Lecture 17: More on PCFG parsing

Probabilistic Context-Free Grammars

For every nonterminal \( X \), define a probability distribution \( P(X \rightarrow \alpha | X) \) over all rules with the same LHS symbol \( X \):

\[
\begin{align*}
S &\rightarrow NP \ VP & 0.8 \\
S &\rightarrow S \ conj \ S & 0.2 \\
NP &\rightarrow Noun & 0.2 \\
NP &\rightarrow Det \ Noun & 0.4 \\
NP &\rightarrow NP \ PP & 0.2 \\
NP &\rightarrow NP \ conj \ NP & 0.2 \\
VP &\rightarrow Verb & 0.4 \\
VP &\rightarrow Verb \ NP & 0.3 \\
VP &\rightarrow Verb \ NP \ NP & 0.1 \\
VP &\rightarrow VP \ PP & 0.2 \\
PP &\rightarrow P \ NP & 1.0
\end{align*}
\]

Transforming a PCFG to Chomsky Normal Form

This grammar is not in Chomsky Normal Form:

- Ternary rules, e.g:
  \( S \rightarrow S \ conj \ S \)
- RHS with nonterminals and terminals
  \( S \rightarrow S \ conj \ S \)
- Unary rules from one nonterminal to another, e.g.:
  \( VP \rightarrow Verb \)

Transforming a PCFG to Chomsky Normal Form

\[
\begin{align*}
S &\rightarrow NP \ VP & 0.8 \\
S &\rightarrow S \ conj \ S & 0.2 \\
\text{conjS} &\rightarrow Conj \ S & 1.0 \\
NP &\rightarrow N & 0.2 \\
NP &\rightarrow Det \ Noun & 0.4 \\
NP &\rightarrow NP \ PP & 0.2 \\
NP &\rightarrow NP \ conj \ NP & 1.0 \\
\text{conjNP} &\rightarrow Conj \ NP & 1.0 \\
VP &\rightarrow Verb & 0.3 \\
VP &\rightarrow Verb \ NP & 0.3 \\
VP &\rightarrow Verb \ NP \ NP & 0.1 \\
VP &\rightarrow VP \ PP & 0.3 \\
PP &\rightarrow Prep \ NP & 1.0 \\
\text{Prep} &\rightarrow P & 1.0 \\
\text{Noun} &\rightarrow N & 1.0 \\
\text{Verb} &\rightarrow V & 1.0
\end{align*}
\]
Transforming a PCFG to Chomsky Normal Form

What we did here:
- New NTs for ternary rules
  \[ S \rightarrow S \text{ conj } S \quad 0.2 \]
  \[ \text{conjS} \rightarrow \text{Conj } S \quad 1.0 \]
- Introduced new terminals
  \[ \text{Conj} \rightarrow C \quad 1.0 \]
  \[ \text{Det} \rightarrow D \quad 1.0 \]
- Removed unary rules from one NT to another, e.g.:
  \[ VP \rightarrow \text{Verb} \quad 0.3 \]
  \[ \text{is now} \]
  \[ VP \rightarrow \text{V} \quad 0.3 \]

PCFG parsing (decoding): Probabilistic CKY

Probabilistic CKY: Viterbi

Like standard CKY, but with probabilities.
Finding the most likely tree is similar to Viterbi for HMMs:

Initializiation:
- \( \text{optional} \) Every chart entry that corresponds to a terminal
  (entries w in cell[i][i]) has a Viterbi probability \( P_{\text{VIT}}(w[i][i]) = 1 \)
- Every entry for a non-terminal \( X \) in cell[i][j] has Viterbi probability \( P_{\text{VIT}}(X[i][j]) = P(X \rightarrow w | X) \) [and a single backpointer to w[i][i] (*)]

Recurrence: For every entry that corresponds to a non-terminal \( X \)
  in cell[i][j], keep only the highest-scoring pair of backpointers
to any pair of children \( (Y \text{ in cell}[i][k] \text{ and } Z \text{ in cell}[k+1][j]) \):
  \[ P_{\text{VIT}}(X[i][j]) = \text{argmax}_{Y,Z,k} P_{\text{VIT}}(Y[i][k]) \times P_{\text{VIT}}(Z[k+1][j]) \times P(X \rightarrow Y \text{ Z} | X) \]

Final step: Return the Viterbi parse for the start symbol \( S \)
in the top cell[1][n].

*This is unnecessary for simple PCFGs, but can be helpful for more complex probability models
How well can a PCFG model the distribution of trees?

PCFGs make independence assumptions:
Only the label of a node determines what children it has.

Factors that influence these assumptions:
Shape of the trees: A corpus with flat trees (i.e. few nodes/sentence) results in a model with few independence assumptions.

