Speech and Language Processing: Statistical Parsing

Chapter 14

Statistical Parsing

- Statistical parsing uses a probabilistic model of syntax in order to assign probabilities to each parse tree.
- Provides principled approach to resolving syntactic ambiguity.
- Allows supervised learning of parsers from tree-banks of parse trees provided by human linguists.
Probabilistic Context Free Grammar (PCFG)

- A PCFG is a probabilistic version of a CFG where each production has a probability.
- Probabilities of all productions rewriting a given non-terminal must add to 1, defining a distribution for each non-terminal.
- String generation is now probabilistic where production probabilities are used to non-deterministically select a production for rewriting a given non-terminal.

Simple PCFG for ATIS English

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Prob</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.8</td>
<td>Det → the</td>
</tr>
<tr>
<td>S → Aux NP VP</td>
<td>0.1</td>
<td>Noun → book</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
<td>Verb → book</td>
</tr>
<tr>
<td>NP → Pronoun</td>
<td>0.2</td>
<td>Pronoun → I</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>0.2</td>
<td>Proper-Noun → Houston</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>0.3</td>
<td>Aux → does</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>VP → Verb</td>
<td>0.2</td>
<td>Prep → from</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>
Sentence Probability

- Assume productions for each node are chosen independently.
- Probability of derivation is the product of the probabilities of its productions.

\[ P(D_1) = 0.1 \times 0.5 \times 0.5 \times 0.6 \times 0.6 \times \\ 0.5 \times 0.3 \times 1.0 \times 0.2 \times 0.2 \times \\ 0.5 \times 0.8 \]
\[ = 0.0000216 \]

Other Parses?

- book the flight through Houston
Syntactic Disambiguation

- Resolve ambiguity by picking most probable parse tree.

\[ P(D_2) = \]

\[
\begin{align*}
S & \quad 0.1 \\
VP & \quad 0.3 \\
NP & \quad 0.5 \\
Verb & \quad 0.6 \\
book & \\
Det & \quad 0.3 \\
Nominal & \\
Prep & \quad 0.2 \\
through & \\
the & \quad 0.5 \\
Noun & \\
flight & \\
PP & \quad 1.0 \\
NP & \quad 0.2 \\
Proper-Noun & \\
Houston & \quad 0.8
\end{align*}
\]

\[ = 0.00001296 \]
Syntactic Disambiguation

• Resolve ambiguity by picking most probable parse tree.

\[
P(D_2) = 0.1 \times 0.3 \times 0.5 \times 0.6 \times 0.5 \times 0.6 \times 0.3 \times 1.0 \times 0.5 \times 0.2 \times 0.2 \times 0.8 = 0.00001296
\]

Disambiguation Result?
Sentence Probability

• Probability of a sentence is the sum of the probabilities of all of its derivations.

\[ P(\text{"book the flight through Houston"}) = P(D_1) + P(D_2) = 0.0000216 + 0.00001296 = 0.00003456 \]

Three Useful PCFG Tasks

• Observation likelihood: To classify and order sentences.

• Most likely derivation: To determine the most likely parse tree for a sentence.

• Maximum likelihood training: To train a PCFG to fit empirical training data.
PCFG: Most Likely Derivation

• There is an analog to the Viterbi algorithm to efficiently determine the most probable derivation (parse tree) for a sentence.

John liked the dog in the pen.

PCFG: Most Likely Derivation

• There is an analog to the Viterbi algorithm to efficiently determine the most probable derivation (parse tree) for a sentence.

John liked the dog in the pen.
Probabilistic CKY

• CKY can be modified for PCFG parsing by including in each cell a probability for each non-terminal.
• Cell\[i,j\] must retain the *most probable* derivation of each constituent (non-terminal) covering words \(i + 1\) through \(j\) together with its associated probability.
• When transforming the grammar to CNF, must set production probabilities to preserve the probability of derivations.

