Towards Broad Coverage Surface Realization with CCG

Michael White, Rajakrishnan Rajkumar, Scott Martin

Department of Linguistics
The Ohio State University
{mwhite, raja, scott}@ling.ohio-state.edu

UCNLG+MT Workshop
11 September 2007
### The Talk in a Nutshell

- Progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (CCG; Steedman, 2000)

---

**Towards Broad Coverage CCG Realization**
The Talk in a Nutshell

- Progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (CCG; Steedman, 2000)
- Respectable initial results (after doing many of the tedious and obvious tasks)
Progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (CCG; Steedman, 2000)

Respectable initial results (after doing many of the tedious and obvious tasks)

Details on the corpus-based grammar engineering process
The Talk in a Nutshell

- Progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (CCG; Steedman, 2000)
- Respectable initial results (after doing many of the tedious and obvious tasks)
- Details on the corpus-based grammar engineering process
- Lessons learned: supertags can help n-grams
Progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (CCG; Steedman, 2000)

Respectable initial results (after doing many of the tedious and obvious tasks)

Details on the corpus-based grammar engineering process

Lessons learned: supertags can help n-grams — but, need a generation supertagger!
The Talk in a Nutshell

- Progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (CCG; Steedman, 2000)
- Respectable initial results (after doing many of the tedious and obvious tasks)
- Details on the corpus-based grammar engineering process
- Lessons learned: supertags can help n-grams — but, need a generation supertagger!
- Much left to explore
<table>
<thead>
<tr>
<th>Why CCG?</th>
</tr>
</thead>
</table>

- State of the art statistical parsing results (Clark and Curran, 2007)
Why CCG?

- State of the art statistical parsing results (Clark and Curran, 2007)
- CCGbank (Hockenmaier, 2003) — a corpus of CCG derivations created by transforming the Penn Treebank
Why CCG?

- State of the art statistical parsing results (Clark and Curran, 2007)
- CCGbank (Hockenmaier, 2003) — a corpus of CCG derivations created by transforming the Penn Treebank
- OpenCCG (White, 2006) chart realizer
  - Used with small, precise grammars in various dialogue systems
  - Supports disjunctive logical forms ("packed" inputs)
  - Has API for statistical scoring models
  - Represents words as factor bundles (form, stem, POS, supertag, etc.)
Why CCG?

- State of the art statistical parsing results (Clark and Curran, 2007)
- CCGbank (Hockenmaier, 2003) — a corpus of CCG derivations created by transforming the Penn Treebank
- OpenCCG (White, 2006) chart realizer
  - Used with small, precise grammars in various dialogue systems
  - Supports disjunctive logical forms ("packed" inputs)
  - Has API for statistical scoring models
  - Represents words as factor bundles (form, stem, POS, supertag, etc.)

⇒ So just turn the crank with the CCGbank, no?
Why CCG?

- State of the art statistical parsing results (Clark and Curran, 2007)
- CCGbank (Hockenmaier, 2003) — a corpus of CCG derivations created by transforming the Penn Treebank
- OpenCCG (White, 2006) chart realizer
  - Used with small, precise grammars in various dialogue systems
  - Supports disjunctive logical forms (“packed” inputs)
  - Has API for statistical scoring models
  - Represents words as factor bundles (form, stem, POS, supertag, etc.)

⇒ So just turn the crank with the CCGbank, no? *(Actually, need to add semantics — cf. Bos’s Boxer — and improve OpenCCG’s performance with large grammars)*
Unlike Halogen (Langkilde-Geary, 2002) and FUF/SURGE (Callaway, 2003), OpenCCG uses a bidirectional grammar.
Unlike Halogen (Langkilde-Geary, 2002) and FUF/SURGE (Callaway, 2003), OpenCCG uses a bidirectional grammar. More similar to HPSG/LFG approaches (Carroll and Oepen, 2005; Nakanishi et al., 2005; Cahill and van Genabith, 2006), except for our greater emphasis on more traditional logical forms.
Two Similar Dependency Graphs

(a) The design (is|’s) based on the Funny Day collection by Villeroy and Boch.