Labeling of the trees: A corpus with many node labels (nonterminals) results in a model with few independence assumptions.

Example 1: flat trees

```
S
  I eat sushi with tuna
```

What sentences would a PCFG estimated from this corpus generate?

Example 2: deep trees, few labels

```
I S S S S S
  eat sushi with chopsticks
```

What sentences would a PCFG estimated from this corpus generate?

Example 3: deep trees, many labels

```
I S S1 S2 S
  eat sushi with S3 tuna
```
```
I S S1 S2 S3
  eat sushi with chopsticks
```

What sentences would a PCFG estimated from this corpus generate?
Aside: Bias/Variance tradeoff

A probability model has low bias if it makes few independence assumptions.
⇒ It can capture the structures in the training data.

This typically leads to a more fine-grained partitioning of the training data.

Hence, fewer data points are available to estimate the model parameters.

This increases the variance of the model.
⇒ This yields a poor estimate of the distribution.

Precision and recall

Precision and recall were originally developed as evaluation metrics for information retrieval:
- **Precision**: What percentage of retrieved documents are relevant to the query?
- **Recall**: What percentage of relevant documents were retrieved?

In NLP, they are often used in addition to accuracy:
- **Precision**: What percentage of items that were assigned label X do actually have label X in the test data?
- **Recall**: What percentage of items that have label X in the test data were assigned label X by the system?

Particularly useful when there are more than two labels.

True vs. false positives, false negatives

Items labeled X in the gold standard (‘truth’)
= TP + FN

Items labeled X by the system
= TP + FP

- **True positives**: Items that were labeled X by the system, and should be labeled X.
- **False positives**: Items that were labeled X by the system, but should not be labeled X.
- **False negatives**: Items that were not labeled X by the system, but should be labeled X.
Precision, recall, f-measure

Precision: \( P = \frac{TP}{TP + FP} \)
Recall: \( R = \frac{TP}{TP + FN} \)
F-measure: harmonic mean of precision and recall
\[
F = \frac{2 \cdot P \cdot R}{P + R}
\]

Items labeled X in the gold standard (‘truth’)
= TP + FN
Items labeled X by the system
= TP + FP

Evalb (“Parseval”)

Measures recovery of phrase-structure trees.
Labeled: span and label (NP, PP,...) has to be right.
[Earlier variant— unlabeled: span of nodes has to be right]

Two aspects of evaluation
Precision: How many of the predicted nodes are correct?
Recall: How many of the correct nodes were predicted?
Usually combined into one metric (F-measure):
\[
P = \frac{\text{#correctly predicted nodes}}{\text{#predicted nodes}}
\]
\[
R = \frac{\text{#correctly predicted nodes}}{\text{#correct nodes}}
\]
\[
F = \frac{2PR}{P + R}
\]

Parseval in practice

Gold standard
Parser output

Eat sushi with tuna: Precision: 4/5 Recall: 4/5
Eat sushi with chopsticks: Precision: 4/5 Recall: 4/5

Penn Treebank parsing

CS447: Natural Language Processing (J. Hockenmaier)
The Penn Treebank

The first publicly available syntactically annotated corpus
- Wall Street Journal (50,000 sentences, 1 million words)
- also Switchboard, Brown corpus, ATIS

The annotation:
- POS-tagged (Ratnaparkhi’s MXPOST)
- Manually annotated with phrase-structure trees
- Richer than standard CFG: Traces and other null elements used to represent non-local dependencies (designed to allow extraction of predicate-argument structure) [more on this later in the semester]

Standard data set for English parsers

The Treebank label set

48 preterminals (tags):
- 36 POS tags, 12 other symbols (punctuation etc.)
- Simplified version of Brown tagset (87 tags)
  (cf. Lancaster-Oslo/Bergen (LOB) tag set: 126 tags)

14 nonterminals:
- standard inventory (S, NP, VP,...)

A simple example

Relatively flat structures:
- There is no noun level
- VP arguments and adjuncts appear at the same level

Function tags, e.g. -SBJ (subject), -MNR (manner)

A more realistic (partial) example

Until Congress acts, the government hasn't any authority to issue new debt obligations of any kind, the Treasury said .... .
The Penn Treebank CFG

The Penn Treebank uses very flat rules, e.g.:

NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB RT JJ JJ NNS
NP → RB JJ JJ NNP NNS
NP → RB JJ NNP NNP NNP JJ JJ NN
NP → PT JJ NN CC JJ JJ JJ NN NNS
NP → RB DT JJ NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBZ NN NNP NNP FW NNP
NP → NP JJ , JJ ',' SBAR ',' NNS

– Many of these rules appear only once.
– Many of these rules are very similar.
– Can we pool these counts?