### Probabilistic Grammar Conversion

<table>
<thead>
<tr>
<th>Original Grammar</th>
<th>Chomsky Normal Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S \rightarrow NP\ VP)</td>
<td>(S \rightarrow NP\ VP)</td>
</tr>
<tr>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>(S \rightarrow \text{Aux NP VP})</td>
<td>(S \rightarrow X1\ VP)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>(S \rightarrow VP)</td>
<td>(X1 \rightarrow \text{Aux NP})</td>
</tr>
<tr>
<td>0.1</td>
<td>1.0</td>
</tr>
<tr>
<td>(S \rightarrow NP)</td>
<td>(S \rightarrow \text{book} \mid \text{include} \mid \text{prefer})</td>
</tr>
<tr>
<td>NP → Pronoun</td>
<td>0.01 0.004 0.006</td>
</tr>
<tr>
<td>0.2</td>
<td>(S \rightarrow \text{Verb NP})</td>
</tr>
<tr>
<td>0.05</td>
<td>(S \rightarrow \text{VP PP})</td>
</tr>
<tr>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>(\text{NP} \rightarrow \text{I} \mid \text{he} \mid \text{she} \mid \text{me})</td>
</tr>
<tr>
<td>0.2</td>
<td>0.1 0.02 0.02 0.06</td>
</tr>
<tr>
<td>0.16</td>
<td>(\text{NP} \rightarrow \text{Houston} \mid \text{NWA})</td>
</tr>
<tr>
<td>0.04</td>
<td>0.16 .04</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>(NP \rightarrow \text{Det Nominal})</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>(\text{Nominal} \rightarrow \text{book} \mid \text{flight} \mid \text{meal} \mid \text{money})</td>
</tr>
<tr>
<td>0.3</td>
<td>0.03 0.15 0.06 0.06</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>(\text{Nominal} \rightarrow \text{Nominal Noun})</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>(\text{Nominal} \rightarrow \text{Nominal PP})</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → Verb</td>
<td>(\text{VP} \rightarrow \text{book} \mid \text{include} \mid \text{prefer})</td>
</tr>
<tr>
<td>0.2</td>
<td>0.1 0.04 0.06</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>(\text{VP} \rightarrow \text{Verb NP})</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>(\text{VP} \rightarrow \text{VP PP})</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>(\text{PP} \rightarrow \text{Prep NP})</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Book       the        flight    through  Houston

S : .01, VP : .1, Verb : .5
Nominal : .03
Noun : .1

None

Det : .6

Nominal : .15
Noun : .5

NP : .6 * .6 * .15 = .054

VP : .5 * .5 * .054 = .0135
Probabilistic CKY Parser

```
<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S : .01, VP: .1, Verb: .5</td>
<td>Nominal: .03</td>
<td>Noun: .1</td>
<td>None</td>
</tr>
<tr>
<td>S : .05 * .5 * .054</td>
<td>NP: .6 * .6 * .15</td>
<td>.054</td>
<td>None</td>
</tr>
<tr>
<td>Det: .6</td>
<td>Nominal: .15</td>
<td>Noun: .5</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
```

Probabilistic CKY Parser

```
<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>S : .01, VP: .1, Verb: .5</td>
<td>Nominal: .03</td>
<td>Noun: .1</td>
<td>None</td>
</tr>
<tr>
<td>S : .05 * .5 * .054</td>
<td>NP: .6 * .6 * .15</td>
<td>.054</td>
<td>None</td>
</tr>
<tr>
<td>Det: .6</td>
<td>Nominal: .15</td>
<td>Noun: .5</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
```

19

20
Probabilistic CKY Parser

Book       the        flight    through  Houston

| S :.01, VP :.1, Verb :.5, Nominal :.03, Noun :.1 | None | None |
| NP :.6 * .6 * .15 = .054 | None |
| NP :.16, PropNoun :.8 | None |
| Prep : .2 | PP : 1.0 * .2 * .16 = .032 |
| Nominal : .5 * .15 * .032 = .0024 |

Probabilistic CKY Parser

Book       the        flight    through  Houston

| S :.01, VP :.1, Verb :.5, Nominal :.03, Noun :.1 | None | None |
| NP :.6 * .6 * .15 = .054 | None |
| NP :.16, PropNoun :.8 | None |
| Prep : .2 | PP : 1.0 * .2 * .16 = .032 |
| Nominal : .5 * .15 * .032 = .0024 |
Probabilistic CKY Parser

Book       the        flight    through Houston

S : .01, VP : .1,
Verb : .5,
Nominal : .03
Noun : .1

Det : .6
Nominal : .15
Noun : .5
None

NP : .6 * .6 * .15 = .054

VP : .5 * .5 * .054 = .0135

S : .05 * .5 * .054 = .0000216

Prep : .2
PP : 1.0 * .2 * .16 = .032

NP : .16
PropNoun : .8
Probabilistic CKY Parser

Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.
PCFG: Supervised Training

- If parse trees are provided for training sentences, a grammar and its parameters can be estimated directly from counts accumulated from the tree-bank (with appropriate smoothing).

```
<table>
<thead>
<tr>
<th>Production</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>0.9</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
</tr>
<tr>
<td>NP → Det A N</td>
<td>0.5</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>0.3</td>
</tr>
<tr>
<td>NP → PropN</td>
<td>0.2</td>
</tr>
<tr>
<td>A → ε</td>
<td>0.6</td>
</tr>
<tr>
<td>A → Adj A</td>
<td>0.4</td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP → V NP</td>
<td>0.7</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3</td>
</tr>
</tbody>
</table>
```

