Natural Language Models and Interfaces
Part B, lecture 3

Ivan Titov

Institute for Logic, Language and Computation
Today

- (Treebank) PCFG weaknesses
- PCFG extensions
  - Structural annotation
  - Lexicalization
- PCFG as a language model
\[ p(T) = 1.0 \times 0.2 \times 1.0 \times 0.4 \times 0.5 \times 0.3 \times \]
\[ 0.5 \times 0.3 \times 0.2 \times 1.0 \times 0.6 \times 0.5 \times 0.3 \times 0.7 \]
\[ = 2.26 \times 10^{-5} \]
CKY / Viterbi Algorithm

John Cocke  Tadao Kasami  Andrew J. Viterbi
For every \( C \) choose \( C_1, C_2 \) and mid such that

\[ P(T_1) \times P(T_2) \times P(C \rightarrow C_1C_2) \]

is maximal, where \( T_1 \) and \( T_2 \) are left and right subtrees.
CKY in action

<table>
<thead>
<tr>
<th>lead</th>
<th>can</th>
<th>poison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Preterminal rules

- $S \rightarrow NP \ VP$
- $VP \rightarrow M \ V$
- $VP \rightarrow V$
- $NP \rightarrow N$
- $NP \rightarrow N \ NP$

Inner rules

- $N \rightarrow can$
- $N \rightarrow lead$
- $N \rightarrow poison$
- $M \rightarrow can$
- $M \rightarrow must$
- $V \rightarrow poison$
- $V \rightarrow lead$
We will discuss these models today.

POS tags are not part of the evaluation even though (normally) predicted by a parser.

The best results reported (as of 2012)
Today

- (Treebank) PCFG weaknesses
- PCFG extensions
  - Structural annotation
  - Lexicalization
- PCFG as a language model
Treebank PCFG

- Directly read-off rules from the treebank:

```
S \rightarrow NP VP 1
NP \rightarrow PN N 1
PN \rightarrow My 1
N \rightarrow Dog 1
VP \rightarrow V NP 1
NP \rightarrow DN 1
D \rightarrow a 1
N \rightarrow sausage 1
```

- The results are not great: around 72% F1

In practice, binarized ones (we discussed this last time)
Weaknesses of (treebank) PCFGs

- They do not encode lexical preferences
- They do not encode structural properties (beyond single rules)
Subject and object NPs are (statistically) very different

- NPs under $S$ vs. NPs under $VP$

Independence assumptions in PCFGs are too strong for this grammar
Subject and object NPs are (statistically) very different

- NPs under S vs. NPs under VP

<table>
<thead>
<tr>
<th>Types of NP</th>
<th>NP PP</th>
<th>D N</th>
<th>PN</th>
</tr>
</thead>
<tbody>
<tr>
<td>All NPs</td>
<td>11%</td>
<td>9%</td>
<td>6%</td>
</tr>
<tr>
<td>NPs under S (subjects)</td>
<td>9%</td>
<td>9%</td>
<td>21%</td>
</tr>
<tr>
<td>NPs under VP (objects)</td>
<td>23%</td>
<td>7%</td>
<td>4%</td>
</tr>
</tbody>
</table>

Many more pronouns as subjects; much less frequently prepositional phrases as subjects.
Subject and object NPs are (statistically) very different

- NPs under S vs. NPs under VP

How can we modify the grammar?
Context-free constraint

- Subject and object NPs are (statistically) very different
  - NPs under S vs. NPs under VP

Structural annotation, specifically grandparent annotation \[\text{[Johnson 98]}\]

\[
\begin{align*}
S & \quad \text{NP}^\text{S} \\
\text{PN} & \quad \text{NP} & \quad \text{VP} \\
\text{N} & \quad \text{V} & \quad \text{NP}^\text{VP} \\
\text{My dog} & \quad \text{ate} & \quad \text{a sausage}
\end{align*}
\]
Context-free constraint

- Subject and object NPs are (statistically) very different
  - NPs under S vs. NPs under VP

