Parsing Fast and Deep
with a wide-coverage lexicalised-grammar parser

Stephen Clark
Natural Language and Information Processing Research Group
University of Cambridge Computer Laboratory

DISI, Trento
October 2011
Penn Treebank Parsing

```
S
   NP
   DT the
   JJ proposed
   NNS changes
   VP
   RB also
   MD would
   VP
   VB allow
   S
   NP
   NNS executives
   VP
   TO to
   VB report
   NP
   NNS exercises
   PP
   P of
   NP
   NNS options
   ADVP
   RB early
   CONJP
   CC and
   RB often
```

Parsing Fast and Deep
A More Difficult Parsing Task

the same *stump* which had impaled the car of many a guest in the past thirty years and which he refused to have *removed*
Large-Scale Parsing Requirements

- As the volume of available data increases, we would like to have efficient parsers which can analyse it.
- But surely deep and fast parsing are incompatible?
Large-Scale Parsing Requirements

- As the volume of available data increases, we would like to have efficient parsers which can analyse it.
- But surely deep and fast parsing are incompatible?
- The C&C CCG parser can recover unbounded dependencies and analyse Wikipedia in a few hours (using 90 CPUs).
History of the C&C Parser

- Began as an EPSRC project in 2000 in Edinburgh with Mark Steedman and Julia Hockenmaier
- Collaboration with James Curran started in 2002
- Unbounded dependency evaluation with Laura Rimell and Mark
- Parser efficiency work with James and the JHU-09 workshop team
Today’s Talk

• Combinatory Categorial Grammar (CCG)
• Parsing with CCG
• Unbounded dependency evaluation
• Self-training for highly efficient parsing
Combinatory Categorial Grammar (CCG)

- CCG (Steedman) is a lexicalised grammar
- An elementary syntactic structure – for CCG a lexical category – is assigned to each word in a sentence
  
  \textit{walked}: \texttt{S\textbackslash NP} ‘give me an NP to my left and I return a sentence’

- A small number of rules define how categories can combine
CCG Lexical Categories

- Atomic categories: $S$, $N$, $NP$, $PP$, ... (not many more)
- Complex categories are built recursively from atomic categories and slashes, which indicate the directions of arguments
CCG Lexical Categories

• Atomic categories: $S$, $N$, $NP$, $PP$, ... (not many more)

• Complex categories are built recursively from atomic categories and slashes, which indicate the directions of arguments

• Example complex categories for verbs
  • intransitive verb: $S\backslash NP$ walked
  • transitive verb: $(S\backslash NP)/NP$ respected
  • ditransitive verb: $((S\backslash NP)/NP)/NP$ gave
A Simple CCG Derivation

\[
\text{interleukin} - 10 \quad \text{inhibits} \quad \text{production}
\]

\[
\text{NP} \quad (S\backslash \text{NP})/\text{NP} \quad \text{NP}
\]
A Simple CCG Derivation

\[
\text{interleukin} - 10 \quad \text{inhibits} \quad \text{production}
\]

\[
NP \quad (S\backslash NP)/NP \quad NP
\]

\[
S\backslash NP \quad \rightarrow
\]

> forward application
A Simple CCG Derivation

\[
\text{interleukin} - 10 \quad \text{inhibits} \quad \text{production}
\]

\[
\begin{align*}
NP & \quad (S\backslash NP)/NP & \quad NP \\
\frac{\text{forward application}}{S\backslash NP} & \quad \frac{\text{backward application}}{S}
\end{align*}
\]
Classical Categorial Grammar

- ‘Classical’ Categorial Grammar only has application rules
- Classical Categorial Grammar is context free

```
S
  /\   \\
NP  (S\NP)/NP  NP
  |      |     |
interleukin-10 inhibits production
```
Classical Categorial Grammar

- ‘Classical’ Categorial Grammar only has application rules
- Classical Categorial Grammar is context free

 Parsing Fast and Deep
A More Interesting CCG Derivation

\[
\begin{align*}
\text{The} & \quad \text{company} \quad \text{which} \quad \text{Microsoft} \quad \text{bought} \\
NP/N & \quad N & (NP\backslash NP)/(S/NP) & NP & (S\backslash NP)/NP
\end{align*}
\]
A More Interesting CCG Derivation

The company which Microsoft bought

\[
\begin{align*}
\text{NP}/N & \quad N \quad (\text{NP}/\text{NP})/(\text{S}/\text{NP}) \\
\text{NP}/\text{NP} & \quad (\text{S}/\text{NP})/\text{NP} \\
\text{S}/(\text{S}/\text{NP}) & \quad \to \text{T}
\end{align*}
\]

