. As an example of the inherent shortcomings of such a dependency grammar, it is structurally unable to distinguish whether the subject or object should be attached to the verb first. Since both parses involve the same set of productions (VERB → NOUN VERB for the subject and VERB → VERB NOUN for the object), both analyses will have equal likelihood. This is a consequence of the grammar symbols not being sufficiently rich to distinguish between a verb, verb phrase, and sentence: they are all headed by a verb, but have very different syntactic behavior.

While grammar induction has only recently begun to scale successfully to real language data, distributional clustering has proven much more robust, at least for discovering patterns at the word level [30–32]. In distributional clustering, items are clustered based on the distributions of their linear contexts. For example, both *president* and *report* occur frequently between *the* on the left and *said* on the right and might therefore be clustered together. Since there are fairly good methods of discovering word groups which to a certain degree capture part-of-speech distinctions, we, like other authors, take our input to be word cluster sequences. For most of our experiments below, we used the true part-of-speech tags from the Penn treebank sentences, but we also experimented with automatically induced word classes to verify that the success of the system was not due purely to human-chosen word groups. Therefore, instead of *Factory payrolls fell in September*, the system would actual be presented with NN NNS VBD IN NN—the finer-grained versions of NOUN NOUN VERB PREPOSITION NOUN used by the Penn Treebank, which also indicate, for instance, verb form and noun plurality. When it adds clarity, we will accompany such tag sequences with specific word examples, usually the most common realization of the sequence in the treebank.

[1] Note that this is *not* what linguists usually mean by *dependency grammars*; they typically consider lexical dependency grammars in which there are many more symbols, one for each word in the lexicon.

Clark [9] describes a method for distributionally clustering multi-unit spans, just as words have traditionally been clustered. In his system, DT NN (*the company*) might be clustered with DT JJ NN (*the third quarter*). Of course, most spans (and therefore most clusters) are not constituents, and so contain sequences like VB IN DT (*comment on the*) and VB DT (*be a*). He uses a second phase to figure out which clusters are the "good" ones and then to decide how to turn such clusters into a grammar. The next section describes our model, which can be seen as a simpler approach which generalizes the word-level distributional approach to a direct tree-structure induction algorithm.

3. A generative constituent-context model

To exploit the benefits of parameter search and the past success of distributional methods, we introduce a novel model which is designed specifically to enable a more felicitous search space. The fundamental assumption is a much weakened version of classic distributional linguistic constituency tests [14]: constituents appear in constituent contexts. A particular linguistic phenomenon that the system exploits is that long constituents often have short, common equivalents, or *proforms* [14], which appear in similar contexts and whose constituency is easily discovered (or guaranteed).[2] Our model is designed to transfer the constituency of a sequence directly to its containing context, which is intended to then pressure new sequences that occur in that context into being parsed as constituents in the next round of the iterative reestimation process. The model is also designed to exploit the successes of distributional clustering, and can equally well be viewed as doing distributional clustering in the presence of no-overlap constraints.

3.1. Constituents and contexts

Unlike a PCFG, our model describes all contiguous subsequences of a sentence (*spans*), including empty spans, regardless of whether they are constituents or non-constituents (*distituents*). A span encloses a (possibly empty) sequence of terminals, or *yield*, α, such as DT JJ NN.[3] A span occurs in a *context* x, such as $\diamond$—VBZ, where x is the ordered pair of preceding and following terminals ($\diamond$ denotes a sentence boundary and VBZ is a present-tense verb like *is*). A *bracketing* of a sentence is a boolean matrix B, which indicates which spans are constituents and which are not. The correspondence between such bracketings (also called parse triangles or well-formed substring tables) is well-known; see for example [33]. Fig. 1 shows a parse of a short sentence, the bracketing corresponding to that parse, and the labels, yields, and contexts of its constituent spans.

Fig. 2 shows several bracketings of the sentence in Fig. 1. A bracketing B of a sentence is *non-crossing* if, whenever two spans cross, at most one is a constituent in B. A non-crossing bracketing is *tree-equivalent* if the size-one terminal spans and the full-sentence span are constituents, and all size-zero spans are distituents. Fig. 2(a) and (b) are tree-equivalent. Tree-equivalent bracketings B correspond to (unlabeled) trees in the obvious way. A bracketing is *binary* if it corresponds to a binary tree. Fig. 2(b) is binary. We will induce trees by inducing tree-equivalent bracketings.

Our basic generative model over sentences S has two phases. First, we choose a bracketing B according to some distribution $\mathrm{P}(B)$ and then generate the sentence given that bracketing:

$$\mathrm{P}(S, B) = \mathrm{P}(B)\mathrm{P}(S|B).$$

Given B, we fill in each span independently. The context and yield of each span are independent of each other, and generated conditionally on the constituency B_{ij} of that span.

$$\begin{aligned}\mathrm{P}(S|B) &= \prod_{\langle i,j\rangle \in \mathrm{spans}(S)} \mathrm{P}(\alpha_{ij}, x_{ij}|B_{ij})\\ &= \prod_{\langle i,j\rangle} \mathrm{P}(\alpha_{ij}|B_{ij})\mathrm{P}(x_{ij}|B_{ij}).\end{aligned}$$

For the main model of the paper, the bracketing just distinguishes constituents from non-constituents. The distribution $\mathrm{P}(\alpha_{ij}|B_{ij})$ is thus a pair of multinomial distributions over the set of all possible yields: one for constituents ($B_{ij}=c$) and one for distituents ($B_{ij}=d$). Similarly, $\mathrm{P}(x_{ij}|B_{ij})$ is a pair of distributions over contexts. The marginal probability assigned to the sentence S is given by summing over all possible bracketings of S: $\mathrm{P}(S) = \sum_B \mathrm{P}(B)\mathrm{P}(S|B)$.[4] Figs. 3 and 4 illustrate the process. These figures show the slightly more general case, which we discuss in Section 4.2, where constituent brackets can have various labels, but distituent brackets all have the single label d.

