Today’s Main Points

- Training data
- How to evaluate parsers
- Limitations of PCFGs, enhancements & alternatives
  - Lexicalized PCFGs
  - Structure sensitivity
  - Left-corner parsing
  - Faster parsing with beam search
  - Dependency parsers
- Current state of the art

Treebanks

- Pure Grammar Induction Approaches tend not to produce the parse trees that people want

Solution

- Give an example of parse trees that we want
- Make a learning tool learn a grammar

Treebank

- A collection of such example parses
- PennTreebank is most widely used

Treebanks: labeling data

- Penn Treebank
  - Trees are represented via bracketing
  - Fairly flat structures for Noun Phrases
    (NP Arizona real estate loans)
  - Tagged with grammatical and semantic functions
    (-SBJ, -LOC, …)
  - Use empty nodes(*) to indicate understood subjects and extraction gaps
  - NP-over-NP structure
    - Wrong based on linguistic theory
    - Based on chunk(Abney 1991) view

Treebanks

{ ( S / NP-SBJ ( The move) )
  / VP followed
  / ( NP / NP a round )
  / ( PP of
    / ( NP / NP similar increases )
    / ( PP by
      / ( NP / NP other lenders )
      / ( PP against
        / ( NP / NP Arizona real estate loans )))
  / ( S-ADV / NP-SBJ )
  / VP reflecting
    / ( NP / NP a continuing decline )
    / ( PP-LOC in
      / ( NP / NP that market )))
}

Treebanks
Notion of Chunks (Penn Treebank Style) vs. Modern Syntax Theory

Many people have argued that it is better to have linguists constructing treebanks than grammars:
- Because it is easier to work out the correct parse of sentences than to try to determine what all possible manifestations of a certain rule or grammatical construct are.

Treebanking Issues
- Type of data:
  - Task dependent (newspaper, journals, novels, technical manuals, dialogs, email)
- Size:
  - The more the better! (Resource-limited)
- Parse representation:
  - Dependency vs Parse tree
  - Attributes: What do encode? words, morphology, syntax, semantics...
  - Reference & bookkeeping: date time, who did what

Organizational Issues
- Team:
  - 1 Team leader; bookkeeping/hiring
  - 1 Guideline person
  - 1 Linguistic issues person
  - 3-5 Annotators
  - 1-2 Technical staff/programming
  - 2 Checking persons
- Double annotation if possible.

Treebanking Plan
- The main points (after getting funding):
  - Planning
  - Basic guidelines development
  - Annotation & guidelines refinement
  - Consistency checking, guidelines finalization
  - Packaging and distribution
- Time needed:
  - on the order of 2 years per 1 million words
  - only about 1/3 of the total effort is annotation

Parser Evaluation
Evaluation

Ultimate goal is to build system for IE, QA, MT
People are rarely interested in syntactic analysis for its own sake
Evaluate the system for evaluate the parser
For Simplicity and modularization, and Convenience
Compare parses from a parser with the result of hand parsing of a sentence (gold standard)
What is objective criterion that we are trying to maximize?

Tree Accuracy (Exact match)
It is a very tough standard!!!
But in many ways it is a sensible one to use

PARSEVAL Measures
For some purposes, partially correct parses can be useful
Originally for non-statistical parsers
Evaluate the component pieces of a parse
Measures: Precision, Recall, Crossing brackets

(Labeled) Precision
How many brackets in the parse match those in the correct tree (Gold standard)?

(Labeled) Recall
How many of the brackets in the correct tree are in the parse?

Crossing brackets
Average of how many constituents in one tree cross over constituent boundaries in the other tree

Problems with PARSEVAL
Even vanilla PCFG performs quite well
It measures success at the level of individual decisions
You must make many consecutive decisions correctly to be correct on the entire tree.

Problems with PARSEVAL (2)
Behind story
The structure of Penn Treebank
Flat $\rightarrow$ Few brackets $\rightarrow$ Low Crossing brackets
Troublesome brackets are avoided
$\rightarrow$ High Precision/Recall

The errors in precision and recall are minimal
In some cases wrong PP attachment penalizes Precision, Recall and Crossing Bracket Accuracy minimally.

