Algorithms for NLP

Parsing VI

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley
Saksham Singhal -- implemented pseudo-tries. Used implicit caching (stored the most frequent n-grams on top of hash tables) and explicit caching.

Soumya Wadhwa, Tejas Nama -- approximated by ignoring all trigrams with count 1. That dropped BLEU score by less than 0.1 only but freed half the memory!

Craig Stewart -- rehash annealing idea. Made resizing factor and load factor change with every rehash to converge to 0.9 load factor to minimize wasted space.

Griffin Thomas Adams -- Built a "waterfall" tiered cache system

Dean Alderucci -- Built a class to pack data types of arbitrary size into an array of longs. Built a custom implementation of log that ran faster.

Robin Jonathan Algayres -- Context trie!

Raghuram Mandyam Annasamy -- Used database inspired sharding technique on keys

Xianyang Chen -- Compressed hash table and did smarter binary search by indexing chunks with the same last word

Aldrian Obaja -- Implemented NestedMap, achieving 792 MB of memory.

Other things many people did -- LRU caching, packing multiple values (counts and context fertilites) into a single long, binary search instead of hash table.
Grammar Projections

Coarse Grammar

NP
  └── DT
      └── JJ
          └── NN

Fine Grammar

NP^VP
  └── DT^NP
      └── @NP^VP[DT]

NP → DT @NP

NP^VP → DT^NP @NP^VP[DT]

Note: X-Bar Grammars are projections with rules like XP → Y @X or XP → @X Y or @X → X
Efficient Parsing for Structural Annotation
Coarse-to-Fine Pruning

\[ P(X|i, j, S) < \text{threshold} \]

E.g. consider the span 5 to 12:

coarse:                      fine:
...                      CP   NP   VP...

...
For each coarse chart item $X[i,j]$, compute posterior probability:

\[
\frac{\alpha(X, i, j) \cdot \beta(X, i, j)}{\alpha(\text{root}, 0, n)} < \text{threshold}
\]

E.g. consider the span 5 to 12:
Computing Marginals

$$\alpha(X, i, j) = \frac{\sum_{X \to Y Z} \sum_{k \in (i, j)} P(X \to Y Z) \alpha(Y, i, k) \alpha(Z, k, j)}{\sum_{X \to Y Z} \sum_{k \in (i, j)} P(X \to Y Z) \alpha(Y, i, k) \alpha(Z, k, j)}$$
Computing Marginals

\[
\beta(X, i, j) = \sum_{Y \to ZX} \sum_{k \in [0, i]} P(Y \to ZX) \beta(Y, k, j) \alpha(Z, k, i) \\
+ \sum_{Y \to XZ} \sum_{k \in (j, n]} P(Y \to XZ) \beta(Y, i, k) \alpha(Z, j, k)
\]
Efficient Parsing for Lexical Grammars
Lexicalized Trees

- Add “head words” to each phrasal node
  - Syntactic vs. semantic heads
  - Headship not in (most) treebanks
  - Usually use head rules, e.g.:
    - NP:
      - Take leftmost NP
      - Take rightmost N*
      - Take rightmost JJ
      - Take right child
    - VP:
      - Take leftmost VB*
      - Take leftmost VP
      - Take left child

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```

```
S
  /   \\   \\
/     \\
NP    VP
   /   /   \\
/   /     \\
DT  NN    Vt
 the lawyer questioned

SP(questioned)
```
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

\[ VP(\text{saw}) \rightarrow VBD(\text{saw}) \text{ NP-C(her)} \text{ NP(today)} \]

- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps
Lexical Derivation Steps

- A derivation of a local tree [Collins 99]

  Choose a head tag and word

  Choose a complement bag

  Generate children (incl. adjuncts)

  Recursively derive children
Lexicalized CKY

\[
\text{bestScore}(X, i, j, h)
\]

if (j = i + 1)
    return \text{tagScore}(X, s[i])
else
    return \max_k, h', X \rightarrow YZ \\text{score}(X[h] \rightarrow Y[h] \ Z[h']) \times
    \text{bestScore}(Y, i, k, h) \times
    \text{bestScore}(Z, k, j, h')
   
