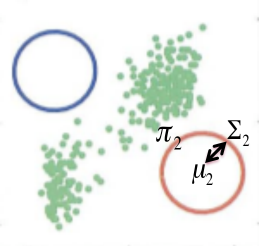
Lecture 4: Constituent-Context Model

Jelle Zuidema
ILLC, Universiteit van Amsterdam

Unsupervised Language Learning, 2014

EM FOR GAUSSIAN MIXTURES

1. Initialize μ_k , Σ_k π_k , one for each Gaussian k



- Tip! Use K-means result to initialize:

$$\mu_k \leftarrow \mu_k$$

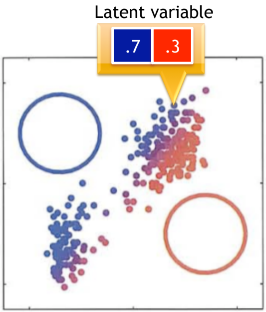
$$\Sigma_k \leftarrow \mathrm{cov}(cluster(K))$$

$$\pi_k \leftarrow \frac{\text{Number of points in k}}{\text{Total number of points}}$$

EM FOR GAUSSIAN MIXTURES

2. **E Step:** For each point X_n, determine its assignment score to each Gaussian k:

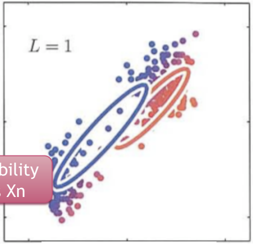
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$



$\gamma(z_{nk})$ is called a “responsibility”: how much is this Gaussian k responsible for this point X_n?

EM FOR GAUSSIAN MIXTURES

3. **M Step:** For each Gaussian k, update parameters using new $\gamma(z_{nk})$



Responsibility for this Xn

Mean of Gaussian k

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

Find the mean that “fits” the assignment scores best

GENERAL EM ALGORITHM

1. Initialize parameters θ^{old}
2. **E Step:** Evaluate $p(Z \mid X, \theta^{old})$
3. **M Step:** Evaluate

Hidden variables

Observed variables

$$\theta^{\text{new}} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{\text{old}})$$

where

Likelihood

$$\mathcal{Q}(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta).$$

4. Evaluate log likelihood. If likelihood or parameters converge, stop. Else $\theta^{old} \leftarrow \theta^{new}$ and go to E Step.

Efficient EM for PCFGs: Inside-outside algorithm

- Hidden variables: the derivations that generated the observed sentences

Efficient EM for PCFGs: Inside-outside algorithm

- Hidden variables: the derivations that generated the observed sentences
- E-step: compute the probability distribution over possible derivations of all the sentences in the corpus;

Efficient EM for PCFGs: Inside-outside algorithm

- Hidden variables: the derivations that generated the observed sentences
- E-step: compute the probability distribution over possible derivations of all the sentences in the corpus;
- M-step: maximum likelihood settings of parameters (rule probabilities) given the current estimate of the derivations;

Efficient EM for PCFGs: Inside-outside algorithm

- Hidden variables: the derivations that generated the observed sentences
- E-step: compute the probability distribution over possible derivations of all the sentences in the corpus;
- M-step: maximum likelihood settings of parameters (rule probabilities) given the current estimate of the derivations;
- For the M-step, we only need to know the *expected relative frequency* of each rule in the derivation of the sentences;

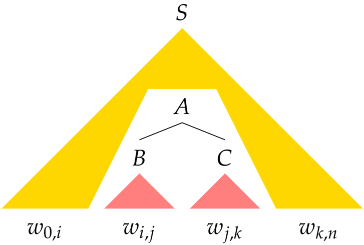
Efficient EM for PCFGs: Inside-outside algorithm

- Hidden variables: the derivations that generated the observed sentences
- E-step: compute the probability distribution over possible derivations of all the sentences in the corpus;
- M-step: maximum likelihood settings of parameters (rule probabilities) given the current estimate of the derivations;
- For the M-step, we only need to know the *expected relative frequency* of each rule in the derivation of the sentences;
- *expected relative frequency* of rules can be computed without computing the entire probability distribution over derivations!