Recall: instead of transforming the grammar we can see this in terms of transforming trees (on preprocessing) and the inducing a PCFG from the transformed treebank
Approaches to enriching a grammar

- Structural annotation \([\text{Johnson 98, Klein and Manning 03}]\)
- Lexicalization \([\text{Collins 99, Charniak 00}]\)

- There was a period in natural language processing when many researchers abandoned PCFGs and focused on richer modeling of context (history-based models) instead

- … but more recent research showed that high accuracy can be achieved with PCFGs if an appropriate grammar is chosen

Also known as grammar transforms
Rule applications depend on past ancestors in the tree (not only parents)  

(Vertical) order 1

Vertical order 2
PCFG weakness: Close Attachment

- Compare 2 configurations from the 1st lecture:

- Close attachment is a-priori more likely (at least in Penn Treebank)

- Here they are semantically equivalent (as the box on the table implies that box is also on the table)

  - … but consider "CEO of Yahoo in 2012"
PCFG weakness: Close Attachment

Can PCFG give a preference to one or another structure?

No, the same rules are used in both constructions, so a PCFG is guaranteed to return the same scores!

Would vertical Markovization help here (encode preference for close attachment)?
PCFG weakness: Close Attachment

From the treebank, the enriched PCFG will assign higher probability to the rule

\[ \text{NP}^{\text{PP}} \rightarrow \text{NP}^{\text{NP}} \quad \text{PP}^{\text{NP}} \]

than to the rule

\[ \text{NP}^{\text{NP}} \rightarrow \text{NP}^{\text{NP}} \quad \text{PP}^{\text{NP}} \]

Consequently higher accuracy (in average) is expected.
Vertical Markovization

[In this part of the slides some illustrations are adapted from Dan Klein]
Recall binarization (transformation to CNF form)

In vertical Markovization we increased context, in horizontal Markovization we want to reduce it.

Can be regarded as horizontal history.
Recall binarization (transformation to CNF form)

Horizontal order $h = \infty$

Horizontal order $h = 1$

![Diagram showing the transformation of the sentence into CNF form with different horizontal orders.](image)

![Bar chart showing the comparison of different horizontal Markov orders.](chart1)

![Bar chart showing the comparison of different horizontal Markov orders.](chart2)
Can we do both?

The German carrier operates from Berlin.
Can we do both?

The German carrier operates from Berlin.

Vertical order \( v = 2 \)
Can we do both?

Vertical order $v = 2$

Horizontal order $h = \infty$
Can we do both?

This is what you *minimally* need for the stage 2.2

Vertical order $v = 2$

Horizontal order $h = 1$
Around 78%, compare with 72% for the original treebank PCFG

Any idea how maybe improve this using techniques we discussed?
PoS tags in Penn Treebank are too coarse

Very obvious for IN tag:

- Assigned both to 'normal' prepositions (to form a prepositional phrase) – in, on, at, … –
- and to subordinating conjunctions (e.g., if)
  - E.g., check if advertising works

This change alone leads to a 2% boost in performance:

- from 78.2 to 80.3

[Klein and Manning 2003]
Splitting: other symbols

- **Split determiners**: on demonstrative ("those") and other (e.g., "the", "a")
- **Split adverbials**: on phrasal and not ("quickly" vs. "very")
- ...

All these changes (and a couple of other ones) lead to 86.3 % F1, a very respectable (and maybe even surprising) performance for an unlexicalized PCFG model.

[Klein and Manning 2003]
Preview: F1 bracket score

The best results reported (as of 2012)
Alternative ideas

- Learning types of nonterminals from data, i.e. automatically enriching the grammar (Latent-annotated PCFGs, LA-PCFG)
  
  - One can think of this as a type of clustering of tree contexts of non-terminal symbols

```
S
  /\  |
NP-156 VP-112
  /\  /\  |
PN-1 N-652 V-134 NP-5
  |  |  |  |
My dog ate a sausage
```

[32%

[Matsuzaki et al., 2005,
Petrov et al., 2006] Around 89.6% F1
Alternative ideas

- Learning types of nonterminals as an array of properties
  - + a smoothing model which uses these arrays