It is worth emphasizing that in this model spans are nested, and therefore hierarchical structures are learned, but

[2] The classic examples are pronouns (*he, she, it,* etc.), which, being one word, are certainly constituents, but there are other examples such as *there* which can serve as a proform for prepositional phrases. Since pronouns occur in the same positions as larger noun phrases (e.g., subject position between the sentence beginning and the first verb), this interchangeability argues for the constituency of those more complex noun phrases.

[3] Recall that our inputs are parts-of-speech sequences using the Penn Treebank tagset. DT JJ NN stands for determiner–adjective–noun, the most frequent Treebank example of which is *the third quarter*.

[4] Viewed as a model generating *sentences*, this model is deficient, placing mass on yield and context choices which will not tile into a valid sentence, either because specifications for positions conflict or because yields of incorrect lengths are chosen. However, we could renormalize by dividing by the mass placed on proper sentences and zeroing the probability of improper bracketings. The rest of the paper, and results, would be unchanged except for notation to track the renormalization constant.

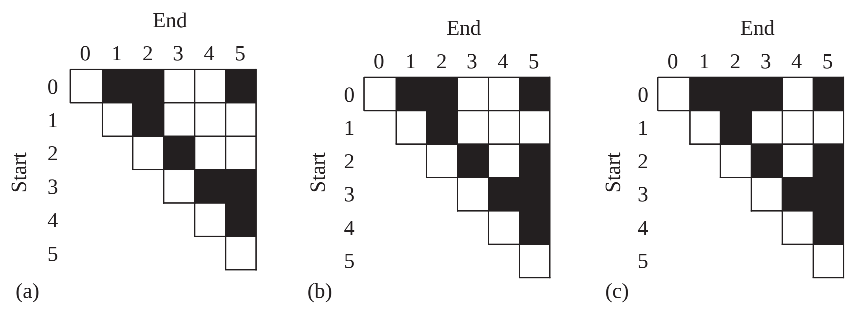


Fig. 2. Three bracketings of the sentence in Fig. 1: constituent spans in black. (b) corresponds to the binary parse in Fig. 1; (a) does not contain the $\langle 2,5\rangle$ VP bracket, while (c) contains a $\langle 0,3\rangle$ bracket crossing that VP bracket.

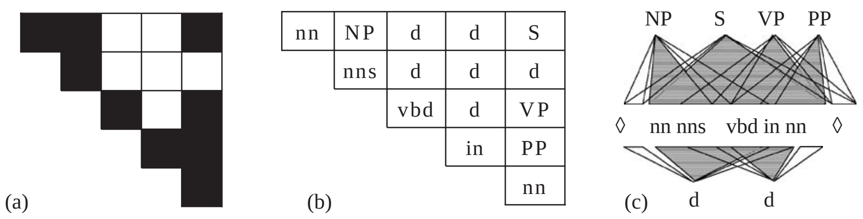
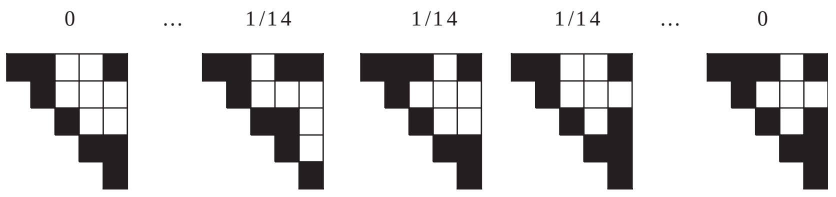


Fig. 3. The generative process. (a) A binary tree-equivalent bracketing is chosen at random. (b) Constituent (black) nodes are labeled with a constituent label, distituent (white) nodes are labeled *d*. In the two-class case, there is only a single consituent label *c*, and so this step can be conceptually omitted. (c) *Each* span generates its yield and context. Derivations which are not coherent are given mass zero.



(a) $P(B)$, bracketing.

L	B=*true*	B=*false*
NP	0.3	0.0
S	0.2	0.0
VP	0.1	0.0
…	…	…
d	0.0	1.0

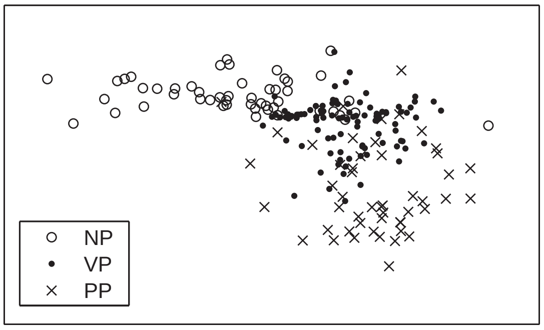
(b) $P(L \mid B)$, labeling.

$P(\alpha\|\text{NP})$		$P(x\|\text{NP})$		$P(\alpha\|d)$	
DT NN	0.10	◇ – VBZ	0.13	IN DT	0.06
DT JJ NN	0.08	VBZ – ◇	0.11	VBZ DT	0.05
PRP	0.07	IN – ◇	0.09	DT JJ	0.04
…	…	…	…	…	…
NNP NNP	0.05	RB – MD	0.03	NNP MD	0.02

(c) $P(\alpha \mid L)$ and $P(x \mid L)$, yields and contexts.

Fig. 4. The probabilities in the model. (a) The distribution over bracketings is always uniform over all binary tree-equivalent bracketings (14 for size 5). (b) The distribution over class labels varies during estimation. (c) The probabilities of different sequences and context also vary. Numbers here are illustrative of what an “ideal” setting might be.

recursion is not explicitly modeled (though some recursive facts are captured by modeling contexts). The model thus identifies an interesting middle-ground where the nested constituents learned do implicitly show the recursive structure of natural language, but do not explicitly encode it in a grammar. While not having an explicit model of recursion may seem to fly in the face of conventional wisdom about representing natural language, it is in fact a substantial part of why this system is so successful: (1) inducing deeply hidden recursive structure makes the search very hard and (2)



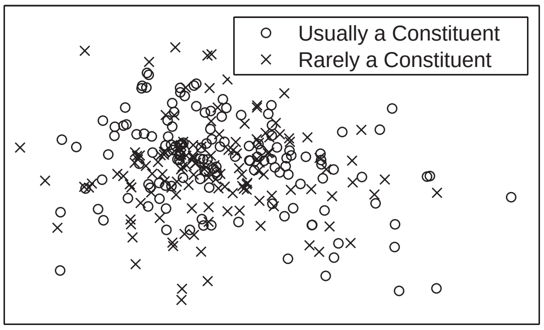


Fig. 5. The most frequent yields of (a) three constituent types and (b) constituents and distituents, as context vectors, projected onto their first two principal components. Clustering is effective at labeling, but not detecting constituents.

context-freedom assumptions have been shown to be wildly false for PCFGs of small to moderate size [34], making their induction by maximum likelihood or MAP methods challenging.