On the other hand, attaching low instead of high, then every node in the right-branching tree will be wrong: serious harm to Precision, Recall, and Crossing Bracket Accuracy.
Evaluation

Do PARSEVAL measures succeed in real tasks?
Many small parsing mistakes might not affect tasks of semantic interpretation
(Bonnema 1996, 1997)
Tree Accuracy of the Parser: 62%
Correct Semantic Interpretations: 88%
(Hermajakob and Mooney 1997)

At the moment, people feel PARSEVAL measures are adequate for the comparing parsers

Lexicalized Parsing

Limitations of PCFGs

- PCFGs assume:
  - Place invariance
  - Context free: P(rule) independent of
    - words outside span
    - also, words with overlapping derivation
  - Ancestor free: P(rule) independent of
    - Non-terminals above.
- Lack of sensitivity to lexical information
- Lack of sensitivity to structural frequencies

Lack of Lexical Dependency

Means that
P(VP $\rightarrow$ V NP NP) is independent of the particular verb involved!
... but much more likely with ditransitive verbs (like *gave*).

He *gave* the boy a ball.
He *ran* to the store.

The Need for Lexical Dependency

Probabilities dependent on Lexical words
Problem 1: Verb subcategorization
VP expansion is independent of the choice of verb
However ...

<table>
<thead>
<tr>
<th>verb</th>
<th>come</th>
<th>take</th>
<th>think</th>
<th>want</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP $\rightarrow$ V</td>
<td>9.5%</td>
<td>2.6%</td>
<td>4.6%</td>
<td>5.7%</td>
</tr>
<tr>
<td>VP $\rightarrow$ V NP</td>
<td>1.1%</td>
<td>32.1%</td>
<td>0.2%</td>
<td>13.9%</td>
</tr>
<tr>
<td>VP $\rightarrow$ V PP</td>
<td>34.5%</td>
<td>3.1%</td>
<td>7.1%</td>
<td>0.3%</td>
</tr>
<tr>
<td>VP $\rightarrow$ V SBAR</td>
<td>6.6%</td>
<td>0.3%</td>
<td>73.0%</td>
<td>0.2%</td>
</tr>
<tr>
<td>VP $\rightarrow$ V S</td>
<td>2.2%</td>
<td>1.3%</td>
<td>4.8%</td>
<td>70.8%</td>
</tr>
</tbody>
</table>

Including actual words information when making decisions about tree structure is necessary

Weakening the independence assumption of PCFG

Probabilities dependent on PCFG
Problem 2: Phrasal Attachment
Lexical content of phrases provide information for decision
Syntactic category of the phrases provide very little information
Standard PCFG is worse than n-gram models
Weakening the independence assumption of PCFG
Probabilities dependent on Lexical words
Solution
Lexicalize CFG: Each phrasal node with its head word

Another case of PP attachment ambiguity

Another case of PP attachment ambiguity

A case of coordination ambiguity

Heads in Context-Free Rules
Add annotations specifying the "head" of each rule:

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vt=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition
More about heads

- Each context-free rule has one “special” child that is the head of the rule. e.g.,
  
  \[
  \begin{align*}
  S & \Rightarrow \text{NP} \quad \text{VP} \\
  \text{VP} & \Rightarrow \text{Vt} \quad \text{NP} \\
  \text{NP} & \Rightarrow \text{DT} \quad \text{NN} \quad \text{NN}
  \end{align*}
  \]
  
  (VP is the head)
  
  (Vt is the head)
  
  (NN is the head)

- A core idea in linguistics (X-bar Theory, Head-Driven Phrase Structure Grammar)

- Some intuitions:
  
  - The central sub-constituent of each rule.
  
  - The semantic predicate in each rule.