Dynamic programming for $E_p(f_{A \to B\,C} | w)$

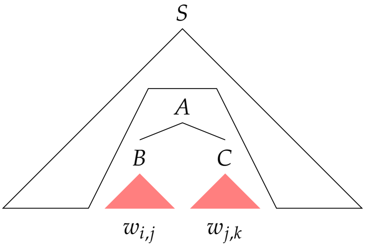
$$E_p(f_{A \to B\,C} | w) =$$

$$\frac{\sum_{0 \le i < j < k \le n} \mathrm{P}(S \Rightarrow^* w_{1,i}\, A\, w_{k,n})\, p(A \to B\,C)\, \mathrm{P}(B \Rightarrow^* w_{i,j})\, \mathrm{P}(C \Rightarrow^* w_{j,k})}{\mathrm{P}_G(w)}$$



Dynamic programming recursion

$$\begin{aligned}&\mathrm{P}_G(A \Rightarrow^* w_{i,k})\\ &\quad = \sum_{j=i+1}^{k-1} \sum_{A \rightarrow B\, C \in R(A)} p(A \rightarrow B\, C)\mathrm{P}_G(B \Rightarrow^* w_{i,j})\mathrm{P}_G(C \Rightarrow^* w_{j,k})\end{aligned}$$



$\mathrm{P}_G(A \Rightarrow^* w_{i,k})$ is called the *inside probability* of A spanning $w_{i,k}$.

Language modeling using dynamic programming

- *Goal:* To compute $\mathrm{P}_G(w) = \sum_{\psi \in \Psi_G(w)} \mathrm{P}_G(\psi) = \mathrm{P}_G(S \Rightarrow^* w)$
- *Data structure:* A table called a *chart* recording $\mathrm{P}_G(A \Rightarrow^* w_{i,k})$ for all $A \in N$ and $0 \leq i < k \leq |w|$
- *Base case:* For all $i = 1, \ldots, n$ and $A \rightarrow w_i$, compute:

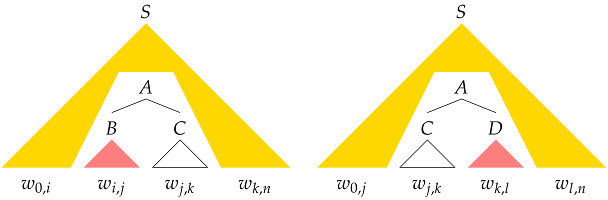
$$\mathrm{P}_G(A \Rightarrow^* w_{i-1,i}) \;=\; p(A \rightarrow w_i)$$

- *Recursion:* For all $k - i = 2, \ldots, n$ and $A \in N$, compute:

$$\mathrm{P}_G(A \Rightarrow^* w_{i,k}) = \sum_{j=i+1}^{k-1} \sum_{A \rightarrow B\,C \in R(A)} p(A \rightarrow B\,C)\mathrm{P}_G(B \Rightarrow^* w_{i,j})\mathrm{P}_G(C \Rightarrow^* w_{j,k})$$

Recursion in $P_G(S \Rightarrow^* w_{0,i}\, A\, w_{k,n})$

$$
\begin{aligned}
P(S \Rightarrow^* w_{0,j}\, C\, w_{k,n}) = \\
&\sum_{i=0}^{j-1} \sum_{\substack{A,B \in N \\ A \to B\,C \in R}} P(S \Rightarrow^* w_{0,i}\, A\, w_{k,n})\, p(A \to B\,C)\, P(B \Rightarrow^* w_{i,j}) \\
+ &\sum_{l=k+1}^{n} \sum_{\substack{A,D \in N \\ A \to C\,D \in R}} P(S \Rightarrow^* w_{0,j}\, A\, w_{l,n})\, p(A \to C\,D)\, P(D \Rightarrow^* w_{k,l})
\end{aligned}
$$



Calculating "outside probabilities"

Construct a table of "outside probabilities"
$P_G(S \Rightarrow^* w_{0,i}\, A\, w_{k,n})$ for all $0 \leq i < k \leq n$ and $A \in N$

Recursion from *larger to smaller* substrings in w.