This is a simplification as the actual model has been tested in the context of CFGs but using a left-corner automata

Both this model (and to a lesser degree LA-PCFGs) may not be regarded as entirely unlexicalized

In the same ballpark (around 89.0 – 90.0 depending on the setting)

Alternative ideas

- DOP ("Data-oriented parsing"): larger fragments of trees are taken into account

- Roughly a grammar is defined in terms of larger tree fragments
  - Decomposition of a tree into fragments can be induced using statistical modeling techniques


Again achieves very competitive performance but can be slow
Approaches to enriching a grammar

- Structural annotation  [Johnson 98, Klein and Manning 03]
- Lexicalization  [Collins 99, Charniak 00]
How do we decide that the two sentences below have different analysis?

Clearly, this is hard to image that this can be resolved without modeling lexical dependencies in some way.
Lexicalization: motivation

- Another example (coordination ambiguity):

How different are scores by a treebank PCFG?

- What kind of lexical dependencies one would want to capture?
Lexicalized models

- Lexicalized models are effectively models of dependency structure
- Remember we discussed how (potentially) dependencies can be extracted from syntactic trees:

```
  S
 / \  
NP  VP
 /     
PN  V   NP
 |     |
 My  dog ate D  N
   |     |
    |     a  sandwich
```

One potential rule to extract nsubj dependency

- Now we will discuss:
  - More specifically how these head extraction rules look like
  - How dependency information can be integrated into the grammar
Dependency structures (Recap)

A dependency: My is a dependent, dog is a head

Subtrees (roughly) correspond to phrases in the constituent representation. For each phrase we have its syntactic head

Dependency types (labels on edges) are not used in this context
Head recovery rules

- For every rule in a grammar one can define where the head for the parent nonterminal should be taken from:

  \[
  S \rightarrow VP \ NP \\
  NP \rightarrow PN \ N \\
  NP \rightarrow D \ N \\
  VP \rightarrow V \ NP
  \]

  These rules are not specified for each production but defined in a procedural fashion:

  for NP rules, if the right side contains N, choose the rightmost N as a head; otherwise, …

- Rules are applied recursively, bottom up:

  Each binary rule corresponds to an arc
What is tricky with lexicalization?

- Grammar is very large:
  - How do we parse with such a grammar?
  - Rules are very 'detailed', how do we estimate probabilities of these rules?
Potentially rules can be binarized in a much the same way as for unlexicalized PCFG, we will arrive at a grammar with the rules:

\[
\begin{align*}
C(h) & \rightarrow C_1(h) C_2(w) \\
C(h) & \rightarrow C_1(w) C_2(h) \\
C(h) & \rightarrow h
\end{align*}
\]

How large is such a grammar?

For now we can assume that all combinations of nonterminals and terminals are possible; this will be the case because of aggressive smoothing.

\[O(\left| V \right|^3 \left| \Sigma \right|^2)\] huge!
Recall: CKY time complexity was $O(|G|n^3)$

Grammar size for the lexicalized grammar is $O(|V|^3|\Sigma|^2)$

Naïve application of CKY: $O(|V|^3|\Sigma|^2n^3)$

Any idea why it can done much faster? (though not too fast anyway)

- Not all the rules are relevant: the number of rules potentially applicable to a sentence is $O(|V|^3n^2)$
- It yields parsing in $O(|V|^3n^5)$

Not terribly fast but manageable (especially with pruning techniques we discussed)
Estimation with lexicalized PCFG

- **Problem:** we need to estimate probabilities of the rules like
  \[ VP(saw) \rightarrow VBD(saw) \ NP(her) \ NP(today) \]

- We cannot hope to get good estimates even using horizontal Markovization techniques we discussed

- **Solution:** rules are broken down to even smaller steps, roughly:

  First generate unlexicialized children (except for the head) using a Markov process (not in one step!)

  Then fill in children – one by one (not in one step!)