\max_k, h', X \rightarrow YZ \\text{score}(X[h] \rightarrow Y[h'] \ Z[h]) \times
    \text{bestScore}(Y, i, k, h') \times
    \text{bestScore}(Z, k, j, h)
Quartic Parsing

- Turns out, you can do (a little) better [Eisner 99]

- Gives an $O(n^4)$ algorithm
- Still prohibitive in practice if not pruned
Pruning with Beams

- The Collins parser prunes with per-cell beams [Collins 99]
  - Essentially, run the $O(n^5)$ CKY
  - Remember only a few hypotheses for each span $<i,j>$.
  - If we keep $K$ hypotheses at each span, then we do at most $O(nK^2)$ work per span (why?)
  - Keeps things more or less cubic (and in practice is more like linear!)

- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)
The Charniak parser prunes using a two-pass, coarse-to-fine approach [Charniak 97+]

- First, parse with the base grammar
- For each $X:[i,j]$ calculate $P(X|i,j,s)$
  - This isn’t trivial, and there are clever speed ups
- Second, do the full $O(n^5)$ CKY
  - Skip any $X:[i,j]$ which had low (say, < 0.0001) posterior
  - Avoids almost all work in the second phase!

- Charniak et al 06: can use more passes
- Petrov et al 07: can use many more passes
Results

Some results

- Collins 99 – 88.6 F1 (generative lexical)
- Charniak and Johnson 05 – 89.7 / 91.3 F1 (generative lexical / reranked)
- Petrov et al 06 – 90.7 F1 (generative unlexical)
- McClosky et al 06 – 92.1 F1 (gen + rerank + self-train)
Latent Variable PCFGs
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ’98]
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ’98]
- Head lexicalization [Collins ’99, Charniak ’00]
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ’98]
- Head lexicalization [Collins ’99, Charniak ’00]
- Automatic clustering?
Latent Variable Grammars

Parse Tree

Sentence \( w \)

Derivations \( t : T \)

Parameters \( \theta \)
Learning Latent Annotations

EM algorithm:
- Brackets are known
- Base categories are known
- Only induce subcategories

Just like Forward-Backward for HMMs.
Refinement of the DT tag

- DT
  - the (0.50)
  - a (0.24)
  - The (0.08)
  - a (0.61)
  - the (0.19)
  - an (0.11)
  - the (0.80)
  - The (0.15)
  - a (0.01)
  - this (0.39)
  - that (0.28)
  - That (0.11)
  - some (0.20)
  - all (0.19)
  - those (0.12)
Hierarchical refinement
Hierarchical Estimation Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat Training</td>
<td>87.3</td>
</tr>
<tr>
<td>Hierarchical Training</td>
<td>88.4</td>
</tr>
</tbody>
</table>
Refinement of the ‘, tag

- Splitting all categories equally is wasteful:
Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

```
the (0.54)
a (0.25)
The (0.09)
```

```
a (0.61)
the (0.19)
an (0.11)
```

```
the (0.80)
The (0.15)
a (0.01)
```

```
the (0.96)
a (0.01)
The (0.01)
```

```
The (0.93)
A (0.02)
No (0.01)
```
Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>88.4</td>
</tr>
<tr>
<td>With 50% Merging</td>
<td>89.5</td>
</tr>
</tbody>
</table>
Number of Phrasal Subcategories
Number of Lexical Subcategories

![Graph showing the number of lexical subcategories for various parts of speech. The x-axis represents different parts of speech, and the y-axis shows the number of subcategories. The graph highlights the relative frequency of each subcategory, with NNP, NNS, VBN, RB, VB, VBD, CD, PRP, IN, DT, CC, JJ, JJR, JJS, and other categories listed. The y-axis ranges from 0 to 70.]
Learned Splits