To induce structure, we run EM over this model, treating the sentences S as observed and the bracketings B as unobserved. The parameters Θ of the model are the constituency-conditional yield and context distributions $\mathrm{P}(\alpha|b)$ and $\mathrm{P}(x|b)$, and the possible bracketings $\mathrm{P}(B)$. However, as discussed below, we keep $\mathrm{P}(B)$ fixed in our model.[5]

If $\mathrm{P}(B)$ is uniform over all (possibly crossing) bracketings, then this procedure will be equivalent to soft-clustering with two equal-prior classes. There is reason to believe that such soft clusterings alone will not produce valuable distinctions, even if we introduce a large number of constituent classes. The distituents must necessarily outnumber the constituents, and so such distributional clustering will result in mostly distituent classes. Clark [9] finds exactly this effect, and must resort to a filtering heuristic to separate constituent and distituent clusters. To underscore the difference between the bracketing and labeling tasks, consider Fig. 5. In both plots, each point is a frequent tag sequence, assigned to the (normalized) vector of its context frequencies. Each plot has been projected onto the first two principal components of its respective data set. The left plot shows the most frequent sequences of three constituent types. Even in just two dimensions, the clusters seem coherent, and it is easy to believe that they would be found by a clustering algorithm in the full space. On the right, sequences have been labeled according to whether their occurrences are constituents more or less of the time than a cutoff (of 0.2). These pictures are suggestive rather than being proofs, but all the evidence we have suggests that the distinction between constituent and distituent seems much less easily discernible than the distinction between various categories.

We can turn what at first seems to be distributional clustering into tree induction by confining $\mathrm{P}(B)$ to put mass only on tree-equivalent bracketings. In particular, consider $\mathrm{P}_{\text{bin}}(B)$ which is uniform over binary bracketings and zero elsewhere (illustrated in Fig. 4(a)). If we take this bracketing distribution, then when we sum over data completions, we will only aggregate over bracketings which correspond to valid binary trees. This restriction is the basis for our algorithm.

3.2. The induction algorithm

We now essentially have our induction algorithm. We take the prior $\mathrm{P}(B)$ to be $\mathrm{P}_{\text{bin}}(B)$, so that all binary trees are equally likely. The $\mathrm{P}(B)$ term is *not* learned or otherwise varied from $\mathrm{P}_{\text{bin}}(B)$ during convergence. We then apply the EM algorithm:

E-Step. Find the conditional completion likelihoods $\mathrm{P}(B|S, \Theta)$ according to the current Θ.

M-Step. Fixing $\mathrm{P}(B|S, \Theta)$, find the Θ' which maximizes $\sum_B \mathrm{P}(B|S, \Theta) \log \mathrm{P}(S, B|\Theta')$.

The completions (bracketings) cannot be efficiently enumerated, and so a cubic dynamic program similar to the inside–outside algorithm is used to calculate the expected counts of each yield and context, both as constituents and distituents. Relative frequency estimates (which are the ML estimates for this model) are used to set Θ'.

To begin the process, we did not (as is usual) begin at the E-step with an initial guess at Θ. Rather, we began at the M-step, using an initial guess at a distribution over completions $\mathrm{P}(B|S, \Theta)$. This was because in our model we did not have a ready estimate of a mean or neutral value of Θ. One option for an initial guess over completions $\mathrm{P}(B|S, \Theta)$ was the prior, uniform distribution over binary trees $\mathrm{P}_{\text{bin}}(B)$. That

[5] Learning $\mathrm{P}(B)$ as well is a simple extension to the model, but requires some parameterization of bracketing distributions, because the space of bracketings is extremely large. Learning $\mathrm{P}(B)$ could, for example, allow the model to capture right- or left-branching tendencies. However, in our experiments, parameterizations of $\mathrm{P}(B)$ in terms of branching tendencies became self-reinforcing, resulting in entirely left- or right-branching structure.

was undesirable as an initial posterior because, combinatorialy, almost all trees are relatively balanced. On the other hand, in language, we want to allow unbalanced structures to have a reasonable chance of being discovered. Therefore, consider the following uniform-splitting process of generating binary trees over k terminals: choose a split point at random, then recursively build trees by this process on each side of the split. This process gives a distribution P_{split} which puts relatively more weight on unbalanced trees, but only in a very general, non language-specific way. This distribution was not used as $P(B)$ in the model itself, however. It seemed to bias too strongly *against* balanced structures, and led to entirely linear-branching posterior structures.

Probability distributions built from natural language data typically require smoothing to cope with data sparseness. The smoothing used was straightforward. For each yield α or context x, we added 10 counts of that item as a constituent and 50 as a distituent. This reflected the relative skew of random spans being more likely to be distituents.[6]

4. Experiments

We performed most experiments on the 7422 sentences in the Penn treebank Wall Street Journal section which contained no more than 10 words after the removal of punctuation and null elements (WSJ-10). For most experiments, we take treebank part-of-speech sequences as input, but in Section 4.3, we use distributionally induced tags as input. Performance with induced tags is somewhat reduced, but still gives better performance than previous models. Evaluation was done by seeing whether proposed constituent spans are also in the treebank parse, measuring unlabeled constituent precision, recall, and their harmonic mean F_1. Constituents which could not be gotten wrong (namely those of span one and those spanning entire sentences) were discarded.[7] The basic experiments, as described above, do not label constituents. An advantage to having only a single constituent class is that it encourages constituents of one type to be found even when they occur in a context which canonically holds another type. For example, NPS and PPs both occur between a verb and the end of the sentence, and they can transfer constituency to each other through that context.