(b) The design (is|’s) based on Villeroy and Boch’s Funny Day series.
(c) The design (is|’s) based on (the Funny Day (collection|series) by Villeroy and Boch | Villeroy and Boch’s Funny Day (collection|series)).
LF in Hybrid Logic Dependency Semantics (HLDS)

@e(be ∧ ⟨TENSE⟩pres ∧ ⟨MOOD⟩dcl ∧
⟨ARG⟩(d ∧ design ∧ ⟨DET⟩the ∧ ⟨NUM⟩sg) ∧
⟨PROP⟩(p ∧ based_on ∧
⟨ARTIFACT⟩d ∧
⟨SOURCE⟩(c ∧ collection ∧ ⟨DET⟩the ∧ ⟨NUM⟩sg ∧
⟨HASPROP⟩(f ∧ Funny_Day) ∧
⟨CREATOR⟩(v ∧ V&B))))

(a)
Disjunctive LF in HLDS

\[@_e (be ∧ ⟨TENSE⟩pres ∧ ⟨MOOD⟩dcl ∧ \\ ⟨ARG⟩(d ∧ design ∧ ⟨DET⟩the ∧ ⟨NUM⟩sg) ∧ \\ ⟨PROP⟩(p ∧ based_on ∧ \\ ⟨ARTIFACT⟩d ∧ \\ ⟨SOURCE⟩(c ∧ ⟨NUM⟩sg ∧ (⟨DET⟩the)? ∧ \\ (collection ∨ series) ∧ \\ ⟨HASPROP⟩(f ∧ Funny_Day) ∧ \\ ((⟨CREATOR⟩[_v] ∨ ⟨GENOWNER⟩[_v])))) \\
∧ @_v (Villeroy_and_Boch)
(c)\]
Flattening

(2) 0: \(e(be)\), 1: \(e(TENSE\text{pres})\), 2: \(e(MOOD\text{dcl})\), 3: \(e(ARG\text{d})\), 4: \(d\text{design}\), 5: \(d\text{DETthe}\), 6: \(d\text{NUMsg}\), 7: \(e\text{PROPp}\), 8: \(p\text{based_on}\), 9: \(p\text{ARTIFACTd}\), 10: \(p\text{SOURCEc}\), 11: \(c\text{NUMsg}\), 12: \(c\text{DETthe}\), 13: \(c\text{collection}\), 14: \(c\text{series}\), 15: \(c\text{HASPROPf}\), 16: \(f\text{Funny.Day}\), 17: \(c\text{CREATORv}\), 18: \(c\text{GENOWNERv}\), 19: \(v\text{Villeroy_and_Boch}\)

(3) \(alt_{0,0} = \{13\}; \ alt_{0,1} = \{14\}\)  
\(alt_{1,0} = \{17, 19\}; \ alt_{1,1} = \{18, 19\}\)  
\(opt_0 = \{12\}\)
Edges

Packed Edges

- In packing mode, a *representative* edge maintains a list of alternative edges whose signs have the same category (but different word sequences)
- Representative edges stand in for their alternative edges during chart construction
Edges

Packed Edges

- In packing mode, a representative edge maintains a list of alternative edges whose signs have the same category (but different word sequences)
- Representative edges stand in for their alternative edges during chart construction

The Disjunctive Case

Inspired by Shemtov (1997); see INLG-06 paper for details...
Lexical Instantiation

(4)  a. \{11, 13, 14\} \textit{collection} \vdash n_c : \\
\quad @_c(\textit{collection}) \land @_c(\langle \text{NUM} \rangle \text{sg})

b. \{11, 13, 14\} \textit{series} \vdash n_c : \\
\quad @_c(\textit{series}) \land @_c(\langle \text{NUM} \rangle \text{sg})

c. \{17\} \text{alt}_{1,0} \textit{by} \vdash n_c \backslash n_c / np_v : \\
\quad @_c(\langle \text{CREATOR} \rangle v)

d. \{18\} \text{alt}_{1,1} \textit{'s} \vdash np_c / n_c \backslash np_v : \\
\quad @_c(\langle \text{GENOWNER} \rangle v)

e. \{19\} \text{alt}_{1,0}; \text{alt}_{1,1} \textit{Villeroy and Boch} \vdash np_v : \\
\quad @_v(\text{V&B})
Derivation