English

Estimating Production Probabilities

- Set of production rules can be taken directly from the set of rewrites in the treebank.
- Parameters can be directly estimated from frequency counts in the treebank.

\[
P(\alpha \rightarrow \beta | \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}
\]
Vanilla PCFG Limitations

• Since probabilities of productions do not rely on specific words or concepts, only general structural disambiguation is possible (e.g. prefer to attach PPs to Nominals).
• Consequently, vanilla PCFGs cannot resolve syntactic ambiguities that require semantics to resolve, e.g. ate with fork vs. meatballs.
• In order to work well, PCFGs must be lexicalized, i.e. productions must be specialized to specific words by including their head-word in their LHS non-terminals (e.g. VP-ate).

Example of Importance of Lexicalization

• A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
• But the desired preference can depend on specific words.
Example of Importance of Lexicalization

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
- But the desired preference can depend on specific words.

\[
\begin{align*}
S & \rightarrow NP \ VP & 0.9 \\
S & \rightarrow \ VP & 0.1 \\
NP & \rightarrow \ Det \ A \ N & 0.5 \\
NP & \rightarrow NP \ PP & 0.3 \\
NP & \rightarrow Prop\ N & 0.2 \\
A & \rightarrow \varepsilon & 0.6 \\
A & \rightarrow Adj \ A & 0.4 \\
PP & \rightarrow \ Prep \ NP & 1.0 \\
VP & \rightarrow V \ NP & 0.7 \\
VP & \rightarrow VP \ PP & 0.3 \\
\end{align*}
\]

English

Head Words

- Syntactic phrases usually have a word in them that is most “central” to the phrase.
- Linguists have defined the concept of a lexical head of a phrase.
- Simple rules can identify the head of any phrase by percolating head words up the parse tree.
  - Head of a VP is the main verb
  - Head of an NP is the main noun
  - Head of a PP is the preposition
  - Head of a sentence is the head of its VP
Lexicalized Productions

- Specialized productions can be generated by including the head word and its POS of each non-terminal as part of that non-terminal’s symbol.

[Diagram of lexicalized productions with examples of sentences: John liked the dog. John put the pen in the dog.]

Lexicalized Productions

[Same diagram as above]
Parameterizing Lexicalized Productions

• Accurately estimating parameters on such a large number of very specialized productions could require enormous amounts of treebank data.
• Need some way of estimating parameters for lexicalized productions that makes reasonable independence assumptions so that accurate probabilities for very specific rules can be learned.

Collins’ Parser

• Collins’ (1999) parser assumes a simple generative model of lexicalized productions.
• Models productions based on context to the left and the right of the head daughter.
  – LHS → $L_n L_{n-1} \ldots L_1 H R_1 \ldots R_{m-1} R_m$
• First generate the head (H) and then repeatedly generate left ($L_i$) and right ($R_i$) context symbols until the symbol STOP is generated.
Sample Production Generation

\[ VP_{put-VBD} \rightarrow VBD_{put-VBD} \text{ NP}_{dog-NN} PP_{in-IN} \]

Note: Penn treebank tends to have fairly flat parse trees that produce long productions.

\[ VP_{put-VBD} \rightarrow \text{ STOP} \ VBD_{put-VBD} \text{ NP}_{dog-NN} PP_{in-IN} \text{ STOP} \]

\[
P_L(\text{STOP} | VP_{put-VBD}) \times P_R(VBD | VP_{put-VBD}) \times P_R(\text{NP}_{dog-NN} | VP_{put-VBD}) \times P_R(PP_{in-IN} | VP_{put-VBD}) \times P_R(\text{STOP} | VP_{put-VBD})
\]

Estimating Production Generation Parameters

- Estimate \( P_H \), \( P_L \), and \( P_R \) parameters from treebank data.

\[
P_R(PP_{in-IN} | VP_{put-VBD}) = \frac{\text{Count}(PP_{in-IN} \text{ right of head in a } VP_{put-VBD} \text{ production})}{\text{Count}(\text{symbol right of head in a } VP_{put-VBD})}
\]

\[
P_R(\text{NP}_{dog-NN} | VP_{put-VBD}) = \frac{\text{Count}(\text{NP}_{dog-NN} \text{ right of head in a } VP_{put-VBD} \text{ production})}{\text{Count}(\text{symbol right of head in a } VP_{put-VBD})}
\]

- Smooth estimates by linearly interpolating with simpler models conditioned on just POS tag or no lexical info.

\[
\text{sm}P_R(PP_{in-IN} | VP_{put-VBD}) = \lambda_1 P_R(PP_{in-IN} | VP_{put-VBD}) + (1-\lambda_1) \left( \lambda_2 P_R(PP_{in-IN} | VP_{VBD}) + (1-\lambda_2) P_R(PP_{in-IN} | VP) \right)
\]
Missed Context Dependence

- Another problem with CFGs is that which production is used to expand a non-terminal is independent of its context.
- However, this independence is frequently violated for normal grammars.
  - NPs that are subjects are more likely to be pronouns than NPs that are objects.