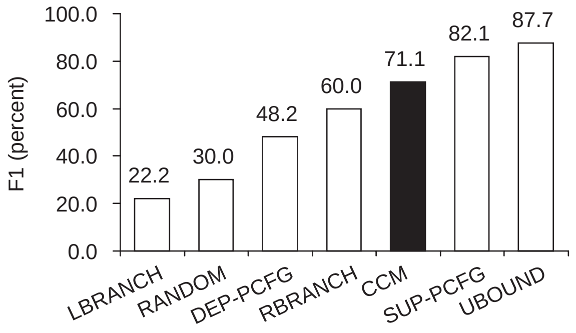


Fig. 6. F_1 for various models on WSJ-10.

Fig. 6 shows the F_1 score for various methods of parsing. RANDOM chooses a binary tree uniformly at random from the set of binary trees.[8] This is the unsupervised baseline. DEP-PCFG is the result of duplicating the experiments of Carroll and Charniak [5], using EM to train a dependency-structured PCFG. LBRANCH and RBRANCH choose the completely left- and right-branching structures, respectively. RBRANCH is a frequently used baseline for supervised parsing, but it should be stressed that it encodes a significant fact about English structure, and an induction system need not beat it to claim a degree of success. CCM is our system, as described above. SUP-PCFG is a *supervised* PCFG parser trained on a 90-10 split of this data, using the treebank grammar, with the Viterbi (maximum posterior likelihood) parses right-binarized.[9] UBOUND is the upper bound on how well a binary system can do against the treebank sentences. The treebank sentences are generally flatter than binary, hence limiting the maximum precision which can be attained, since additional brackets added to provide a binary tree will be counted as wrong.

The CCM does extremely well at 71.1%, much better than right-branching structure. One common issue with grammar induction systems is a tendency to chunk in a bottom-up fashion. Especially since the CCM does not model recursive structure explicitly, one might be concerned that the high overall accuracy is due to a high accuracy on short-span constituents. Fig. 7 shows that this is not true. Recall drops slightly for mid-size constituents, but longer constituents are as reliably proposed as short ones. Another effect illustrated

[6] The required smoothing here contrasts with our earlier discriminative model of Klein and Manning [35], which was sensitive as to smoothing method, and required a massive amount of it.

[7] Since reproducible evaluation is important, a few more notes: this is different from the original (unlabeled) bracketing measures proposed in the PARSEVAL standard [36], which did not count single words as constituents, but did give points for putting a bracket over the entire sentence. Secondly, bracket labels and multiplicity are just ignored. Finally, our evaluation macro-averaged statistics per-sentence, rather than micro-averaging per-bracket. The F_1 scores we report are slightly lower for having discarded the trivial sentence-level bracket, but slightly higher for having macro-averaged; the net effect was small (and slightly negative). Below, we also present results using the EVALB program for comparability. The EVALB program from http://nlp.cs.nyu.edu/evalb/ is commonly used as a standard implementation of these metrics for supervised parser evaluation. But we note that while one can get results from it that ignore bracket labels, it never ignores bracket multiplicity, nor does it ignore span-one brackets. These two constraints seem less satisfactory to us as measures for evaluating unsupervised constituency decisions.

[8] This choice corresponds to choosing from $P_{\text{bin}}(B)$, and is different from making random parsing decisions, corresponding to $P_{\text{split}}(B)$, which gave a higher score of 34.8%.

[9] Without post-binarization, the F_1 score was 88.9.

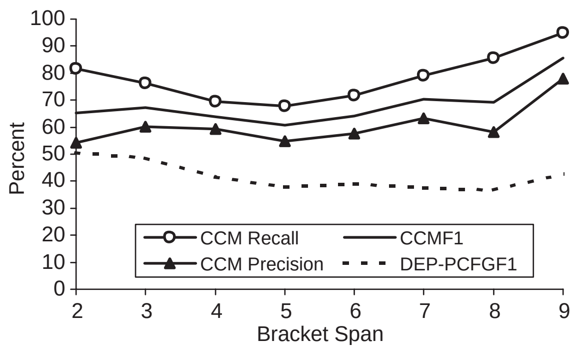


Fig. 7. Accuracy scores for CCM-induced structures by span size. The drop in precision for span length 2 is largely due to analysis inside NPs which is omitted by the treebank. Also shown is F_1 for the induced PCFG. The PCFG shows higher accuracy on small spans, while the CCM is more even.

System	UP	UR	F_1	CB
EMILE	51.6	16.8	25.4	**0.84**
ABL	43.6	35.6	39.2	2.12
CDC-40	53.4	34.6	42.0	1.46
RBRANCH	39.9	46.4	42.9	2.18
COND-CCM	54.4	46.8	50.3	1.61
CCM	**55.4**	**47.6**	**51.2**	1.45

Fig. 8. Comparative ATIS parsing results (unlabeled precision (UP) and unlabeled recal (UR), as scored by evalb). This system (CCM) outperforms the best previously published systems' numbers. EMILE is described in [18], ABL in [11], and CDC-40 in [9]. RBRANCH is the right-branching (supervised) baseline. COND-CCM is a previous conditional formulation of our model, described in [35].

in this graph is that, for span 2, constituents have low precision for their recall. This contrast is primarily due to the single largest difference between the system's induced structures and those in the treebank: the treebank does not parse into NPs such as DT JJ NN, while our system does, and generally does so correctly, identifying $\overline{\text{N}}$ units like JJ NN. This overproposal drops span-2 precision. In contrast, Fig. 7 also shows the F_1 for DEP-PCFG, which does exhibit a drop in F_1 over larger spans.

The top row of Fig. 10 shows the recall of non-trivial brackets, split according the brackets' labels in the treebank. Unsurprisingly, NP recall is highest, but other categories are also high. Because we ignore trivial constituents, the comparatively low S represents *only* embedded sentences, which are harder even for supervised systems.