Rules which recover heads:

Example rules for NPs

Adding Headwords to Trees

Adding Headtags to Trees

Explosion of number of rules

New rules might look like:

\[
\begin{align*}
\text{VP} & \Rightarrow \text{gave} \\
\text{VP} & \Rightarrow \text{gave} \quad \text{NP} \quad \text{man} \quad \text{NP} \quad \text{book}
\end{align*}
\]

But this would be a massive explosion in number of rules (and parameters)

Lexicalized Parsing, with smoothing
Lexicalized parsing [Charniak 1997]

- A very simple, conservative model of lexicalized PCFG
- Probabilistic conditioning is “top-down” (but actual computation is bottom-up)

\[
\begin{align*}
\hat{P}(h|ph,c,pc) &= \lambda_1(e)P_{\text{MLE}}(h|ph,c,pc) \\
&+ \lambda_2(e)P_{\text{MLE}}(h|C(ph),c,pc) \\
&+ \lambda_3(e)P_{\text{MLE}}(h|c,pc) + \lambda_4(e)P_{\text{MLE}}(h|c) \\
\end{align*}
\]

- \( \lambda_1(e) \) is here a function of how much one would expect to see a certain occurrence, given the amount of training data, word counts, etc.
- \( C(ph) \) is semantic class of parent headword
- Techniques like these for dealing with data sparseness are vital to successful model construction

[Charniak 1997] smoothing example

\[
\begin{align*}
P(h|ph,c,pc) &= 0.000627 \\
P(h|c,pc) &= 0.00533 \\
P(h|c) &= 0.000557 \\
\end{align*}
\]

- Allows utilization of rich highly conditioned estimates, but smooths when sufficient data is unavailable
- One can’t just use MLEs: one commonly sees previously unseen events, which would have probability 0.

[Charniak 1997] Rule probability with similar smoothing

\[
P(r|h, hc, pc) = \lambda_1(e)P(r|h, hc, pc) \\
&+ \lambda_2(e)P(r|h, hc) \\
&+ \lambda_3(e)P(r|C(h), hc) \\
&+ \lambda_4(e)P(r|hc, pc) \\
&+ \lambda_5(e)P(r|hc) \\
\]

Sparseness and the Penn Treebank

- The Penn Treebank - 1 million words of parsed English
- WSJ – has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - 965,000 constituents, but only 66 WHADJP, of which only 6 aren’t how much or how many, but there is an infinite space of these (how clever/original/incompetent (at risk assessment and evaluation))
- Most of the probabilities that you would like to compute, you can’t compute
**Sparseness and the Penn Treebank**

- Most intelligent processing depends on bilingually statistical likelihoods of relationships between pairs of words.
- Extremely sparse, even on topics central to the WSJ:
  - stocks plummeted  2 occurrences
  - stocks stabilized  1 occurrence
  - stocks skyrocketed  0 occurrences
- So far there has been very modest success augmenting the Penn Treebank with extra unannotated materials or using semantic classes or clusters (cf. Charniak 1997, Charniak 2000) – as soon as there are more than tiny amounts of annotated training data.

**Collins 1997:**

**Markov model out from head**

- Charniak (1997) expands each phrase structure tree in a single step.
- This is good for capturing dependencies between child nodes.
- But it is bad because of data sparseness.
- A pure dependency, one child at a time, model is worse.
- But one can do better by in between models, such as generating the children as a Markov process on both sides of the head (Collins 1997; Charniak 2000)

**Modeling Rule Productions as Markov Processes**

- Step 1: generate category of head child

  \[ S(\text{told},V[6]) \]

  \[ \Downarrow \]

  \[ S(\text{told},V[6]) \]

  \[ \bigg| \text{VP(\text{told},V[6])} \bigg| \]

  \[ P_h(\text{VP} \mid S, \text{told}, V[6]) \]