Splitting Non-Terminals

- To provide more contextual information, non-terminals can be split into multiple new non-terminals based on their parent in the parse tree using parent annotation.
  - A subject NP becomes NP^S since its parent node is an S.
  - An object NP becomes NP^VP since its parent node is a VP
Parent Annotation Example

Split and Merge

- Non-terminal splitting greatly increases the size of the grammar and the number of parameters that need to be learned from limited training data.
- Best approach is to only split non-terminals when it improves the accuracy of the grammar.
- May also help to merge some non-terminals to remove some un-helpful distinctions and learn more accurate parameters for the merged productions.
- Method: Heuristically search for a combination of splits and merges that produces a grammar that maximizes the likelihood of the training treebank.
Parsing Evaluation Metrics

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If $P$ is the system’s parse tree and $T$ is the human parse tree (the “gold standard”):
  - $\text{Recall} = \frac{\text{(# correct constituents in } P \text{)}}{\text{( # constituents in } T \text{)}}$
  - $\text{Precision} = \frac{\text{(# correct constituents in } P \text{)}}{\text{( # constituents in } P \text{)}}$

- Labeled Precision and labeled recall require getting the non-terminal label on the constituent node correct to count as correct.

- $F_1$ is the harmonic mean of precision and recall.
Treebank Results

• Results of current state-of-the-art systems on the English Penn WSJ treebank are slightly greater than 90% labeled precision and recall.

Discriminative Parse Reranking

• Motivation: Even when the top-ranked parse not correct, frequently the correct parse is one of those ranked highly by a statistical parser.
• Use a discriminative classifier that is trained to select the best parse from the N-best parses produced by the original parser.
• Reranker can exploit global features of the entire parse whereas a PCFG is restricted to making decisions based on local info.
2-Stage Reranking Approach

- Adapt the PCFG parser to produce an \textit{N-best list} of the most probable parses in addition to the most-likely one.
- Extract from each of these parses, a set of global features that help determine if it is a good parse tree.
- Train a discriminative classifier (e.g. logistic regression) using the best parse in each N-best list as positive and others as negative.

Parse Reranking
Sample Parse Tree Features

- Probability of the parse from the PCFG.
- The number of parallel conjuncts.
  - “the bird in the tree and the squirrel on the ground”
  - “the bird and the squirrel in the tree”
- The degree to which the parse tree is right branching.
  - English parses tend to be right branching (cf. parse of “Book the flight through Houston”)

Evaluation of Reranking

- Reranking is limited by *oracle accuracy*, i.e. the accuracy that results when an omniscient oracle picks the best parse from the N-best list.
- Typical current oracle accuracy is around $F_1=97\%$
- Reranking can generally improve test accuracy of current PCFG models a percentage point or two.
Human Parsing

- Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence.
- Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster.

Garden Path Sentences

- People are confused by sentences that seem to have a particular syntactic structure but then suddenly violate this structure, so the listener is “lead down the garden path”.
  - The horse raced past the barn fell.
    - vs. The horse raced past the barn broke his leg.
  - The complex houses married students.
  - The old man the sea.
  - While Anna dressed the baby spit up on the bed.
Unification Grammars (Ch 15)

- In order to handle agreement issues more effectively, each constituent has a list of features such as number, person, gender, etc. which may or not be specified for a given constituent.
- In order for two constituents to combine to form a larger constituent, their features must **unify**, i.e. consistently combine into a merged set of features.
- Expressive grammars and parsers (e.g. HPSG – head driven phrase structure grammar) have been developed using this approach and have been partially integrated with modern statistical models of disambiguation.

Mildly Context-Sensitive Grammars

- Some grammatical formalisms provide a degree of context-sensitivity that helps capture aspects of NL syntax that are not easily handled by CFGs.

- Combinatory Categorial Grammar (CCG) consists of:
  - **Categorial Lexicon** that associates a syntactic and semantic category with each word.
  - **Combinatory Rules** that define how categories combine to form other categories.
Statistical Parsing Conclusions

- Statistical models such as PCFGs allow for probabilistic resolution of ambiguities.
- PCFGs can be easily learned from treebanks.
- Lexicalization and non-terminal splitting are required to effectively resolve many ambiguities.
- Current statistical parsers are quite accurate but not yet at the level of human-expert agreement.