[Collins 99]
F1 bracket score

- Treebank PCFG
- Unlexicalized PCFG (Klein and Manning, 2003)
- Lexicalized PCFG (Collins, 1999)
- Automatically Induced PCFG (Petrov et al., 2006)
- The best results reported (as of 2012)
What kind of mistakes the parser makes?

- Though the Collins parser is generally fairly accurate (around 90% on NP-chunking), it still makes a lot of mistakes in PP-attachment:

  - Around 80%

- and coordination ambiguity:

  - Around 50-60%

Maybe the dataset is too small to capture corresponding lexical correlations?
Today

- (Treebank) PCFG weaknesses
- PCFG extensions
  - Structural annotation
  - Lexicalization
- PCFG as a language model
Language modeling

- The goal of language modeling is to **estimate how likely is a sequence of words**
  - You considered ngram language models in the Part A
  - They are an important component in, for example, in machine translation and speech recognition

- **Estimate the probability of a sequence of a word sequence** $p(w_1, \ldots, w_n)$
  - or (essentially equivalently) estimate the probability of the next word $p(w_n|w_1, \ldots, w_{n-1})$
Language modeling

- **Ngram models are not very good at capturing long dependencies**
  
  - Availability of Web-scale data means that they are not as bad as long (up to 5 words) ngrams estimated from over a trillion of tokens is available for English.
  
  - Still it can easily lead to ungrammatical sentence
    
    - *John has not even reached his/her destination*
    
    - *Er kam am Freitagabend nach einem harten Arbeitstag und dem üblichen Ärger, der ihn schon seit Jahren immer wieder an seinem Arbeitsplatz plagt, mit fraglicher Freude auf ein Mahl, das seine Frau ihm, wie er hoffte, bereits aufgetischt hatte, endlich zu Hause an.*
      
      [Wikipedia]

- Syntactic information should help us in modeling these long dependencies
Syntactic language models

- To get the probability $P(x)$ of a word sequence $x$ we need to sum probabilities $P(T)$ of all its syntactic derivations $T \in G(x)$

$$P(x) = \sum_{T \in G(x)} P(T)$$

$$P(\text{lead can poison}) = P(\text{lead} \rightarrow \text{can} \rightarrow \text{poison}) + P(\text{lead} \rightarrow \text{can} \rightarrow \text{poison})$$
Syntactic language models

- To get the probability $P(x)$ of a word sequence $x$ we need to sum probabilities $P(T)$ of all its syntactic derivations $T \in G(x)$

$$P(x) = \sum_{T \in G(x)} P(T)$$

How can we do this efficiently (for PCFGs)?

- (Recall we know how to parse efficiently:
  $$\arg \max_{T \in G(x)} P(T)$$)
Modifying CKY

- In CKY we maintained the probability of the most probable subtree for each signature.
- Here, we will maintain the sum of probabilities of all the subtrees for each signature.

- The CKY algorithm is an instance of the Viterbi algorithm, also known as Max-Product.
- We will look into the algorithm called Sum-Product.

Have you considered HMM as a language model in Part A?
for each max from 2 to n

for each min from max - 2 down to 0

for each syntactic category C

double best = undefined

for each binary rule C \rightarrow C_1 C_2

for each mid from min + 1 to max - 1

double t_1 = chart[min][mid][C_1]

double t_2 = chart[mid][max][C_2]

double candidate = t_1 * t_2 * p(C \rightarrow C_1 C_2)

if candidate > best then

best = candidate

chart[min][max][C] = best
The probability of the sentence can be read from `chart[0][n][s]`
Material for the Exam

- CFG and PCFGs
  - incl. tree probability; estimation from a treebank
- CNF form, binarization
- CKY algorithm for CFG and PCFG
  - Incl. understanding the data structure (the chart), understanding the unary closures both for the probabilistic case and not, time complexity of the algorithm
- Ideas about weaknesses of PCFGs
- Structural annotation: vertical and horizontal Markovization
- Lexicalization: how to get rules, basic understanding of estimation
- Computing the probability of a word sequence with PCFGs