Inducing a Grammar from CCGbank

- Grammar (lexicon) can be read off the leaves of the trees
Chart Parsing with CCG

- **Stage 1**
  - Assign POS tags and lexical categories to words in the sentence
  - Use taggers to assign the POS tags and categories
    - based on standard Maximum Entropy tagging techniques
- **Stage 2**
  - Combine the categories using the combinatory rules
  - Can use standard bottom-up CKY chart-parsing algorithm
- **Stage 3**
  - Find the highest scoring derivation according to some model
    - e.g. generative model, CRF, perceptron
  - Viterbi algorithm finds this efficiently
CCG Supertagging

He goes on the road with his piano

A bitter conflict with global implications

- Baseline tagging accuracy is $\approx 72\%$
  - baseline is to assign tag most frequently seen with word in training data, and assign $N$ to unseen words
- Baseline for Penn Treebank POS tagging is $\approx 90\%$
CCG Multitagging

- Per-word tagging accuracy is \( \approx 92\% \)
- Potentially assign more than one category to a word
  - assign all categories whose probability is within some factor \( \beta \) of the highest probability category
- Accuracy is over 97% at only 1.4 categories per word
- Accuracy is now high enough to serve as a front-end to the parser
CKY Algorithm

chart[i][j] is a cell containing categories spanning words from i to i + j

initialise chart with categories of span 1 (lexical categories)

LOOP over span of result category (j = 2 to SENT_LENGTH)
LOOP over start position of left combining category (i = 0 to SENT_LENGTH - j)
  LOOP over span of left combining category (k = 1 to j - 1)
    chart[i][j] += Combine(chart[i][k], chart[i + k][j - k])
Chart Parsing

- DP algorithms can be run over the packed representation
- The *Viterbi* algorithm finds the highest scoring derivation
Linear Parsing Model

Score\( (d, S') = \sum_i \lambda_i f_i(d) = \bar{\lambda} \cdot \phi(d) \)

- Features are counts over \( d \)
  - root category of \( d \) (plus lexical head)
  - \( \langle \)lexical category, lexical item\( \rangle \) pairs
  - rule feature: \( S \rightarrow NP \ S \backslash NP \) (plus lexical head)
  - predicate argument dependency: \( \text{subj}(\text{bought, IBM}) \) (plus distance)
  - “Back-king-off” features with words replaced by POS tags

- Use Perceptron training to set the weights
Training Data from CCGbank

subj(persuades, Marks)
obj(persuades, Brooks)
subj(merge, Brooks)
to-inf(persuades, merge)
Feature Representation

\[ f_i : D \rightarrow \mathcal{N} \quad (3000000 \leq i \leq 1) \]
Linear Parsing Model

\[
\text{Score}(d, s) = \sum \lambda_i f_i(d) = \bar{\lambda} \cdot \bar{f}(d)
\]

- \(f_i\) are the features (defined by hand)
- \(\lambda_i\) are the corresponding weights (which need to be learned)
Perceptron Training

\[
\text{Score}(d, S) = \sum_i \lambda_i f_i(d) = \bar{\lambda} \cdot \phi(d)
\]

**Inputs:** training examples \((x_i, y_i)\)

**Initialisation:** set \(\bar{\lambda} = 0\)

**Algorithm:**

\[
\text{for } t = 1..T, i = 1..N
\]

\[
\text{calculate } z_i = \arg \max_{y \in \text{GEN}(x_i)} \Phi(x_i, y) \cdot \bar{\lambda}
\]

\[
\text{if } z_i \neq y_i
\]

\[
\bar{\lambda} = \bar{\lambda} + \Phi(x_i, y_i) - \Phi(x_i, z_i)
\]

**Outputs:** \(\bar{\lambda}\)
### Perceptron Training

The diagram represents the training process of a perceiver, where the top level (5) is the highest level of analysis. The training starts from the bottom level (1) and moves upwards, predicting the correct output based on the input features. The output vector $w_0$ is defined as:

$$w_0 = <0,0,0,0,0,0,0,0,0>$$

The diagram shows the transition rules from one level to the next, starting from the input features at the bottom and moving up to the final output at the top.
Perceptron Training
Perceptron Training

Update weights:

\[ W_1 = <0, 1, 0, ..., -1, 0, ..., -1, 0, -1, ..., 0> \]