1. \{8-10\} \textit{based_on} \vdash s_p \setminus np_d/np_c
2. \{12\} \textit{the} \vdash np_c/n_c
3. \{15, 16\} \textit{Funny_Day} \vdash n_c/n_c
4. \{11, 13, 14\} \textit{collection} \vdash n_c
   \{11, 13, 14\} \textit{series} \vdash n_c
5. \{17\} alt_{1,0} \textit{by} \vdash n_c \setminus n_c/np_v
6. \{18\} alt_{1,1} \text{'}s \vdash np_c/n_c \setminus np_v
7. \{19\} alt_{1,0}; alt_{1,1} \textit{Villeroy_and_Boch} \vdash np_v
8. \{11, 13-16\} \textit{FD [collection]} \vdash n_c \ (3 \ 4 \,>)

9. \{17-19\} by \textit{V&B} \vdash n_c \backslash n_c \ (5 \ 7 \,>)

10. \{17-19\} \textit{V&B 's} \vdash np_c \slash n_c \ (7 \ 6 \, <)

11. \{11, 13-19\} \textit{FD [coll.] by V&B} \vdash n_c \ (8 \ 9 \, <)

12. \{11, 13-19\} \textit{V&B 's FD [coll.]} \vdash np_c \ (10 \ 8 \, >)

13. \{11-19\} the \textit{FD [coll.] by V&B} \vdash np_c \ (2 \ 11 \, >)

\{11-19\} \textit{V&B 's FD [coll.]} \vdash np_c \ (12 \ \text{optC})

14. \{8-19\} \textit{b._on [the FD [coll.] ...]} \vdash s_p \backslash np_d \ (1 \ 13 \, >)
Unpacking

- Complete edges are unpacked bottom-up, a la Langkilde (2000)
- Pruning and scoring configured via API
- At present, edges are pruned only within equivalence classes, during the unpacking stage
Unpacking

- Complete edges are unpacked bottom-up, à la Langkilde (2000)
- Pruning and scoring configured via API
- At present, edges are pruned only within equivalence classes, during the unpacking stage — this ensures that pruning does not cause the realizer to fail (i.e., fail to find a complete derivation)
Unpacking

- Complete edges are unpacked bottom-up, a la Langkilde (2000)
- Pruning and scoring configured via API
- At present, edges are pruned only within equivalence classes, during the unpacking stage — this ensures that pruning does not cause the realizer to fail (i.e., fail to find a complete derivation)
- But, with large grammars, considering all lexical category assignments often leads to inordinately large charts
Anytime Best-First Search

- In the anytime best-first mode, the packing and unpacking stages are essentially interleaved.
- The search can be cut off after configurable time limits, without first completing the packed chart.
Anytime Best-First Search

- In the anytime best-first mode, the packing and unpacking stages are essentially interleaved.
- The search can be cut off after configurable time limits, without first completing the packed chart.
- If no complete realization is found within the time limit, fragments are greedily assembled.
Greedy Fragment Assembly

1. Start with the best partial realization (by semantic coverage)
Greedy Fragment Assembly

1. Start with the best partial realization (by semantic coverage)

2. Successively select the best partial realization whose semantic coverage is disjoint from those selected so far
Greedy Fragment Assembly

1. Start with the best partial realization (by semantic coverage)
2. Successively select the best partial realization whose semantic coverage is disjoint from those selected so far
3. Again starting with the original best partial realization, greedily concatenate the remaining selected edges (by score), trying both orders
Corpus Conversion

- Transform CCGbank to reflect lexicalized treatment of coordination assumed in newer, multimodal version of CCG (Baldridge and Kruijff, 2003)
Corpus Conversion

- Transform CCGbank to reflect lexicalized treatment of coordination assumed in newer, multimodal version of CCG (Baldridge and Kruijff, 2003)
- Reanalyze punctuation along the lines of Doran’s (1998) lexicalized TAG analysis
Corpus Conversion