To facilitate comparison with other recent work, Fig. 8 shows the accuracy of both our system and some other recent systems tested on the ATIS corpus, and evaluated according to the EVALB program.[10] The F_1 numbers are lower for this corpus and evaluation method.[11] Still, CCM beats not only RBRANCH (by 8.3%), but also our previous conditional COND-CCM [35] and the next closest unsupervised system (which does not beat RBRANCH in F_1).

4.1. Error analysis

Parsing figures can only be a component of evaluating an unsupervised induction system. Low scores may indicate systematic alternate analyses rather than true confusion, and the Penn treebank is a sometimes arbitrary or even inconsistent gold standard. To give a better sense of the kinds of errors the system is or is not making, we can look at which sequences are most often over-proposed, or most often under-proposed, compared to the treebank parses.

Fig. 9 shows the 10 most frequently over- and under-proposed sequences. The system's main error trends can be seen directly from these two lists. It forms MD VB verb groups systematically, and it attaches the possessive particle to the right, like a determiner, rather than to the left.[12] It provides binary-branching analyses within NPs, normally resulting in correct extra $\overline{\text{N}}$ constituents, like JJ NN, which are not bracketed in the treebank. In this case, the system's analysis is widely held to be superior to the treebank's, which was made extremely flat by design. However, not all differences represent acceptable alternate analyses. Most seriously, the system tends to attach post-verbal prepositions to the verb and it gets confused by long sequences of nouns (splitting them into multiple separate noun phrases). A significant improvement over earlier systems is the absence of subject-verb groups, which disappeared when we switched to $P_{\text{split}}(B)$ for initial completions; the more balanced subject-verb analysis had a substantial combinatorial advantage with $P_{\text{bin}}(B)$.

4.2. Multiple constituent classes

We also ran the system with multiple constituent classes, using a slightly more complex generative model in which the bracketing generates a labeling which then generates the constituents and contexts. The sets of labels for constituent spans and distituent spans are forced to be disjoint (see Figs. 3 and 4).

[10] Our system was trained on the same WSJ data, and then tested on ATIS. EMILE and ABL are lexical systems described in [11,18]. CDC-40, from [9], reflects training on much more data (12M words).

[11] The primary cause of the lower F_1 is that the ATIS corpus is replete with span-one NPs; adding an extra bracket around *all* single words raises our EVALB recall to 71.9; removing all unaries from the ATIS gold standard gives an F_1 of 63.3%.

[12] Linguists have at times argued for both analyses: Halliday [37] and Abney [38], respectively.

	Overproposed		Underproposed	
Rank	Type	Example	Type	Example
1	JJ NN	last year	NNP POS	Friday's
2	MD VB	will be	TO CD CD	to 1 million
3	DT NN	a share	NN NNS	interest rates
4	NNP NNP	New York	NN NN	vice president
5	RB VB	n't be	TO VB	to be
6	JJ NNS	common shares	IN CD	in 1988
7	NNP NN	Bush administration	NNP NNP POS	Wall Street's
8	RB VBN	n't been	DT NN POS	the company's
9	IN NN	in addition	RB CD	only one
10	POS NN	's stock	IN DT	of the (of that)

Fig. 9. Constituent yields most frequently over- and under-proposed by our system. Also shown are the most frequent example of that sequence, for concreteness.

Classes	Tags	Precision	Recall	F_1	NP Recall	PP Recall	VP Recall	S Recall
2	Treebank	**63.8**	**80.2**	**71.1**	**83.4**	**78.5**	78.6	40.7
12	Treebank	63.6	80.0	70.9	82.2	59.1	82.8	57.0
2	Induced	56.8	71.1	63.2	52.8	56.2	**90.0**	**60.5**

Fig. 10. Scores for the 2- and 12-class model with Treebank tags, and the 2-class model with induced tags.

Intuitively, it seems that more classes should help, by allowing the system to distinguish different types of constituents and constituent contexts. However, it seemed to slightly hurt parsing accuracy overall. Fig. 10 compares the performance for 2 versus 12 classes; in both cases, only one of the classes was allocated for distituents. Overall F_1 dropped very slightly with 12 classes, but the category recall numbers indicate that the errors shifted around substantially. PP accuracy is lower, which is not surprising considering that PPs tend to appear rather optionally and in contexts in which other, easier categories also frequently appear. On the other hand, embedded sentence recall is substantially higher, possibly because of more effective use of the top-level sentences which occur in the signature context $\diamond$–$\diamond$.

The classes found, as might be expected, range from clearly identifiable to nonsense. Note that simply directly clustering all sequences into 12 categories produced almost entirely the latter, with clusters representing various distituent types. Fig. 11 shows several of the 12 classes. Class 0 is the model's distituent class. Its most frequent members are a mix of obvious distituents (IN DT, DT JJ, IN DT, NN VBZ) and seemingly good sequences like NNP NNP. However, there are many sequences of 3 or more NNP tags in a row, and not all adjacent pairs can possibly be constituents at the same time. Class 1 is mainly common NP sequences, class 2 is proper NPs, class 3 is NPs which involve numbers, and class 6 is $\overline{\text{N}}$ sequences, which tend to be linguistically right but unmarked in the treebank. Class 4 is a mix of seemingly good NPs, often from positions like VBZ–NN where they were *not* constituents, and other sequences that share such contexts with otherwise good NP sequences. This is a danger of not jointly modeling yield and context, and of not modeling any kind of recursive structure. Class 5 is mainly composed of verb phrases and verb groups. No class corresponded neatly to PPs: perhaps because they have no signature contexts. The 2-class model is effective at identifying them only because they share contexts with a range of other constituent types (such as NPs and VPs). Given the success of Clark [9] in clustering sequences without bracketing constraints, it is likely that the multi-class model can be made to be superior to the two-class model; however this would likely require a degree of tying of probabilities across classes.