- \( f_1, f_20, f_{55}, f_{100}, f_{210}, f_{345} \)
- \( f_{19}, f_{25}, f_{75}, f_{150}, f_{211}, f_{346}, f_{450}, f_{500}, f_{525} \)
- \( f_{15}, f_{21}, f_{56}, f_{120}, f_{212}, f_{348}, f_{419} \)
Perceptron Training

$W_1 = <0,1,0,...,-1,0,...,-1,0,1,0,-1,...,0>$
Perceptron Training

$W_1 = <0, 1, 0, ..., -1, 0, ..., -1, 0, 1, 0, -1, ..., 0>$

$f_{11}, f_{21}, f_{57}, f_{90}, f_{145}, f_{250}$

$f_{21}, f_{25}, f_{76}, f_{151}, f_{222}, f_{348}, f_{444}, f_{507}, f_{575}$

$f_{17}, f_{45}, f_{155}, f_{167}, f_{678}$
Perceptron Training

$W_2 = \langle 0, 2, -1, ..., -1, 1, ..., -1, 0, 1, 0, -2, ..., -1 \rangle$

$f_{11}, f_{21}, f_{57}, f_{90}, f_{145}, f_{250}$
$f_{21}, f_{25}, f_{76}, f_{151}, f_{222}, f_{348}, f_{444}, f_{507}, f_{575}$
$f_{17}, f_{45}, f_{155}, f_{167}, f_{678}$
DP vs. Beam Search

• DP requires the optimal sub-problem property
• For efficient parsing this restricts the feature set
• An alternative is to apply a beam to each cell
• Now no restrictions on the features
• Max-violation perceptron used for training
Parser Evaluation

- Compare output of the parser with a *gold standard*
- Exact match metric sometimes used but a little crude
- Partial match against a set of *grammatical relations* currently the method of choice
  - measures recovery of semantically important relations
  - relatively theory-neutral representation
Head-based GRs

- *She gave the present to Kim*
  (ncsubj gave She _)
  (dobj gave present)
  (iobj gave to)
  (dobj to Kim)
  (det present the)

- *The company wants to wean itself away from expensive gimmicks*
  (xcomp to wants wean)
  (iobj wean from)
  (ncmod prt wean away)
  (dobj wean itself)
  (dobj from gimmicks)
  (ncmod _ gimmicks expensive)
  ...

Mapping CCG Dependencies to GRs

- Argument slots in CCG dependencies are mapped to GRs

<table>
<thead>
<tr>
<th>CCG lexical category</th>
<th>arg slot</th>
<th>GR</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S[dcl]\NP_1)/NP_2</td>
<td>1</td>
<td>(nsubj %l %f)</td>
</tr>
<tr>
<td>(S[dcl]\NP_1)/NP_2</td>
<td>2</td>
<td>(dobj %l %f)</td>
</tr>
<tr>
<td>(NP/NP_1)/NP_2</td>
<td>1</td>
<td>(prep %f %l)</td>
</tr>
<tr>
<td>(NP/NP_1)/NP_2</td>
<td>2</td>
<td>(pobj %l %f)</td>
</tr>
<tr>
<td>NP[nb]/N_1</td>
<td>1</td>
<td>(det %f %l)</td>
</tr>
</tbody>
</table>

- Mapping is many-to-many
Test Suite: DepBank

- 700 sentences of newspaper text manually annotated with GRSs
- Calculate precision and recall over GRSs

\[
\text{Prec} = \frac{\# \text{ correct}}{\# \text{ proposed by parser}} \quad \text{Rec} = \frac{\# \text{ correct}}{\# \text{ in gold standard}}
\]

\[
F\text{-score} = \frac{2 PR}{P + R}
\]
## Parsing Accuracy

<table>
<thead>
<tr>
<th>GR</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ncsobj</td>
<td>79.6</td>
</tr>
<tr>
<td>dobj</td>
<td>87.7</td>
</tr>
<tr>
<td>obj2</td>
<td>66.7</td>
</tr>
<tr>
<td>iobj</td>
<td>73.4</td>
</tr>
<tr>
<td>clausal</td>
<td>75.0</td>
</tr>
<tr>
<td>ncmcd</td>
<td>76.1</td>
</tr>
<tr>
<td>aux</td>
<td>92.8</td>
</tr>
<tr>
<td>det</td>
<td>95.1</td>
</tr>
<tr>
<td>conj</td>
<td>77.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prec</th>
<th>Rec</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>84.1</td>
<td>82.8</td>
<td>83.4</td>
</tr>
</tbody>
</table>