- Transform CCGbank to reflect lexicalized treatment of coordination assumed in newer, multimodal version of CCG (Baldridge and Kruijff, 2003)
- Reanalyze punctuation along the lines of Doran’s (1998) lexicalized TAG analysis
- Change unification constraints to support semantic rather than surface syntactic dependencies (complementizers, infinitival-to, expletive subjects, case-marking prepositions)
Corpus Conversion

- Transform CCGbank to reflect lexicalized treatment of coordination assumed in newer, multimodal version of CCG (Baldridge and Kruijff, 2003)
- Reanalyze punctuation along the lines of Doran’s (1998) lexicalized TAG analysis
- Change unification constraints to support semantic rather than surface syntactic dependencies (complementizers, infinitival-to, expletive subjects, case-marking prepositions)

⇒ Viewed as a grammar engineering process! (And accordingly, implemented in a general fashion as an XSLT pipeline)
Grammar Extraction

From the converted CCGbank, a lexico-grammar is extracted and augmented with logical forms

- Extracted categories, unary rules and lexical assignments must meet specified frequency thresholds
From the converted CCGbank, a lexico-grammar is extracted and augmented with logical forms

- Extracted categories, unary rules and lexical assignments must meet specified frequency thresholds
- For unseen open class words, lexical smoothing assigns most frequent categories for POS
Grammar Extraction

From the converted CCGbank, a lexico-grammar is extracted and augmented with logical forms

- Extracted categories, unary rules and lexical assignments must meet specified frequency thresholds
- For unseen open class words, lexical smoothing assigns most frequent categories for POS
- Logical forms are inserted using around two dozen XSLT templates, with numbered semantic roles (a la PropBank)
Example Logical Form Insertion Templates

(1) \[s_{1:dcl} \backslash np_2/np_3 \implies s_{1:dcl,x_1} \backslash np_{2:x_2}/np_{3:x_3} : \text{@}_x_1(\ast\text{pred}\ast \land \langle\text{ARG0}\rangle x_2 \land \langle\text{ARG1}\rangle x_3)\]

(2) \[s_{1:pss} \backslash np_2 \implies s_{1:pss,x_1} \backslash np_{2:x_2} : \text{@}_x_1(\ast\text{pred}\ast \land \langle\text{ARG1}\rangle x_2)\]

(3) \[np_{1}/n_1 \implies np_{1:x_1}/n_{1:x_1} : \text{@}_x_1(\langle\text{DET}\rangle (d \land \ast\text{pred}\ast))\]

(4) \[np_{1}/n_1 \backslash np_2 \implies np_{1:x_1}/n_{1:x_1} \backslash np_{2:x_2} : \text{@}_x_1(\langle\text{GENOWN}\rangle x_2)\]
Creating Dev/Train/Test Files

- To obtain logical form inputs for the realizer, the extracted grammar is used to constrain parse the corpus files.
- When the gold standard derivation succeeds, the resulting logical form is saved.
- Sentence-internal punctuation is skipped when necessary.
Coverage

Paper/Current:

<table>
<thead>
<tr>
<th></th>
<th>LF created</th>
<th>single root</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev (00)</td>
<td>95.1% / 98.0%</td>
<td>67.4% / 76.4%</td>
</tr>
<tr>
<td>test (23)</td>
<td>94.3% / 96.0%</td>
<td>69.7% / 77.2%</td>
</tr>
</tbody>
</table>
Coverage

Paper/Current:

<table>
<thead>
<tr>
<th>test set</th>
<th>LF created</th>
<th>single root</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev (00)</td>
<td>95.1% / 98.0%</td>
<td>67.4% / 76.4%</td>
</tr>
<tr>
<td>test (23)</td>
<td>94.3% / 96.0%</td>
<td>69.7% / 77.2%</td>
</tr>
</tbody>
</table>

- LFs with multiple roots have missing dependencies
Coverage

Paper/Current:

<table>
<thead>
<tr>
<th></th>
<th>LF created</th>
<th>single root</th>
</tr>
</thead>
<tbody>
<tr>
<td>test set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dev (00)</td>
<td>95.1% / 98.0%</td>
<td>67.4% / 76.4%</td>
</tr>
<tr>
<td>test (23)</td>
<td>94.3% / 96.0%</td>
<td>69.7% / 77.2%</td>
</tr>
</tbody>
</table>