Base case: $P(S \Rightarrow^* w_{0,0}\, S\, w_{n,n}) = 1$

Recursion: $P(S \Rightarrow^* w_{0,j}\, C\, w_{k,n}) =$

$$\sum_{i=0}^{j-1} \sum_{\substack{A,B \in N \\ A \to B\,C \in R}} P(S \Rightarrow^* w_{0,i}\, A\, w_{k,n}) p(A \to B\,C) P(B \Rightarrow^* w_{i,j})$$

$$+ \sum_{l=k+1}^{n} \sum_{\substack{A,D \in N \\ A \to C\,D \in R}} P(S \Rightarrow^* w_{0,j}\, A\, w_{l,n}) p(A \to C\,D) P(D \Rightarrow^* w_{k,l})$$

The EM algorithm for PCFGs

Input: a corpus of strings $\boldsymbol{w} = w_1, \ldots, w_n$

Guess initial production probabilities $p^{(0)}$

For $t = 1, 2, \ldots$ do:

1. Calculate *expected frequency* $\sum_{i=1}^{n} \mathrm{E}_{p^{(t-1)}}(f_{A\to\alpha}|w_i)$ of each production:

$$\mathrm{E}_p(f_{A\to\alpha}|w) = \sum_{\psi\in\Psi_G(w)} f_{A\to\alpha}(\psi)\mathrm{P}_p(\psi)$$

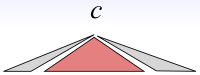
2. Set $p^{(t)}$ to the *relative expected frequency* of each production

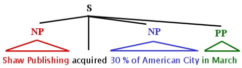
$$p^{(t)}(A\to\alpha) = \frac{\sum_{i=1}^{n} \mathrm{E}_{p^{(t-1)}}(f_{A\to\alpha}|w_i)}{\sum_{A\to\alpha'} \sum_{i=1}^{n} \mathrm{E}_{p^{(t-1)}}(f_{A\to\alpha'}|w_i)}$$

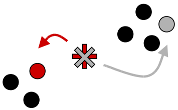
It is as if $p^{(t)}$ were estimated from a visible corpus Ψ_G in which each tree ψ occurs $\sum_{i=1}^{n} \mathrm{P}_{p^{(t-1)}}(\psi|w_i)$ times.

A Nested Distributional Model

- Klein and Manning (2002) propose a model that:
  - Ties spans to linear contexts (like distributional clustering)
  - Considers only proper tree structures
  - Has no symmetries to break (like a dependency model)







Generative model

- S is a sentence
- B is a bracketing
- $P_{bin}(B)$: uniform prob. over binary bracketings
- α_{ij} - parts-of-speech from i to j
- x_{ij} – context of α_{ij}
- $P(S,B) = P_{bin}(B)\, P(S|B)$
- $P(S|B) = \prod P(\alpha_{ij}|B_{ij}) P(x_{ij}|B_{ij})$

Generative model

- S is a sentence
- B is a bracketing
- $P_{bin}(B)$: uniform prob. over binary bracketings
- α_{ij} - parts-of-speech from i to j
- x_{ij} – context of α_{ij}
- $P(S,B) = P_{bin}(B)\, P(S|B)$
- $P(S|B) = \prod P(\alpha_{ij}|B_{ij}) P(x_{ij}|B_{ij})$

Generative model

- S is a sentence
- B is a bracketing
- $P_{bin}(B)$: uniform prob. over binary bracketings
- α_{ij} - parts-of-speech from i to j
- x_{ij} – context of α_{ij}
- $P(S,B) = P_{bin}(B)\ P(S|B)$
- $P(S|B) = \prod_{i<j} P(\alpha_{ij}|B_{ij})P(x_{ij}|B_{ij})$

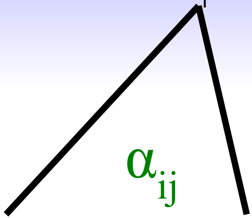
O	Factory	Paysrolls	Fell	In	September	
	T	T			T	O
		T				Factory
			T		T	Payrolls
				T	T	Fell
					T	In
						September

(((Factory) (Payrolls)) (((Fell) ((In) (September)))))

NN NN VBP IN NN

Constituent...