- **Proper Nouns (NNP):**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NNP-12</td>
<td>John</td>
<td>Robert</td>
<td>James</td>
</tr>
<tr>
<td>NNP-2</td>
<td>J.</td>
<td>E.</td>
<td>L.</td>
</tr>
<tr>
<td>NNP-1</td>
<td>Bush</td>
<td>Noriega</td>
<td>Peters</td>
</tr>
<tr>
<td>NNP-15</td>
<td>New</td>
<td>San</td>
<td>Wall</td>
</tr>
<tr>
<td>NNP-3</td>
<td>York</td>
<td>Francisco</td>
<td>Street</td>
</tr>
</tbody>
</table>

- **Personal pronouns (PRP):**

<table>
<thead>
<tr>
<th>PRP-0</th>
<th>It</th>
<th>He</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRP-1</td>
<td>it</td>
<td>he</td>
<td>they</td>
</tr>
<tr>
<td>PRP-2</td>
<td>it</td>
<td>them</td>
<td>him</td>
</tr>
</tbody>
</table>
### Learned Splits

- **Relative adverbs (RBR):**

<table>
<thead>
<tr>
<th>RBR-0</th>
<th>further</th>
<th>lower</th>
<th>higher</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBR-1</td>
<td>more</td>
<td>less</td>
<td>More</td>
</tr>
<tr>
<td>RBR-2</td>
<td>earlier</td>
<td>Earlier</td>
<td>later</td>
</tr>
</tbody>
</table>

- **Cardinal Numbers (CD):**

<table>
<thead>
<tr>
<th>CD-7</th>
<th>one</th>
<th>two</th>
<th>Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-4</td>
<td>1989</td>
<td>1990</td>
<td>1988</td>
</tr>
<tr>
<td>CD-11</td>
<td>million</td>
<td>billion</td>
<td>trillion</td>
</tr>
<tr>
<td>CD-0</td>
<td>1</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>CD-3</td>
<td>1</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>CD-9</td>
<td>78</td>
<td>58</td>
<td>34</td>
</tr>
</tbody>
</table>
## Final Results (Accuracy)

<table>
<thead>
<tr>
<th>Language</th>
<th>Method</th>
<th>≤ 40 words F1</th>
<th>all F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td>Charniak&amp;Johnson ‘05 (generative)</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>90.6</td>
<td>90.1</td>
</tr>
<tr>
<td>GER</td>
<td>Dubey ‘05</td>
<td>76.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>80.8</td>
<td>80.1</td>
</tr>
<tr>
<td>CHN</td>
<td>Chiang et al. ‘02</td>
<td>80.0</td>
<td>76.6</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>86.3</td>
<td>83.4</td>
</tr>
</tbody>
</table>

Still higher numbers from reranking / self-training methods
Efficient Parsing for Hierarchical Grammars
Coarse-to-Fine Inference

- Example: PP attachment

```
S
  NP
    PRP
      They
  VP
    ??????????

V
  raised

NP
  DT
    a
  NN
    point

PP
  IN
    of
  NP
    order
```
Hierarchical Pruning

- coarse:
- split in two:
- split in four:
- split in eight:
Bracket Posteriors

Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new S&L bailout agency can raise capital. Creating another potential obstacle to the government's sale of sick thrifts.
1621 min
111 min
35 min
15 min
(no search error)
Other Syntactic Models
Lexicalized parsers can be seen as producing *dependency trees*.