- LFs with multiple roots have missing dependencies
- Problems usually due to LF templates, but have found bugs in CCGbank
Exact regeneration

⇒ Also helpful to look at whether sentence can be exactly regenerated with oracle n-grams, from target string

<table>
<thead>
<tr>
<th></th>
<th>grammar</th>
<th>complete</th>
<th>exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsj_0003</td>
<td>wsj_0003</td>
<td>86.7%</td>
<td>86.7%</td>
</tr>
<tr>
<td>dev</td>
<td></td>
<td>76.7%</td>
<td>70.0%</td>
</tr>
<tr>
<td>train</td>
<td></td>
<td>63.3%</td>
<td>56.7%</td>
</tr>
<tr>
<td>dev (00)</td>
<td>dev</td>
<td>59.1%</td>
<td>53.4%</td>
</tr>
<tr>
<td>train</td>
<td></td>
<td>46.6%</td>
<td>37.7%</td>
</tr>
</tbody>
</table>
N-gram Models

- Factored trigram models over words, part-of-speech tags and supertags
- Data from standard training sections (02–21)
- SRILM toolkit
- Null (arbitrary choice) baseline
Word, POS and Supertag Models

\[
p(\vec{F}_i^n) \approx \prod_{i=1}^{n} p(\vec{F}_i | \vec{F}_{i-2}, \vec{F}_{i-1})
\]

\[
p^W(\vec{F}_i | \vec{F}_{i-2}) = p(W_i | W_{i-2}, W_{i-1})
\]
\[
p^P(\vec{F}_i | \vec{F}_{i-2}) = p(P_i | P_{i-2}, P_{i-1})
\]
\[
p^S(\vec{F}_i | \vec{F}_{i-2}) = p(S_i | P_{i-2}, P_{i-1})
\]
Chained, Interpolated Models

\[ p^{PS}(F_i \mid F_{i-2}^{i-1}) = p(P_i \mid P_{i-2}^{i-1})p(S_i \mid P_{i-2}^i) \]  \hspace{1cm} (3)

\[ p^{W+P}(F_i \mid F_{i-2}^{i-1}) = \lambda_1 p^W(F_i \mid F_{i-2}^{i-1}) + \lambda_2 p^P(F_i \mid F_{i-2}^{i-1}) \hspace{1cm} (4) \]

\[ p^{W+PS}(F_i \mid F_{i-2}^{i-1}) = \lambda_1 p^W(F_i \mid F_{i-2}^{i-1}) + \lambda_2 p^{PS}(F_i \mid F_{i-2}^{i-1}) \]
## Initial Non-Blind Development Results

<table>
<thead>
<tr>
<th>scoring model</th>
<th>exact</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>word 3g + pos 3g * stag 3g</td>
<td>14.8%</td>
<td>0.6615</td>
</tr>
<tr>
<td>word 3g + pos 3g</td>
<td>13.7%</td>
<td>0.6407</td>
</tr>
<tr>
<td>word 3g, interp. Kneser-Ney</td>
<td>12.2%</td>
<td>0.6247</td>
</tr>
<tr>
<td>word 3g, Good-Turing</td>
<td>11.7%</td>
<td>0.6219</td>
</tr>
<tr>
<td>pos 3g * supertag 3g</td>
<td>10.6%</td>
<td>0.6042</td>
</tr>
<tr>
<td>supertag 3g</td>
<td>10.0%</td>
<td>0.5886</td>
</tr>
<tr>
<td>pos 3g</td>
<td>8.0%</td>
<td>0.5413</td>
</tr>
<tr>
<td>null</td>
<td>5.1%</td>
<td>0.5251</td>
</tr>
</tbody>
</table>
## Initial Results With Usual Splits