O	Factory	Paysrolls	Fell	In	September	
	T	T			T	O
		T				Factory
			T		T	Payrolls
				T	T	Fell
					T	In
						September

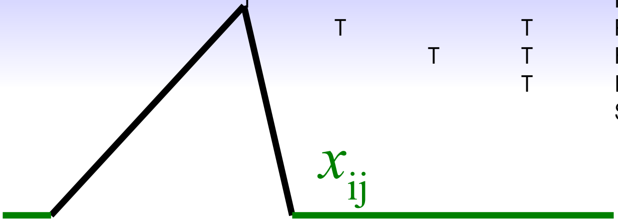


(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

... Context

O	Factory	Paysrolls	Fell	In	September	
	T	T			T	O
		T				Factory
			T		T	Payrolls
				T	T	Fell
					T	In
						September

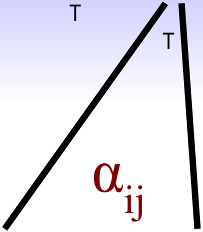


(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

Distituent...

O	Factory	Paysrolls	Fell	In	September	
	T	T			T	O
		T				Factory
			T		T	Payrolls
				T	T	Fell
					T	In
						September

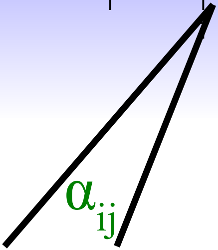


(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

Constituent...

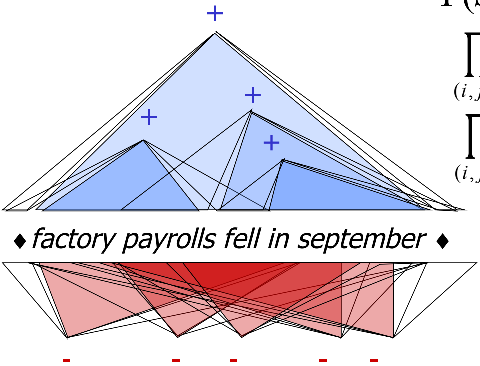
O	Factory	Paysrolls	Fell	In	September	
	T	T			T	O
						Factory
			T		T	Payrolls
				T	T	Fell
					T	In
						September



(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

Constituent-Context Model (CCM)



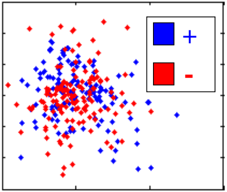
$$P(S|T) = \prod_{(i,j)\in T} P(\phi_{ij}|+)P(\chi_{ij}|+) \prod_{(i,j)\notin T} P(\phi_{ij}|-)P(\chi_{ij}|-)$$

$P(fpfis|+)$

$P(\blacklozenge\_\_\blacklozenge|+)$

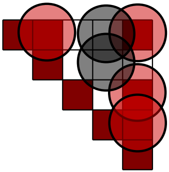
$P(fp|+)$

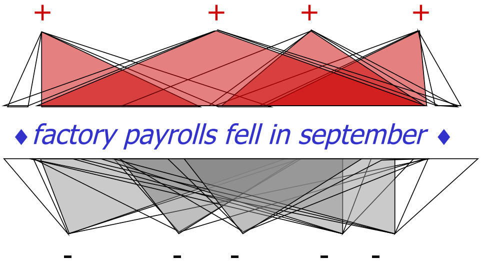
$P(\blacklozenge\_\_fell|+)$



Constituent-Context Model (CCM)

$$P(S,B) = P(B)\,P(S|B)$$





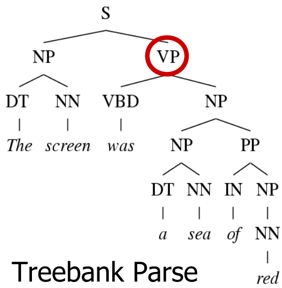
$P(B)$ $P(S|B)$

Constituent-Context Model (CCM; Klein and Manning 2002)

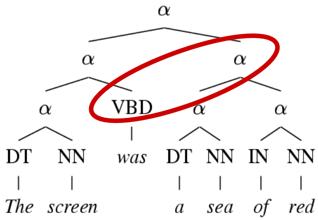
- Generative model where *every* possible constituent/distituent generates its yield as well as its context;
- Parameters of the model are the probabilities with which yields/contexts are generated;
- Parameters are initialized using a clever scheme (Klein, 2005);
- Parameters are optimized using EM & early stopping