Each local binary tree corresponds to an attachment in the dependency graph.
Pure dependency parsing is only cubic [Eisner 99]

Some work on non-projective dependencies
- Common in, e.g. Czech parsing
- Can do with MST algorithms [McDonald and Pereira 05]
Shift-Reduce Parsers

- Another way to derive a tree:

- Parsing
  - No useful dynamic programming search
  - Can still use beam search [Ratnaparkhi 97]
Tree Insertion Grammars

- Rewrite large (possibly lexicalized) subtrees in a single step

Formally, a tree-insertion grammar

- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable parse is NP-complete
TIG: Insertion

\[
\phi
\]

\[
\psi
\]

\[
\phi'
\]

\[
\psi
\]

\[
S \\
NP \downarrow \\
V \\
saw
\]

\[
S \\
NP \\
D \downarrow N \\
\text{man}
\]

\[
S \\
NP \\
D \downarrow N \\
\text{man} \\
V \\
saw
\]
Tree-adjoining grammars

- Start with local trees
- Can insert structure with adjunction operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- ... as well as other weird stuff that CFGs don’t capture well (e.g. cross-serial dependencies)
TAG: Long Distance
CCG Parsing

- **Combinatory Categorial Grammar**
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

\[
\begin{align*}
  John &\vdash \text{NP} \\
  shares &\vdash \text{NP} \\
  buys &\vdash (S\backslash\text{NP})/\text{NP} \\
  sleeps &\vdash S\backslash\text{NP} \\
  well &\vdash (S\backslash\text{NP})\backslash(S\backslash\text{NP})
\end{align*}
\]

\[
\begin{array}{c}
S \\
\text{NP} \\
John \\
(S\backslash\text{NP})/\text{NP} \\
\text{NP} \\
buys \\
shares
\end{array}
\]
Empty Elements
Empty Elements

- In the PTB, three kinds of empty elements:
  - Null items (usually complementizers)
  - Dislocation (WH-traces, topicalization, relative clause and heavy NP extraposition)
  - Control (raising, passives, control, shared argumentation)

- Need to reconstruct these (and resolve any indexation)
Example: English

Farmers was quick yesterday TO point out the problems

NP-3
NNP VBD ADJP NP S-2
Farmers was JJ S NN NP VP
quick *ICH*-2 yesterday *-3 TO to VP

NP
NP

SBAR

NP

WHNP-1

S

NP

VP

PRP VBZ NP

it sees *T*-1
Example: German

Erst nicht until
lange Zeit long time
später later

wird will

*T2*

PROAV *T1* begun

damit with it
den RMV the RMV
to zu schaffen

to form
# Types of Empties

<table>
<thead>
<tr>
<th>Antecedent</th>
<th>POS</th>
<th>Label</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>NP</td>
<td>*</td>
<td>18,334</td>
<td>NP trace (e.g., *Sam was seen *)</td>
</tr>
<tr>
<td></td>
<td>NP</td>
<td>*</td>
<td>9,812</td>
<td>NP PRO (e.g., *to sleep is nice)</td>
</tr>
<tr>
<td>WHNP</td>
<td>NP</td>
<td><em>T</em></td>
<td>8,620</td>
<td>WH trace (e.g., the woman who you saw <em>T</em>)</td>
</tr>
<tr>
<td></td>
<td><em>U</em></td>
<td></td>
<td>7,478</td>
<td>Empty units (e.g., $25 <em>U</em>)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td>5,635</td>
<td>Empty complementizers (e.g., Sam said 0 Sasha snores)</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td><em>T</em></td>
<td>4,063</td>
<td>Moved clauses (e.g., Sam had to go, Sasha explained <em>T</em>)</td>
</tr>
<tr>
<td>WHADVP</td>
<td>ADVP</td>
<td><em>T</em></td>
<td>2,492</td>
<td>WH-trace (e.g., Sam explained how to leave <em>T</em>)</td>
</tr>
<tr>
<td></td>
<td>SBAR</td>
<td></td>
<td>2,033</td>
<td>Empty clauses (e.g., Sam had to go, Sasha explained (SBAR))</td>
</tr>
<tr>
<td></td>
<td>WHNP</td>
<td>0</td>
<td>1,759</td>
<td>Empty relative pronouns (e.g., the woman 0 we saw)</td>
</tr>
<tr>
<td></td>
<td>WHADVP</td>
<td>0</td>
<td>575</td>
<td>Empty relative pronouns (e.g., no reason 0 to leave)</td>
</tr>
</tbody>
</table>