- Step 2: generate left modifiers in a Markov chain

  \[ S(\text{told},V[6]) \]

  \[ \Downarrow \]

  \[ \text{VP(\text{told},V[6])} \]

  \[ \bigg| \bigg| \text{NP(\text{Hillary},NNP)} \bigg| \bigg| \text{S,VP(\text{told},V[6]),LEFT} \bigg| \bigg| \]

  \[ P_h(\text{VP} \mid S, \text{told}, V[6]) \times P_h(\text{NP(\text{Hillary},NNP)} \mid S, \text{VP(\text{told},V[6]),LEFT}) \times \]

  \[ P_h(\text{NP(yesterday,NN)} \mid S, \text{VP(\text{told},V[6]),LEFT}) \]
Modeling Rule Productions as Markov Processes

- Step 2: generate left modifiers in a Markov chain

\[ P(D) \cdot P(V|D) \cdot P(M|V,D) \cdot P(S|M,V,D) \]

A Refinement: Adding a Distance Variable

- \( \Delta = 1 \) if position is adjacent to the head.

\[ P(S(told,V[6])) \]

Adding the Complement/Adjunct Distinction

- \( Hillary \) is the subject
- \( yesterday \) is a temporal modifier
- But nothing to distinguish them.

Adding Tags Making the Complement/Adjunct Distinction

Adding dependency on structure
Weakening the independence assumption of PCFG

Probabilities dependent on structural context
PCFGs are also deficient on purely structural grounds too
Really context independent?

<table>
<thead>
<tr>
<th>Expansion</th>
<th>% as Subj</th>
<th>% as Obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP → PP</td>
<td>13.7%</td>
<td>2.1%</td>
</tr>
<tr>
<td>NP → NNP</td>
<td>3.5%</td>
<td>0.9%</td>
</tr>
<tr>
<td>NP → DT NN</td>
<td>5.6%</td>
<td>4.6%</td>
</tr>
<tr>
<td>NP → NN</td>
<td>1.4%</td>
<td>2.8%</td>
</tr>
<tr>
<td>NP → NP SBAR</td>
<td>0.5%</td>
<td>2.6%</td>
</tr>
<tr>
<td>NP → NP PP</td>
<td>5.6%</td>
<td>14.1%</td>
</tr>
</tbody>
</table>

Left Corner Parsing

§ Technique for 1 word of lookahead in algorithms like Earley's
§ (can also do multi-word lookahead but it's harder)

Basic Earley's Algorithm

<table>
<thead>
<tr>
<th>0</th>
<th>Papa</th>
<th>1</th>
<th>attach</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ROOT</td>
<td>S</td>
<td>0 NP Papa</td>
</tr>
<tr>
<td>1</td>
<td>NP</td>
<td></td>
<td>1 NP PP</td>
</tr>
<tr>
<td>6</td>
<td>NP</td>
<td>Det</td>
<td>6 NP PP</td>
</tr>
<tr>
<td>7</td>
<td>NP</td>
<td>PP</td>
<td>7 NP PP</td>
</tr>
<tr>
<td>10</td>
<td>NP</td>
<td>Papa</td>
<td>10 Papa</td>
</tr>
<tr>
<td>12</td>
<td>Det</td>
<td>the</td>
<td>12 Det the</td>
</tr>
<tr>
<td>13</td>
<td>Det</td>
<td>a</td>
<td>13 Det a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>Papa</th>
<th>1</th>
<th>predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ROOT</td>
<td>S</td>
<td>0 NP Papa</td>
</tr>
<tr>
<td>7</td>
<td>NP</td>
<td>PP</td>
<td>7 NP PP</td>
</tr>
<tr>
<td>12</td>
<td>NP</td>
<td>Det</td>
<td>12 NP Det</td>
</tr>
<tr>
<td>13</td>
<td>NP</td>
<td>a</td>
<td>13 NP a</td>
</tr>
<tr>
<td>14</td>
<td>Det</td>
<td>the</td>
<td>14 Det the</td>
</tr>
<tr>
<td>15</td>
<td>Det</td>
<td>a</td>
<td>15 Det a</td>
</tr>
</tbody>
</table>
1-word lookahead would help

No point in looking for words or constituents that can’t start with *ate*.
Merging Right-Hand Sides