<table>
<thead>
<tr>
<th>test set</th>
<th>scoring model</th>
<th>exact</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev</td>
<td>w3g + pos3g * stag3g</td>
<td>8.1%</td>
<td>0.5578</td>
</tr>
<tr>
<td></td>
<td>word 3g + pos 3g</td>
<td>7.1%</td>
<td>0.5210</td>
</tr>
<tr>
<td></td>
<td>word 3g, Kneser-Ney</td>
<td>6.5%</td>
<td>0.4872</td>
</tr>
<tr>
<td></td>
<td>null</td>
<td>2.2%</td>
<td>0.3697</td>
</tr>
<tr>
<td>test</td>
<td>w3g + pos3g * stag3g</td>
<td>9.8%</td>
<td>0.5768</td>
</tr>
<tr>
<td></td>
<td>word 3g, Kneser-Ney</td>
<td>6.9%</td>
<td>0.5178</td>
</tr>
</tbody>
</table>
Updated Results With Best Model

Paper/Current:

<table>
<thead>
<tr>
<th>test set</th>
<th>condition</th>
<th>exact</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev</td>
<td>non-blind</td>
<td>14.8% / 24.3%</td>
<td>0.6615 / 0.7317</td>
</tr>
<tr>
<td></td>
<td>usual</td>
<td>8.1% / 12.4%</td>
<td>0.5578 / 0.6101</td>
</tr>
<tr>
<td>test</td>
<td>usual</td>
<td>9.8% / 13.0%</td>
<td>0.5768 / 0.6223</td>
</tr>
</tbody>
</table>
## BLEU Comparison (PTB 23)

(N.B.: direct comparison difficult!)

<table>
<thead>
<tr>
<th></th>
<th>coverage</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCCG (07)</td>
<td>96.0%</td>
<td>0.6223</td>
</tr>
<tr>
<td>Cahill &amp; van Genabith (06)</td>
<td>98.5%</td>
<td>0.6651</td>
</tr>
<tr>
<td>Langkilde-Geary (02), ‘Permute, no dir’</td>
<td>83%</td>
<td>0.757</td>
</tr>
<tr>
<td>Nakanishi et al. (05), ≤ 20w</td>
<td>90.8%</td>
<td>0.7733</td>
</tr>
</tbody>
</table>
Need a Supertagger for Realization!

⇒ Search errors revealed by generating with oracle n-grams (from target string), vs. best model

Oracle/Best:

<table>
<thead>
<tr>
<th>test set</th>
<th>grammar</th>
<th>complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsj_0003</td>
<td>wsj_0003</td>
<td>86.7% / 86.7%</td>
</tr>
<tr>
<td>dev</td>
<td>dev</td>
<td>76.7% / 76.7%</td>
</tr>
<tr>
<td>train</td>
<td></td>
<td>63.3% / 10.0%</td>
</tr>
<tr>
<td>dev (00)</td>
<td>dev</td>
<td>59.1% / 57.0%</td>
</tr>
<tr>
<td>train</td>
<td></td>
<td>46.6% / 21.3%</td>
</tr>
</tbody>
</table>
Once realizations are generally satisfactory, BLEU scores may no longer be useful in measuring progress.
Once realizations are generally satisfactory, BLEU scores may no longer be useful in measuring progress.

- With MT outputs, Callison-Burch et al. (2006) contend that improved BLEU scores are neither necessary nor sufficient to achieve better human evaluation scores.
- Stent et al. (2005) suggest that BLEU is a poor judge of fluency with generators that aim to produce desirable variation (e.g., in discourse).
Example: Good

ref.1 four of the five surviving workers have asbestos-related diseases, including three with recently diagnosed cancer.

0.52 four of the surviving five workers have asbestos-related diseases including three with recently diagnosed cancer.

(Score is BLEU approximation using rank order centroid weights)
Example: Bad

ref.2 although preliminary findings were reported more than a year ago, the latest results appear in today’s New England Journal of Medicine, a forum likely to bring new attention to the problem.

0.65 likely to bring new attention to the problem, today’s New England Journal of Medicine in a forum the latest results appear in although preliminary findings were reported more than a year ago.
Future Work: The Laundry List

1. Supertagger!
2. Google 1T 5-gram counts
3. Grammar improvements (stemming, agreement, punctuation)
4. PropBank integration
5. Perceptron tree models
6. Targeted human evaluations