The diagram illustrates the syntactic structure of some sentences, showing how empty categories are represented and how they relate to the overall sentence structure.
A Pattern-Matching Approach

- [Johnson 02]
Pattern-Matching Details

- Something like transformation-based learning
- Extract patterns
  - Details: transitive verb marking, auxiliaries
  - Details: legal subtrees
- Rank patterns
  - Pruning ranking: by correct / match rate
  - Application priority: by depth
- Pre-order traversal
- Greedy match
<table>
<thead>
<tr>
<th>Count</th>
<th>Match</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>5816</td>
<td>6223</td>
<td>(S (NP (NONE- *) ) VP)</td>
</tr>
<tr>
<td>5605</td>
<td>7895</td>
<td>(SBAR (NONE- 0) S)</td>
</tr>
<tr>
<td>5312</td>
<td>5338</td>
<td>(SBAR WHNP-1 (S (NP (NONE- T*1)) VP))</td>
</tr>
<tr>
<td>4434</td>
<td>5217</td>
<td>(NP QP (NONE- U*))</td>
</tr>
<tr>
<td>1682</td>
<td>1682</td>
<td>(NP $ CD (NONE- U*))</td>
</tr>
<tr>
<td>1327</td>
<td>1593</td>
<td>(VP VBN_t (NP (NONE- ) PP)</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
<td>(ADJP QP (NONE- U*))</td>
</tr>
<tr>
<td>662</td>
<td>1219</td>
<td>(SBAR (WHNP-1 (NONE- 0) (S (NP (NONE- T*1)) VP))</td>
</tr>
<tr>
<td>618</td>
<td>635</td>
<td>(S S-1 , NP (VP VBD (SBAR (NONE- 0) (S (NONE- T*1)))) .)</td>
</tr>
<tr>
<td>499</td>
<td>512</td>
<td>(SINV <code> </code> S-1 , <code> </code> (VP VBZ (S (NONE- T*1))) NP .)</td>
</tr>
<tr>
<td>361</td>
<td>369</td>
<td>(SINV <code> </code> S-1 , <code> </code> (VP VBD (S (NONE- T*1))) NP .)</td>
</tr>
<tr>
<td>352</td>
<td>320</td>
<td>(S NP-1 (VP VBZ (S (NP (NONE- T*1)) VP)))</td>
</tr>
<tr>
<td>346</td>
<td>273</td>
<td>(S NP-1 (VP AUX (VP VBN_t (NP (NONE- T*1)) PP)))</td>
</tr>
<tr>
<td>322</td>
<td>467</td>
<td>(VP VBD_t (NP (NONE- ) PP)</td>
</tr>
<tr>
<td>269</td>
<td>275</td>
<td>(S <code> </code> S-1 , <code> </code> NP (VP VBD (S (NONE- T*1))) .)</td>
</tr>
<tr>
<td>Empty node POS</td>
<td>Label</td>
<td>Section 23</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$P$</td>
</tr>
<tr>
<td>(Overall)</td>
<td></td>
<td>0.93</td>
</tr>
<tr>
<td>NP</td>
<td>*</td>
<td>0.95</td>
</tr>
<tr>
<td>NP</td>
<td><em>T</em></td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td><em>U</em></td>
<td>0.92</td>
</tr>
<tr>
<td>S</td>
<td><em>T</em></td>
<td>0.98</td>
</tr>
<tr>
<td>ADVP</td>
<td><em>T</em></td>
<td>0.91</td>
</tr>
<tr>
<td>SBAR</td>
<td></td>
<td>0.90</td>
</tr>
<tr>
<td>WHNP</td>
<td>0</td>
<td>0.75</td>
</tr>
</tbody>
</table>
Semantic Roles
Semantic Role Labeling (SRL)

- Characterize clauses as relations with roles:

  \([\text{Judge She }] \text{ blames } [\text{Evaluator the Government }] [\text{Reason for failing to do enough to help}]\).