> T type-raising
A More Interesting CCG Derivation

The company which Microsoft bought

\[
\begin{array}{cccc}
\text{NP} / N & N & (\text{NP} \setminus \text{NP}) / (S \setminus \text{NP}) & \text{NP} / (S \setminus \text{NP}) / \text{NP} \\
\text{S} / (S \setminus \text{NP}) & S / \text{NP} & > B
\end{array}
\]

> T type-raising
> B forward composition
A More Interesting CCG Derivation

```
The company which Microsoft bought
```

```
NP/N N (NP\NP)/(S/NP) NP (S/NP)/NP
```

```
S/(S/NP) \rightarrow^T S/NP \rightarrow^B NP/NP
```

```
Parsing Fast and Deep
```
A More Interesting CCG Derivation

The company which Microsoft bought

\[
\begin{align*}
\text{NP}/N & \quad \text{N} \\
\text{NP} & \quad (\text{NP}/\text{NP})/(\text{S}/\text{NP}) \\
\text{NP} & \quad (\text{S}/\text{NP})/\text{NP} \\
\text{S}/(\text{S}/\text{NP}) & \quad \text{T} \\
\text{S}/\text{NP} & \quad \text{B} \\
\text{NP}/\text{NP} & \quad < \\
\text{NP} &
\end{align*}
\]
A **CCG** Question Derivation

\[
\begin{align*}
Whom & \quad \text{did} & \quad he & \quad marry \\
S[wq]/(S[q]/NP) & \quad (S[q]/(S[b]\ NP))/NP & \quad NP & \quad (S[b]\ NP)/NP
\end{align*}
\]
A CCG Question Derivation

\[
\begin{align*}
\text{Whom} & \quad \text{did} \quad \text{he} \quad \text{marry} \\
S[wq]/(S[q]/NP) & \quad (S[q]/(S[b]\NP))/NP & \quad NP \quad (S[b]\NP)/NP \\
& \quad S[q]/(S[b]\NP) \quad > \\
& \quad > \quad \text{forward application}
\end{align*}
\]
A CCG Question Derivation

**Whom** did he marry

\[
S[wq]/(S[q]/NP) 
\quad \frac{(S[q]/(S[b]\backslash NP))/NP \quad NP \quad (S[b]\backslash NP)/NP}{S[q]/(S[b]\backslash NP)} \quad \Rightarrow \quad S[q]/NP 
\]

\[ > \quad \text{forward application} \]

\[ > \quad \text{B} \quad \text{forward composition} \]
A CCG Question Derivation

\[
\begin{align*}
\text{Whom} & \quad \text{did} & \quad \text{he} & \quad \text{marry} \\
S[wq]/(S[q]/NP) & \quad (S[q]/(S[b]\ NP))/NP & \quad NP & \quad (S[b]\ NP)/NP \\
& \quad S[q]/(S[b]\ NP) & \quad \rightarrow \\
& \quad S[q]/NP & \quad \rightarrow \mathbf{B} \\
& \quad S[wq] & \quad \rightarrow \\
\end{align*}
\]

> forward application

> B  forward composition
A Treebank of CCG Derivations

- 40k sentences of newspaper text annotated with CCG derivations
- CCG version of the Penn Treebank (Hockenmaier)
Inducing a Grammar

• For a lexicalised grammar, the grammar essentially is the lexicon – plus a small number of manually defined combinatory rules

• Lexicon can be read off the leaves of the trees
Inducing a Grammar

- ≈ 1200 lexical category types in the CCG treebank
  - this set has high coverage on unseen newspaper data
- In addition to the grammar, the treebank provides training data for the statistical models
Parsing with CCG

- **Stage 1**
  - Assign lexical categories to words in the sentence
  - Use a finite-state *supertagger* to assign the categories
Parsing with CCG

- **Stage 1**
  - Assign lexical categories to words in the sentence
  - Use a finite-state *supertagger* to assign the categories

- **Stage 2**
  - Combine the categories using the combinatory rules
  - Can use standard bottom-up chart-parsing or shift-reduce algorithm
Parsing with CCG

- **Stage 1**
  - Assign lexical categories to words in the sentence
  - Use a finite-state *supertagger* to assign the categories

- **Stage 2**
  - Combine the categories using the combinatory rules
  - Can use standard bottom-up chart-parsing or shift-reduce algorithm