Results: Constituency

CCM: 71.9%



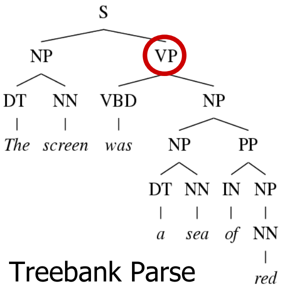
Treebank Parse



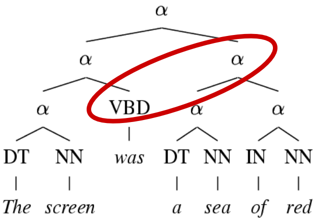
CCM Parse

Results: Constituency

Right-Branch	70.0	



Treebank Parse



CCM Parse

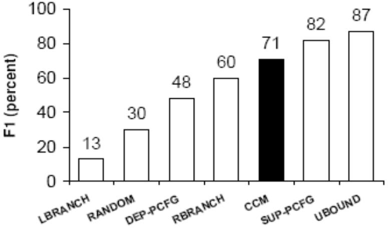
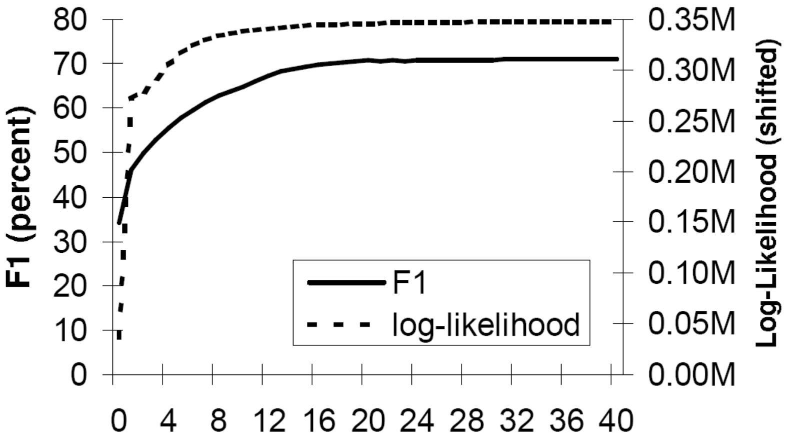


Figure 4: F_1 for various models on WSJ-10.

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

Figure 6: Comparative ATIS parsing results.


F1 (percent)
Log-Likelihood (shifted)
80
70
60
50
40
30
20
10
0
0.35M
0.30M
0.25M
0.20M
0.15M
0.10M
0.05M
0.00M
0 4 8 12 16 20 24 28 32 36 40
F1
log-likelihood

Spectrum of Systematic Errors

CCM analysis better

Treebank analysis better

Analysis	Inside NPs	Possesives	Verb groups
CCM	*the [lazy cat]*	*John ['s cat]*	*[will be] there*
Treebank	*the lazy cat*	*[John 's] cat*	*will [be there]*
CCM Right?	Yes	Maybe	No

But the worst errors are the non-systematic ones (~25%)

How good is CCM?

- How good is CCM's f-score of 71.9% (63.2% with induced POS tags)
- It can be improved to 77.6% if enriched with dependency structure (Klein and Manning 2004)
- Yet, there shortcomings of CCM:
  - Initialization & stopping heuristics play a big role;
  - The generative mode is linguistically not plausible;
  - No **discontiguous** context is taken into account

How good is CCM?

- How good is CCM's f-score of 71.9% (63.2% with induced POS tags)
- It can be improved to 77.6% if enriched with dependency structure (Klein and Manning 2004)
- Yet, there shortcomings of CCM:
  - Initialization & stopping heuristics play a big role;
  - The generative mode is linguistically not plausible;
  - No **discontiguous** context is taken into account

How good is CCM?

- How good is CCM's f-score of 71.9% (63.2% with induced POS tags)
- It can be improved to 77.6% if enriched with dependency structure (Klein and Manning 2004)
- Yet, there are some shortcomings of CCM:
  - Initialization & stopping heuristics play a big role;
  - The generative mode is linguistically not plausible;
  - No **discontiguous** context is taken into account

Maximum likelihood