PCFGs in practice:
Charniak (1996) Tree-bank grammars

How well do PCFGs work on the Penn Treebank?

– Split Treebank into test set (30K words) and training set (300K words).
– Estimate a PCFG from training set.
– Parse test set (with correct POS tags).
– Evaluate unlabeled precision and recall

<table>
<thead>
<tr>
<th>Sentence Length</th>
<th>Average Length</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:12</td>
<td>8.7</td>
<td>88.6</td>
<td>91.7</td>
</tr>
<tr>
<td>2:16</td>
<td>11.4</td>
<td>85.0</td>
<td>87.7</td>
</tr>
<tr>
<td>2:20</td>
<td>13.5</td>
<td>83.5</td>
<td>86.2</td>
</tr>
<tr>
<td>2:25</td>
<td>16.3</td>
<td>82.0</td>
<td>84.0</td>
</tr>
<tr>
<td>2:30</td>
<td>18.7</td>
<td>80.6</td>
<td>82.5</td>
</tr>
<tr>
<td>2:40</td>
<td>21.9</td>
<td>78.8</td>
<td>80.4</td>
</tr>
</tbody>
</table>

Two ways to improve performance

… change the (internal) grammar:
- Parent annotation/state splits:
  Not all NPs/VPs/DTs/… are the same.
  It matters where they are in the tree

… change the probability model:
- Lexicalization:
  Words matter!
- Markovization:
  Generalizing the rules

The parent transformation

PCFGs assume the expansion of any nonterminal is independent of its parent.

But this is not true: NP subjects more likely to be modified than objects.

We can change the grammar by adding the name of the parent node to each nonterminal

(a) VP
   V NP
   NP PP
   Det N P NP

(b) VP'S
   V NP VP
   NP'NP PP NP
   Det N P NP PP
Markov PCFGs (Collins parser)

The RHS of each CFG rule consists of:
- one head $H_X$
- $n$ left sisters $L_1 \ldots L_n$
- $m$ right sisters $R_1 \ldots R_m$

Replace rule probabilities with a generative process:
For each nonterminal $X$:
- generate its head $H_X$ (nonterminal or terminal)
- then generate its left sisters $L_1 \ldots L_n$ and a STOP symbol conditioned on $H_X$
- then generate its right sisters $R_1 \ldots R_m$ and a STOP symbol conditioned on $H_X$

Lexicalization

PCFGs can’t distinguish between “eat sushi with chopsticks” and “eat sushi with tuna”.

We need to take words into account!

$$P(\text{VP} \xrightarrow{\text{eat}} \text{VP PP with chopsticks} | \text{VP eat})$$

vs.

$$P(\text{VP} \xrightarrow{\text{eat}} \text{VP PP with tuna} | \text{VP eat})$$

Problem: sparse data (PP with fattywhitel... tuna....)

Solution: only take head words into account!

Assumption: each constituent has one head word.

Lexicalized PCFGs

At the root (start symbol $S$), generate the head word of the sentence, $w_s$, with $P(w_s)$

Lexicalized rule probabilities:
Every nonterminal is lexicalized: $X_{w_s}$
Condition rules $X_{w_s} \rightarrow \alpha Y \beta$ on the lexicalized LHS $X_{w_s}$.

$$P( X_{w_s} \rightarrow \alpha Y \beta | X_{w_s})$$

Word-word dependencies:
For each nonterminal $Y$ in RHS of a rule $X_{w_s} \rightarrow \alpha Y \beta$,
condition $w_Y$ (the head word of $Y$) on $X$ and $w_s$:

$$P( w_Y | Y, X, w_s)$$

Dealing with unknown words

A lexicalized PCFG assigns zero probability to any word that does not appear in the training data.

Solution:

Training: Replace rare words in training data with a token ‘UNKNOWN’.

Testing: Replace unseen words with ‘UNKNOWN’
Refining the set of categories

Unlexicalized Parsing (Klein & Manning '03)
Unlexicalized PCFGs with various transformations of the training data and the model, e.g.:
- Parent annotation (of terminals and nonterminals): distinguish preposition IN from subordinating conjunction IN etc.
- Add head tag to nonterminals (e.g. distinguish finite from infinite VPs)
- Add distance features
Accuracy: 86.3 Precision and 85.1 Recall

The Berkeley parser (Petrov et al. '06, '07)
Automatically learns refinements of the nonterminals
Accuracy: 90.2 Precision, 89.9 Recall

Summary
The Penn Treebank has a large number of very flat rules.
Accurate parsing requires modifications to the basic PCFG model: refining the nonterminals, relaxing the independence assumptions by including grandparent information, modeling word-word dependencies, etc.

How much of this transfers to other treebanks or languages?