- **Stage 3**
  - Find the highest scoring derivation according to some model
    - e.g. generative model, CRF, perceptron
  - Viterbi algorithm finds this efficiently for the chart; used beam-search for the shift-reduce algorithm
Lexical Category Sequence for Newspaper Sentence

\[
\begin{align*}
\text{In} & \quad \text{an} & \quad \text{Oct.} & \quad 19 & \quad \text{review} & \quad \text{of} & \quad \text{The} \\
\text{(S/S)/NP} & \quad \text{NP[nb]/N} & \quad \text{N/N} & \quad \text{N/N} & \quad \text{N} & \quad \text{(NP\,\,NP\,\,NP)/NP} & \quad \text{NP[nb]/N} \\
\text{Misanthrope} & \quad \text{at} & \quad \text{Chicago} & \quad \text{'}s & \quad \text{Goodman} & \quad \text{Theatre} & \quad \text{−LRB−} \\
\text{N} & \quad \text{(NP\,\,NP)/NP} & \quad \text{N} & \quad \text{(NP[nb]/N)/NP} & \quad \text{N/N} & \quad \text{N} & \quad \text{(NP\,\,NP)/S[dcl]} \\
\text{Revitalized} & \quad \text{Classics} & \quad \text{Take} & \quad \text{the} & \quad \text{Stage} & \quad \text{in} & \quad \text{Windy} \\
\text{N/N} & \quad \text{N} & \quad \text{(S[dcl]\,\,NP)/NP} & \quad \text{NP[nb]/N} & \quad \text{N} & \quad \text{(S\,\,NP)/(S\,\,NP)} & \quad \text{N/N} \\
\text{City} & \quad \text{Leisure} & \quad \text{&} & \quad \text{Arts} & \quad \text{−RRB−} & \quad \text{,} \\
\text{N} & \quad \text{(S\,\,S)/(S\,\,S)} & \quad \text{(S\,\,S)/(S\,\,S)} & \quad \text{S\,\,S} & \quad \text{RRB} & \quad \text{,} \\
\text{the} & \quad \text{role} & \quad \text{of} & \quad \text{Celimene} & \quad \text{,} & \quad \text{played} & \quad \text{by} \\
\text{NP[nb]/N} & \quad \text{N} & \quad \text{(NP\,\,NP)/NP} & \quad \text{N} & \quad \text{S[pss]\,\,NP} & \quad \text{(S\,\,NP)/(S\,\,NP)} & \quad \text{NP} \\
\text{Kim} & \quad \text{Cattrall} & \quad \text{,} & \quad \text{was} & \quad \text{mistakenly} & \quad \text{attributed} & \quad \text{to} \\
\text{N/N} & \quad \text{N} & \quad \text{(S[dcl]\,\,NP)/(S[pss]\,\,NP)} & \quad \text{(S\,\,NP)/(S\,\,NP)} & \quad \text{(S[pss]\,\,NP)/PP} & \quad \text{PP/NP} \\
\text{Christina} & \quad \text{Haag} & \quad \text{.} \\
\text{N/N} & \quad \text{N} & \quad \text{.} \\
\end{align*}
\]
### CCG Supertagging

- **He goes on the road with his piano**

  - $\text{np} (S[dcl]\text{np})/\text{pp}$
  - $\text{PP/np}$
  - $\text{np/n}$
  - $\text{n} ((S\text{ np})((S\text{ np}))/\text{np}$
  - $\text{np/n}$
  - $\text{n}$

  - $\text{A bitter conflict with global implications}$

  - $\text{np/n}$
  - $\text{n/n}$
  - $\text{n}$
  - $\text{(np\text{ np})/np}$
  - $\text{n/n}$
  - $\text{n}$

- **Baseline tagging accuracy is $\approx 72\%$**
A Maximum Entropy Supertagger

- Maximum Entropy tagging method can be applied to CCG supertagging
- Features are the words and POS tags in a 5-word window, plus the two previously assigned categories
- Per-word tagging accuracy is $\approx 92\%$
- Multi-tagger can assign categories with a per-word accuracy of 97.9% at only 2.0 categories per word on average
Parsing with CCG

- **Stage 1**
  - Assign lexical categories to words in the sentence
  - Use a finite-state *supertagger* to assign the categories
- **Stage 2**
  - Combine the categories using the combinatory rules
  - Can use standard *bottom-up chart-parsing* or shift-reduce algorithm
- **Stage 3**
  - Find the highest scoring derivation according to some model
    - e.g. generative model, CRF, perceptron
  - *Viterbi* algorithm finds this efficiently for the chart; used beam-search for the shift-reduce algorithm
Chart Parsing