  Holman would characterise this as \textit{blaming} \([\text{Evaluator the poor}]\).

  The letter quotes Black as saying that \([\text{Judge white and Navajo ranchers}]\) misrepresent their livestock losses and \textit{blame} \([\text{Reason everything}] [\text{Evaluator on coyotes}]\).

- Says more than which NP is the subject (but not much more):
- Relations like \textit{subject} are syntactic, relations like \textit{agent} or \textit{message} are semantic

- Typical pipeline:
  - Parse, then label roles
  - Almost all errors locked in by parser
  - Really, SRL is quite a lot easier than parsing
He heard the sound of liquid slurping in a metal container as Farrell approached him from behind.
FrameNet: roles shared between verbs
PropBank: each verb has its own roles
PropBank more used, because it’s layered over the treebank (and so has greater coverage, plus parses)
Note: some linguistic theories postulate fewer roles than FrameNet (e.g. 5-20 total: agent, patient, instrument, etc.)
Sales fell to $251.2 million from $278.7 million.

arg1: Sales
rel: fell
arg4: to $251.2 million
arg3: from $278.7 million
rotate.02 sense: shift from one thing to another
roles: Arg0: causer of shift
Arg1: thing being changed
Arg2: old thing
Arg3: new thing

Many of Wednesday’s winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said. (wsj_1723)
arg0: investors
rel: rotated
arg1: their buying
arg3: to other issues
PropBank Example

**aim.01** sense: intend, plan
roles: Arg0: aimer, planner
Arg1: plan, intent

The Central Council of Church Bell Ringers aims *trace* to improve relations with vicars. *(wsj_0089)*

arg0: The Central Council of Church Bell Ringers
rel: aims
arg1: *trace* to improve relations with vicars

**aim.02** sense: point (weapon) at
roles: Arg0: aimer
Arg1: weapon, etc.
Arg2: target

Banks have been aiming packages at the elderly.
arg0: Banks
rel: aiming
arg1: packages
arg2: at the elderly
Shared Arguments

(NP-SBJ (JJ massive) (JJ internal) (NN debt))
  (VP (VBZ has))
  (VP (VBN forced))
  (S)
    (NP-SBJ-1 (DT the) (NN government))
  (VP)
    (VP (TO to))
    (VP (VB borrow))
    (ADVP-MNR (RB massively))...
Path Features

Path Features

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB↑VP↓PP</td>
<td>PP argument/adjunct</td>
</tr>
<tr>
<td>VB↑VP↑S↓NP</td>
<td>subject</td>
</tr>
<tr>
<td>VB↑VP↓NP</td>
<td>object</td>
</tr>
<tr>
<td>VB↑VP↑VP↑S↓NP</td>
<td>subject (embedded VP)</td>
</tr>
<tr>
<td>VB↑VP↓ADVP</td>
<td>adverbial adjunct</td>
</tr>
<tr>
<td>NN↑NP↑NP↑NP↓PP</td>
<td>prepositional complement of noun</td>
</tr>
</tbody>
</table>
Results

- **Features:**
  - Path from target to filler
  - Filler’s syntactic type, headword, case
  - Target’s identity
  - Sentence voice, etc.
  - Lots of other second-order features

- **Gold vs parsed source trees**
  - SRL is fairly easy on gold trees
  - Harder on automatic parses

<table>
<thead>
<tr>
<th></th>
<th>CORE</th>
<th></th>
<th>ARG M</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>92.2</td>
<td>Acc.</td>
<td>89.9</td>
</tr>
<tr>
<td>84.1</td>
<td>66.5</td>
<td>F1</td>
<td>81.4</td>
</tr>
</tbody>
</table>
Assume the number of parses is very small

We can represent each parse $T$ as a feature vector $\varphi(T)$

- Typically, all local rules are features
- Also non-local features, like how right-branching the overall tree is
- [Charniak and Johnson 05] gives a rich set of features
K-Best Parsing

[Huang and Chiang 05, Pauls, Klein, Quirk 10]