Statistical Parsing

Chapter 12
(selected sections)

Statistical Parsing

The rise of data and statistics
Pre 1990 ("Classical") NLP Parsing

- Wrote symbolic grammar (CFG or often richer) and lexicon
- This scaled badly and didn’t give coverage.

*Fed raises interest rates 0.5% in effort to control inflation*
- Minimal grammar: 36 parses
- Simple 10 rule grammar: 592 parses
- Real-size broad-coverage grammar: millions of parses

Classical NLP Parsing:
The problem and its solution

- Constraints can be added to grammars to limit unlikely/weird parses for sentences
  - But the attempt make the grammars not robust
    - Commonly 30% of sentences in even an edited text would have no parse.
- A less constrained grammar can parse more sentences
  - But simple sentences end up with ever more parses with no way to choose between them
- We need mechanisms that allow us to find the most likely parse(s) for a sentence
  - Statistical parsing lets us work with very loose grammars that admit millions of parses for sentences but still quickly find the best parse(s)
The rise of annotated data: The Penn Treebank

[Marcus et al. 1993, Computational Linguistics]

\[
\begin{align*}
( & S \\
( & NP-SBJ (DT The) (NN move)) \\
( & VP (VBD followed)) \\
( & NP \\
( & NP (DT a) (NN round)) \\
( & PP (IN of)) \\
( & NP \\
( & NP (JJ similar) (NNS increases)) \\
( & PP (IN by)) \\
( & NP (JJ other) (NNS lenders))) \\
( & PP (IN against)) \\
( & NP (NNP Arizona) (JJ real) (NN estate) (NNS loans)))))) \\
( & . . ))) \\
( & S-ADV \\
( & NP-SBJ (NONE- *)) \\
( & VP (VBG reflecting)) \\
( & NP \\
( & NP (DT a) (VBG continuing) (NN decline)) \\
( & PP-LOC (IN in)) \\
( & NP (DT that) (NN market)))))) \\
( & . . )))
\end{align*}
\]

The rise of annotated data

• Starting off, building a treebank seems a lot slower and less useful than building a grammar

• But a treebank gives us many things
  – Reusability of the labor
    • Many parsers, POS taggers, etc.
    • Valuable resource for linguistics
  – Broad coverage
  – Frequencies and distributional information
  – A way to evaluate systems
**Statistical parsing applications**

Statistical parsers are robust and widely used in applications:

- High precision question answering [Pasca and Harabagiu SIGIR 2001]
- Improving biological named entity finding [Finkel et al. JNLPBA 2004]
- Syntactically based sentence compression [Lin and Wilbur 2007]
- Extracting opinions about products [Bloom et al. NAACL 2007]
- Improved interaction in computer games [Gorniak and Roy 2005]
- Helping linguists find data [Resnik et al. BLS 2005]
- Source sentence analysis for machine translation [Xu et al. 2009]
- Relation extraction systems [Fundel et al. Bioinformatics 2006]

---

**Attachment ambiguities**

- Recall that a key parsing decision is how we ‘attach’ various constituents

```
The board approved [its acquisition] [by Royal Trustco Ltd.]
[of Toronto]
[for $27 a share]
[at its monthly meeting].
```
Two problems to solve:

1. Repeated work (last class)

2. Choosing the correct parse (today’s class)

- How do we work out the correct attachment?

- Words are good predictors of attachment
  - Even absent full understanding

- Our statistical parsers will try to exploit such statistics.
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>0.6</td>
</tr>
<tr>
<td>S → VP</td>
<td>0.1</td>
<td>Noun → book</td>
</tr>
<tr>
<td>NP → Pronoun</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>NP → Proper-Noun</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>NP → Det Nominal</td>
<td>0.6</td>
<td>Verb → book</td>
</tr>
<tr>
<td>Nominal → Noun</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>Nominal → Nominal Noun</td>
<td>0.2</td>
<td>Pronoun → I</td>
</tr>
<tr>
<td>Nominal → Nominal PP</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>VP → Verb</td>
<td>0.2</td>
<td>Proper-Noun → Houston</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>0.5</td>
<td>0.8</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3</td>
<td>Aux → does</td>
</tr>
<tr>
<td>PP → Prep NP</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Lexicon</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>Probability</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>Probability</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>Probability</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>Probability</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>Probability</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Probability</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Probability</td>
<td>0.25</td>
<td>0.25</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.