4.3. *Induced parts-of-speech*

A reasonable criticism of the experiments presented so far, is that we assume treebank part-of-speech tags as input. This criticism could be two-fold. First, state-of-the-art supervised PCFGs do not perform nearly so well with their

Class 0				Class 1	
NNP NNP	New York	NN VBD	company said	DT NN	the company
NN IN	percent of	NN NN	vice president	JJ NNS	common shares
IN DT	of the	NNS VBP	companies are	DT NNS	the shares
DT JJ	the first	NNS VBD	officials said	DT JJ NN	the third quarter
NN VBZ	company is	TO VB	to be	NN NNS	interest rates

Class 2		Class 3	
NNP NNP	New York	CD CD	100 million
NNP NNP NNP	New York Stock (Exchange)	CD NN	10 percent
CC NNP	& Co.	IN CD CD	of 2 1/2
POS NN	's stock	CD NNS	million shares
NNP NNP NNP NNP	New York Stock Exchange	CD CD IN CD CD	200 million of 8 7/8

Class 4		Class 5		Class 6	
VBN IN	compared with	MD VB	will be	JJ NN	last year
JJ IN	such as	MD RB VB	wo n't be	JJ NNS	common shares
DT NN	the company	VBN IN	compared with	JJ JJ NN	chief executive officer
JJ CC	economic and	WDT VBZ	which is	CD NNS	million shares
DT JJ NN	the third quarter	JJ IN	such as	NNP NN	Bush administration

Fig. 11. Most frequent members of several classes found.

input delexicalized. We may be reducing data sparsity and making it easier to see a broad picture of the grammar, but we are also limiting how well we can possibly do. It is certainly worth exploring methods which supplement or replace tagged input with lexical input. However, we address here the more serious criticism: that our results stem from clues latent in the treebank tagging information which are conceptually posterior to knowledge of structure. For instance, some Penn Treebank tag distinctions, such as particle (RP) vs. preposition (IN) or predeterminer (PDT) vs. determiner (DT) or adjective (JJ), could be said to import into the tag set distinctions that can only be made syntactically.

To show results from a complete grammar induction system, we also did experiments starting with a clustering of the words in the treebank. We used basically the baseline method of word type clustering in [31] (which is close to the methods of Finch [39]). For (all-lowercased) word types in the Penn treebank, a 1000 element vector was made by counting how often each co-occurred with each of the 500 most common words immediately to the left or right in Treebank text and additional 1994–96 WSJ newswire. These vectors were length-normalized, and then rank-reduced by an SVD, keeping the 50 largest singular vectors. The resulting vectors were clustered into 200 word classes by a weighted *k*-means algorithm, and then grammar induction operated over these classes. We do not believe that the quality of our tags matches that of the better methods of Schütze [31], much less the recent results of Clark [32]. Nevertheless, using these tags as input still gave induced structure substantially above right-branching. Fig. 10 shows the performance with induced tags compared to correct tags. Overall F_1 has dropped, but, interestingly, VP and S recall are higher. This seems to be due to a marked difference between the induced tags and the treebank tags: nouns are scattered among a disproportionally large number of induced tags, increasing the number of common NP sequence types, but decreasing the frequency of each.

	English			German			Chinese		
	UP	UR	F_1	UP	UR	F_1	UP	UR	F_1
LBRANCH	20.5	24.2	22.2	31.1	41.0	35.4	19.4	25.5	22.0
RBRANCH	54.1	67.5	60.0	38.8	56.2	45.9	26.1	31.3	28.4
CCM	63.8	80.2	71.1	52.0	83.8	64.2	32.7	42.2	36.8
UBOUND	78.3	100.0	87.7	60.2	100.0	75.2	75.5	100.0	86.1

Fig. 12. Performance of the model on various languages. English data: 7422 sentences from the Penn treebank; German data: 1438 sentences from the NEGRA corpus; Chinese data 443 sentences from the Chinese treebank.

4.4. *Extension to other human languages*

A grammar induction system should be as language-independent as possible. To test how well our system would work for languages other than English, we ran it

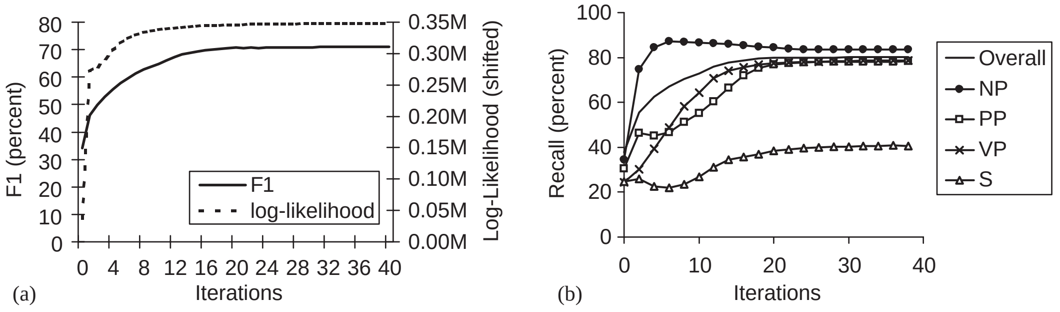


Fig. 13. Overall F_1 is non-decreasing until convergence (a) but per-type F_1 is not necessarily so (b).

essentially unchanged on the two other bracketed treebanks we had available: the NEGRA corpus (German, [40]) and the Chinese Treebank [41]. The NEGRA corpus had 1438 sentences of length less than 10 after removal of punctuation. Smoothing was proportionally reduced, and the system was otherwise run identically to the English experiments. The results are shown in Fig. 12. The German right-branching baseline is lower than for English, at 45.9%, due to the more variable branching behavior of German. Since the German gold parses are flatter than the English ones, the upper bound for binary branching hypotheses is lower, as well, at 75.2%, and the number of training sentences is much smaller. Nonetheless, the system learns structures which are substantially above right branching, scoring 64.2%. At the time of these experiments, the Chinese treebank had only 443 sentences of length up to 10 which had non-empty parses. On this data, the performance was substantially less impressive, with the system scoring only 36.8%, above the right-branching baseline of 28.4%, but far below the upper bound of 86.1%. Worse, unlike for the German and English, convergence in model likelihood did not correspond to maximum F_1. The relatively poor performance on the Chinese data is partially attributable to the much smaller training size, but also likely reflects a real weakness in the system. One of the present model's strengths is its proper identification of the role of function words, for example, grouping determiners with their following nouns. This kind of behavior is much less useful for languages like Chinese, which has few such function words.