- CKY applies naturally to CCG because it is binary branching
- Use of supertagger means that we can pack the complete chart \textit{without any pruning}
- DP algorithms can be run over the chart including Viterbi
Feature Representation

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

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

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

- \(f_i\) are the features (defined by hand)
- \(\lambda_i\) are the corresponding weights
- Weights can be learned using the perceptron (in a few hours)
- Similar results are obtained with a log-linear (CRF) model using maximum likelihood estimation
Linear Parsing Model

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

- \( f_i \) are the features (defined by hand)
- \( \lambda_i \) are the corresponding weights
- Weights can be learned using the perceptron (in a few hours)
- Similar results are obtained with a log-linear (CRF) model using maximum likelihood estimation
- **Nothing CCG-specific about the parsing algorithm nor the model** – except that supertagger allows practical discriminative estimation
Possible Outputs

- CCG derivations
- CCG dependencies
- Grammatical Relations
- Logical forms (Johan Bos’ Boxer)
Accuracy Evaluation

- Natural evaluation for CCG parsers is to measure accuracy of dependency recovery (Grammatical Relations)

```
John       hit            the             ball               with              the                 bat
 SUBJ       DET             DET             DOBJ               PREP
```

Parsing Fast and Deep
Grammatical Relations

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

- *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)
  ...

Parsing Fast and Deep
Accuracy Evaluation

- Compare parser output with gold standard GRs on 560 newspaper sentences (Briscoe and Carroll version of DepBank)
- Overall accuracy is 83.4% F-score
- This is state-of-the-art for grammatical relation recovery
- CCG parser also compares favourably with PTB parsers on PTB parsing task
Recovery of Unbounded Dependencies

We have also developed techniques for recognizing and locating underground nuclear tests through the waves in the ground which they generate.

By Monday, they hope to have a sheaf of documents both sides can trust.

By means of charts showing wave-travel times and depths in the ocean at various locations, it is possible to estimate the rate of approach and probable time of arrival at Hawaii of a tsunami getting under way at any spot in the Pacific.
Test Set of Unbounded Dependencies

- 7 constructions, each with 100 sentences
  - 80 sentences for testing, 20 for development
  - 560 test sentences in total
- Roughly half from WSJ section of the PTB and half from Brown
- Corpus built by pattern matching on PTB and manual correction
The 7 Constructions

<table>
<thead>
<tr>
<th>Construction</th>
<th>WSJ</th>
<th>Brown</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object relative clause</td>
<td>2.3</td>
<td>1.1</td>
<td>1.4</td>
</tr>
<tr>
<td>Object reduced relative</td>
<td>2.7</td>
<td>2.8</td>
<td>2.8</td>
</tr>
<tr>
<td>Subject relative clause</td>
<td>10.1</td>
<td>5.7</td>
<td>7.4</td>
</tr>
<tr>
<td>Free relative</td>
<td>2.6</td>
<td>0.9</td>
<td>1.3</td>
</tr>
<tr>
<td>Right Node Raising</td>
<td>2.2</td>
<td>0.9</td>
<td>1.2</td>
</tr>
<tr>
<td>Subject embedded</td>
<td>2.0</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>Questions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Examples of the 7 Constructions

Object extraction from a relative clause
Each must match Wismans pie with the fragment that he carries with him. 
dobj(carrries, fragment)

Object extraction from a reduced relative clause
Put another way, the decline in the yield suggests stocks have gotten pretty rich in price relative to the dividends they pay, some market analysts say. 
dobj(pay, dividends)

Subject extraction from a relative clause
It consists of a series of pipes and a pressure-measuring chamber which record the rise and fall of the water surface. 
nsubj(record, series)
nsubj(record, chamber)
Examples of the 7 Constructions

**Free relative**
He tried to ignore what his own common sense told him, but it wasn't possible; her motives were too blatant.
dobj(told, what)

**Object wh-question**
What city does the Tour de France end in?
pobj(in, city)

**Right node raising**
For the third year in a row, consumers voted Bill Cosby first and James Garner second in persuasiveness as spokesmen in TV commercials, according to Video Storyboard Tests, New York.
prep(first, in)
prep(second, in)
Examples of the 7 Constructions

Subject extraction from an embedded clause
In assigning to God the responsibility which he learned could not rest with his doctors, Eisenhower gave evidence of that weakening of the moral intuition which was to characterize his administration in the years to follow. nsubj(rest, responsibility)
## Results