- Grammar might have rules
  \[ X \rightarrow A \ G \ H \ P \]
  \[ X \rightarrow B \ G \ H \ P \]
- Could end up with both of these in chart:
  (2, X \rightarrow A \ G H P) in column 5
  (2, X \rightarrow B \ G H P) in column 5
- But these are now interchangeable: if one produces X then so will the other
- To avoid this redundancy, can always use dotted rules of this form: X \rightarrow ... G H P

Merging Right-Hand Sides

- Similarly, grammar might have rules
  \[ X \rightarrow A \ G H Q \]
  \[ X \rightarrow A \ G \ H Q \]
- Could end up with both of these in chart:
  (2, X \rightarrow A \ G H P) in column 5
  (2, X \rightarrow A \ G H Q) in column 5
- Not interchangeable, but we'll be processing them in parallel for a while …
- Solution: write grammar as X \rightarrow A \ G H (P|Q)
Merging Right-Hand Sides

- Combining the two previous cases:
  \[ X \rightarrow A \ G \ H \ P \]
  \[ X \rightarrow A \ G \ H \ Q \]
  \[ X \rightarrow B \ G \ H \ P \]
  \[ X \rightarrow B \ G \ H \ Q \]

becomes

\[ X \rightarrow (A \mid B) \ G \ H \ (P \mid Q) \]

- And often nice to write stuff like
  \[ NP \rightarrow (\text{Det} \mid \text{"}) \text{Adj}^* \text{N} \]

These are regular expressions!

Build their minimal DFAs:

Merging Right-Hand Sides

Probabilistic Left-Corner Grammars

Use richer probabilistic conditioning
Rather than conditioning just on parent (as in PCFGs)

\[ P(\text{NP} \rightarrow \text{Det Adj N} \mid \text{NP}) \]

Condition on left corner and goal categories

\[ P(\text{NP} \rightarrow \text{Det Adj N} \mid \text{Det, VP, S}) \]

Allow left-to-right online parsing (which can hope to explain how people build partial interpretations online)

Faster parsing with beam search

- Heuristically throw away constituents that probably won’t make it into a complete parse.
- Use probabilities to decide which ones.
  - So probs are useful for speed as well as accuracy!
- Both safe and unsafe methods exist
  - Throw x away if \( p(x) < 10^{-200} \) (and lower this threshold if we don’t get a parse)
  - Throw x away if \( p(x) < 100 \times p(y) \) for some y that spans the same set of words
  - Throw x away if \( p(x)q(x) \) is small, where q(x) is an estimate of probability of all rules needed to combine x with the other words in the sentence
Dependency Parsing

Phrase Structure Grammars and Dependency Grammars

Phrase Structure Grammar describes the structure of sentences with phrase structure tree

Alternatively, a Dependency grammar describes the structure with dependencies between words

One word is the head of a sentence and all other words are dependent on that word

Dependent on some other word which connects to the headword through a sequence of dependencies

Two key advantages of Dependency grammar are

- Easy to use lexical information
  - Disambiguation decisions are being made directly with words
  - No need to build a large superstructure
- Dependencies are one way of decomposing PS rules
  - Lots of rare flat trees in Penn Treebank → Sparse Data
  - Can get reasonable probabilistic estimate if we decompose it

Evaluation

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCFGs (Charniak 97)</td>
<td>70.6%</td>
<td>74.8%</td>
</tr>
<tr>
<td>Decision trees (Magerman 95)</td>
<td>84.0%</td>
<td>84.3%</td>
</tr>
<tr>
<td>Lexicalized with backoff (Charniak 97)</td>
<td>86.7%</td>
<td>86.6%</td>
</tr>
<tr>
<td>Lexicalized with Markov (Collins 97 M1)</td>
<td>87.5%</td>
<td>87.7%</td>
</tr>
<tr>
<td>&quot; with subcategorization (Collins 97 M2)</td>
<td>88.1%</td>
<td>88.3%</td>
</tr>
<tr>
<td>MaxEnt-inspired (Charniak 2000)</td>
<td>90.1%</td>
<td>90.1%</td>
</tr>
</tbody>
</table>