4.5. Convergence and stability

A serious issue with previous systems has been their sensitivity to initial search choices and initial parameters. For example, the conditional model of Klein and Manning [35] had the drawback that the variance of final F_1, and qualitative grammars found, was fairly high, depending on small differences in first-round random parses. The model presented here does not suffer from this: while it is clearly sensitive to the quality of the input tagging, it is robust with respect to smoothing parameters and data splits. Varying the smoothing counts a factor of ten in either direction did not change the overall F_1 by more than 1%. Training on random subsets of the training data brought lower performance, but constantly lower over equal-size splits. Moreover, there are no first-round random decisions to be sensitive to; the soft EM procedure is deterministic.

Fig. 13 shows the overall F_1 score and the data likelihood according to our model during convergence.[13] Surprisingly, both are non-decreasing as the system iterates, indicating that data likelihood in this model corresponds well with parse accuracy.[14] Fig. 13 shows recall for various categories by iteration. NP recall exhibits the more typical pattern of a sharp rise followed by a slow fall, but the other categories, after some initial drops, all increase until convergence. These graphs stop at 40 iterations. The system actually converged in both likelihood and F_1 by iteration 38, to within a tolerance of 10^{-10}. The time to convergence varied according to amount of smoothing, number of classes, and tags used, but the system almost always converged within 80 iterations, usually within 40.

4.6. Partial supervision

For practical applications, such as building initial treebanks, there is not a great difference between a totally unsupervised system and a weakly supervised one, in terms of resources required. To test the effect of partial supervision, we supplied the system with treebank parses for various fractions of 90% of the training data, and tested on the remaining 10%. During training, analyses which crossed the brackets of the supervision parses were given zero weight. Fig. 14 shows F_1 on the held-out 10% as supervision percent increased. Accuracy goes up initially, though it drops slightly at very high supervision levels. The interesting conclusion from this graph is that, while this model does not parse as well in a supervised setting as a PCFG, it performs better in an unsupervised setting, and its performance is

[13] The data likelihood is not shown exactly, but rather we show the linear transformation of it calculated by the system.

[14] Pereira and Schabes [6] find otherwise for PCFGs.

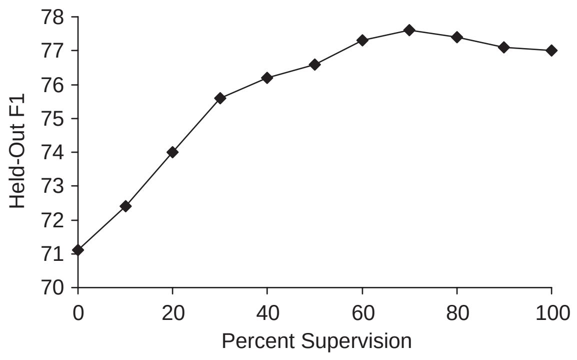


Fig. 14. Partial supervision.

surprisingly close to its supervised limit in that unsupervised setting.

5. Conclusions

We have presented a simple generative model for the unsupervised distributional induction of hierarchical linguistic structure which does not explicitly model recursion. The system achieves the best published unsupervised parsing scores on the WSJ-10 and ATIS data sets. The induction algorithm combines the benefits of EM-based parameter search and distributional clustering methods. We have shown that this method acquires a substantial amount of correct structure, to the point that the most frequent discrepancies between the induced trees and the treebank gold standard are systematic alternate analyses, many of which are linguistically plausible. We have shown that the system is not reliant on supervised POS tag input, and demonstrated increased accuracy, speed, simplicity, and stability compared to previous systems.

References

[1] S.M. Lamb, On the mechanisation of syntactic analysis, in: 1961 Conference on Machine Translation of Languages and Applied Language Analysis, vol. 2, Her Majesty's Stationery Office, London, 1961, pp. 674–685.

[2] J.G. Wolff, Grammar discovery as data compression, in: D.H. Sleeman (Ed.), Proceedings of AISB/GI Conference (Proceedings of the 4th European Conference on Artificial Intelligence), Hamburg, Germany, July 1978, pp. 375–379.

[3] P. Langley, A production system model of first language acquisition, in: Proceedings of the Eighth International Conference on Computational Linguistics, Tokyo, Japan, 1980, pp. 183–189.

[4] J.G. Wolff, Learning syntax and meanings through optimization and distributional analysis, in: Y. Levy, I.M. Schlesinger, M.D.S. Braine (Eds.), Categories and Processes in Language Acquisition, Lawrence Erlbaum, Hillsdale, NJ, 1988, pp. 179–215.

[5] G. Carroll, E. Charniak, Two experiments on learning probabilistic dependency grammars from corpora, in: C. Weir, S. Abney, R. Grishman, R. Weischedel (Eds.), Working Notes of the Workshop Statistically-Based NLP Techniques, AAAI Press, 1992, pp. 1–13.

[6] F. Pereira, Y. Schabes, Inside-outside reestimation from partially bracketed corpora, in: ACL 30, 1992, pp. 128–135.

[7] E. Brill, Automatic grammar induction and parsing free text: a transformation-based approach, in: ACL 31, 1993, pp. 259–265.

[8] A. Stolcke, S.M. Omohundro, Inducing probabilistic grammars by Bayesian model merging, in: Grammatical Inference and Applications: Proceedings of the Second International Colloquium on Grammatical Inference, Springer, Berlin, 1994.

[9] A. Clark, Unsupervised induction of stochastic context-free grammars using distributional clustering, in: The Fifth Conference on Natural Language Learning, 2001.

[10] M. Selfridge, A computer model of child language learning, in: Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, BC, 1981, pp. 92–96.