<table>
<thead>
<tr>
<th></th>
<th>Obj RC</th>
<th>Obj Red</th>
<th>Sbj RC</th>
<th>Free</th>
<th>Obj Q</th>
<th>RNR</th>
<th>Sbj Embed</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;C</td>
<td>59.3</td>
<td>62.6</td>
<td>80.0</td>
<td>72.6</td>
<td>(81.2)</td>
<td>27.5</td>
<td>49.4</td>
<td>22.4</td>
</tr>
<tr>
<td>Enju</td>
<td>47.3</td>
<td>65.9</td>
<td>82.1</td>
<td>76.2</td>
<td>32.5</td>
<td>47.1</td>
<td>32.9</td>
<td>32.9</td>
</tr>
<tr>
<td>DCU</td>
<td>23.1</td>
<td>41.8</td>
<td>56.8</td>
<td>46.4</td>
<td>27.5</td>
<td>40.8</td>
<td>5.9</td>
<td>35.7</td>
</tr>
<tr>
<td>Rasp</td>
<td>16.5</td>
<td>1.1</td>
<td>53.7</td>
<td>17.9</td>
<td>27.5</td>
<td>34.5</td>
<td>15.3</td>
<td>25.3</td>
</tr>
<tr>
<td>Stanford</td>
<td>22.0</td>
<td>1.1</td>
<td>74.7</td>
<td>64.3</td>
<td>41.2</td>
<td>45.4</td>
<td>10.6</td>
<td>38.1</td>
</tr>
</tbody>
</table>

Parsing Fast and Deep
Summary of Results

- Parsing performance on unbounded dependencies much lower than overall parsing accuracies which typically get reported.
- C&C and Enju – robust parsers based on ‘deep’ linguistic formalisms – are the best performers.
- Nivre et al. 2010 evaluates the Nivre and McDonald dependency parsers on this corpus.
  - Overall performance a few % lower than C&C.
Efficiency Improvements

- Parser is already surprisingly fast (40 sentences per second)
  - because of supertagger and highly optimised C++ code
- Can we make it faster still?
- Yes, almost twice as fast:
  - with some pruning of the chart (Zhang et al., 2010)
  - with a novel use of self-training
Self-Training for the CCG Parser

- Self-training loop:
  1. Parse lots of unlabelled data
  2. Retrain supertagger on parser output
  3. Use new supertagger as part of supertagger/parser system

- Self-training has been used before to improve accuracy
- Here we are using self-training to improve efficiency
Self-Training to Improve Efficiency

- Efficiency of the parser depends crucially on the number of lexical categories supplied by the supertagger for each word.
- The best we can do is one lexical category per word.
- How to make the parser as fast as possible (no accuracy loss): supply the lexical category that the parser will end up choosing.
- How can we train the supertagger to provide this category? train it on parser output!
A New Supertagger Task

Previously, watch imports were

\[
\begin{array}{cccc}
\text{S/S} & \text{N/N} & \text{N} & (S[dcl]/NP)/(S[pss]/NP) \\
\text{N} & \text{N} & (S[dcl]/NP)/NP & (S[dcl]/NP)/(S[adj]/NP)
\end{array}
\]

denied such duty-free treatment

\[
\begin{array}{cccc}
(S[pss]/NP)/NP & \text{NP/NP} & \text{N/N} & \text{N} \\
S[pss]/NP & (N/N)/(N/N) & \text{N/N} \\
(S[adj]/NP)/NP & N/N \\
(S[pt]/NP)/NP \\
(S[dcl]/NP)/NP
\end{array}
\]
Results

- Train supertagger on 4M parsed sentences from various domains
- Now the supertagger can confidently provide close to 1 lexical category per word
- Parser speed improvements on all domains, with no accuracy loss:
  - Newswire from 39.6 to **73.3** sentences per second
  - Wikipedia from 50.9 to **73.9** sentences per second
  - Biomedical from 35.1 to **59.3** sentences per second
Conclusion

• Robust, wide coverage parser applied to a variety of text types:
  • newspaper, biomedical, wikipedia, questions
• Uses a formalism-based grammar allowing it to recover long-range, as well as local, dependencies
• Can parse Wikipedia (1 billion words) in a few hours (with access to 90 CPUs)
C&C tools

- Available at:
  http://svn.ask.it.usyd.edu.au/trac/candc/wiki

- All papers available from my web page:
  http://www.cl.cam.ac.uk/~sc609/pubs.html