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

- book the flight through Houston

Syntactic Disambiguation

- Resolve ambiguity by picking most probable parse tree.

\[ P(D_2) = \]

![Diagram of parse tree]

\[ D_2 \]
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.

---

**S → NP VP**
- S → VP: 0.1
- NP → Det A N: 0.5
- NP → NP PP: 0.3
- NP → PropN: 0.2
- A → ε: 0.6
- A → Adj A: 0.4
- PP → Prep NP: 1.0
- VP → V NP: 0.7
- VP → VP PP: 0.3

English:
- 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.

```
S → NP VP    0.9
S → VP       0.1
NP → Det A N 0.5
NP → NP PP    0.3
NP → PropN    0.2
A → ε         0.6
A → Adj A     0.4
PP → Prep NP  1.0
VP → V NP     0.7
VP → VP PP    0.3
```

English

```
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.
(Incomplete) Probabilistic CNF Grammar

S → NP VP
S → X1 VP
X1 → Aux NP
S → book
S → Verb NP
S → VP PP
NP → I | he | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book
Nominal → flight
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book
Verb → book
VP → Verb NP
VP → VP PP
PP → Prep NP

Probabilistic CKY Parser
Probabilistic CKY Parser

Book the flight through Houston

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

Det:.6, Nominal:.15, Noun:.5

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

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

S:.05*.5*.054 =.00135
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).

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.

\[
P(\alpha \rightarrow \beta | \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}
\]
**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”):
  - **Recall** = ($\#$ correct constituents in $P$) / ($\#$ constituents in $T$)
  - **Precision** = ($\#$ correct constituents in $P$) / ($\#$ constituents in $P$)

- **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.

**Evaluating constituency parsing**

Gold standard brackets:  $S\cdot(0\cdot11), NP\cdot(0\cdot2), VP\cdot(2\cdot9), VP\cdot(3\cdot9), NP\cdot(4\cdot6), PP\cdot(6\cdot9), NP\cdot(7\cdot9), NP\cdot(9\cdot10)$

Candidate brackets:  $S\cdot(0\cdot11), NP\cdot(0\cdot2), VP\cdot(2\cdot10), VP\cdot(3\cdot10), NP\cdot(4\cdot6), PP\cdot(6\cdot10), NP\cdot(7\cdot10)$
Evaluating constituency parsing

Gold standard brackets:
S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9), NP-(4:6), PP-(6-9), NP-(7,9), NP-(9:10)

Candidate brackets:
S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:6), PP-(6-10), NP-(7,10)

Labeled Precision 3/7 = 42.9%
Labeled Recall 3/8 = 37.5%
Tagging Accuracy 11/11 = 100.0%

How good are PCFGs?

• Penn WSJ parsing accuracy is high
• Robust
  – Usually admit everything, but with low probability
• Partial solution for grammar ambiguity
  – A PCFG gives some idea of the plausibility of a parse
  – But not so good because the independence assumptions are too strong
• Give a probabilistic language model
  – But in the simple case it performs worse than a trigram model
• The problem seems to be that PCFGs lack the lexicalization of a trigram model
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.

```
S → NP VP  0.9
S → VP     0.1
NP → Det A N 0.5
NP → NP PP  0.3
NP → PropN  0.2
A → ε      0.6
A → Adj A   0.4
PP → Prep NP 1.0
VP → V NP   0.7
VP → VP PP  0.3
```

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.
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.
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.
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.
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.

Try at Home

• http://nlp.cs.berkeley.edu/software.shtml
  – http://tomato.banatao.berkeley.edu:8080/parser/parser.html

• https://nlp.stanford.edu/software/lex-parser.shtml