[11] M. van Zaanen, ABL: Alignment-based learning, in: COLING 18, 2000, pp. 961–967. URL citeseer.nj.nec.com/432996.html.

[12] J.K. Baker, Trainable grammars for speech recognition, in: D.H. Klatt, J.J. Wolf (Eds.), Speech Communication Papers for the 97th Meeting of the Acoustical Society of America, 1979, pp. 547–550.

[13] S.F. Chen, Bayesian grammar induction for language modeling, in: ACL 33, 1995, pp. 228–235.

[14] A. Radford, Transformational Grammar, Cambridge University Press, Cambridge, 1988.

[15] Z. Harris, Methods in Structural Linguistics, University of Chicago Press, Chicago, 1951.

[16] E.M. Gold, Language identification in the limit, Inform. Control 10 (1967) 447–474.

[17] J.J. Horning, A study of grammatical inference, Ph.D. Thesis, Stanford, 1969.

[18] P. Adriaans, E. Haas, Grammar induction as substructural inductive logic programming, in: J. Cussens (Ed.), Proceedings of the First Workshop on Learning Language in Logic, Bled, Slovenia, 1999, pp. 117–127.

[19] K.J. Lang, B.A. Pearlmutter, R.A. Price, Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm, in: Proceedings of the Fourth International Colloquium on Grammatical Inference, of Lecture Notes in Artificial Intelligence, vol. 1433, Springer, Berlin, 1998, pp. 1–12.

[20] N. Abe, M. Warmuth, On the computational complexity of approximating distributions by probabilistic automata, in: Proceedings of the Third Workshop on Computational Learning Theory, Morgan Kaufmann, 1990, pp. 52–66, URL citeseer.nj.nec.com/article/abe90computational.html.

[21] M.J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire, L. Sellie, On the learnability of discrete distributions, in: Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1994, pp. 273–282.

[22] E. Charniak, Tree-bank grammars, in: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI '96), 1996, pp. 1031–1036.

[23] M.J. Collins, Three generative, lexicalised models for statistical parsing, in: ACL 35/EACL 8, 1997, pp. 16–23.

[24] P. Langley, A model of early syntactic development, in: Proceedings of the 20th Annual Conference of the Society for Computational Linguistics, Toronto, Canada, 1982, pp. 145–151.
[25] D.C. Olivier, Stochastic Grammars and language acquisition mechanisms, Ph.D. Thesis, Harvard University, 1968.
[26] K. Lari, S.J. Young, The estimation of stochastic context-free grammars using the inside-outside algorithm, Comput. Speech Language 4 (1990) 35–56.
[27] C. de la Higuera, J. Oncina, E. Vidal, Identification of DFA: Data-dependent versus data-independent algorithms, Proceedings of the International Colloquium on Grammatical Inference (ICGI-96), Lecture Notes in Artificial Intelligence, vol. 1147, Springer, Berlin, 1996, pp. 313–325.
[28] D. Klein, C.D. Manning, Distributional phrase structure induction, in: Proceedings of the Fifth Conference on Natural Language Learning (CoNLL 2001), 2001, pp. 113–120.
[29] M.P. Marcus, B. Santorini, M. Ann Marcinkiewicz, Building a large annotated corpus of English: The Penn treebank, Comput. Linguist. 19 (1993) 313–330.
[30] S. Finch, N. Chater, Bootstrapping syntactic categories using statistical methods, in: W. Daelemans, D. Powers (Eds.), Background and Experiments in Machine Learning of Natural Language, Tilburg University, Institute for Language Technology and AI, 1992, pp. 229–235.
[31] H. Schütze, Distributional part-of-speech tagging, in: EACL, vol. 7, 1995, pp. 141–148.
[32] A. Clark, Inducing syntactic categories by context distribution clustering, in: The Fourth Conference on Natural Language Learning, 2000.
[33] G. Gazdar, C. Mellish, Natural language processing in LISP, Addison-Wesley, Wokingham, England, 1989.
[34] M. Johnson, PCFG models of linguistic tree representations, Comput. Linguist. 24 (1998) 613–632.
[35] D. Klein, C.D. Manning, Natural language grammar induction using a constituent-context model, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14 (NIPS 2001), vol. 1, MIT Press, Cambridge, MA, 2001, pp. 35–42.
[36] E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, T. Strzalkowski, A procedure for quantitatively comparing the syntactic coverage of English grammars, in: Proceedings, Speech and Natural Language Workshop, Pacific Grove, CA, DARPA, 1991, pp. 306–311.
[37] M.A.K. Halliday, An Introduction to Functional Grammar, second ed., Edward Arnold, London, 1994.
[38] S.P. Abney, The English noun phrase in its sentential aspect, Ph.D. Thesis, MIT, Cambridge, MA, 1987.
[39] S.P. Finch, Finding structure in language, Ph.D. Thesis, University of Edinburgh, 1993.
[40] W. Skut, B. Krenn, T. Brants, H. Uszkoreit, An annotation scheme for free word order languages, in: Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97), 1997.
[41] N. Xue, F.-D. Chiou, M. Palmer, Building a large-scale annotated Chinese corpus, in: Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002), 2002.

About the Author—DAN KLEIN is a Ph.D. student in computer science at Stanford University, working on unsupervised language induction and large-scale machine learning for NLP, including statistical parsing, information extraction, fast inference in large dynamic programs, and automatic clustering. He holds a BA from Cornell University (summa cum laude in computer science, linguistics, and math) and a masters in linguistics from Oxford University. Dan's recent academic honors include a British Marshall Fellowship, a Microsoft Graduate Research Fellowship, and a Stanford University Graduate Fellowship.

About the Author—CHRISTOPHER MANNING is an assistant professor of computer science and linguistics at Stanford University. Prior to this, he received his BA (Hons) from the Australian National University, his Ph.D. from Stanford in 1995, and held faculty positions at Carnegie Mellon University and the University of Sydney. His research interests include probabilistic natural language processing, syntax, parsing, computational lexicography, information extraction and text mining. He is the author of three books, including the well-known text Foundations of Statistical Natural Language Processing (MIT Press, 1999, with Hinrich Schuetze).