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)

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)

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

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

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!

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

Why CCG?

- 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.)

$\Rightarrow$ 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)*

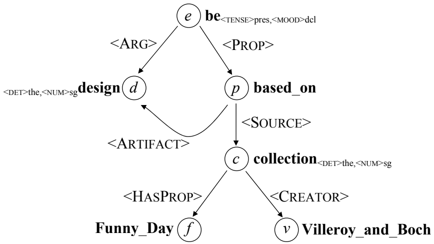
And this differs how ...

- Unlike Halogen (Langkilde-Geary, 2002) and FUF/SURGE (Callaway, 2003), OpenCCG uses a bidirectional grammar

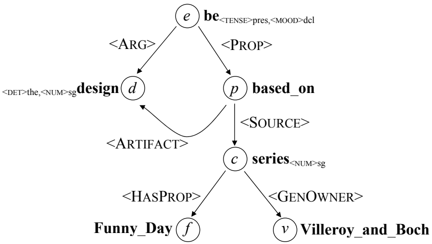
And this differs how . . .

- 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

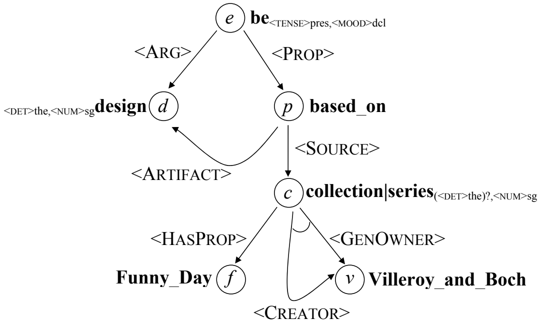


(a) *The design (is|’s) based on the Funny Day collection by Villeroy and Boch.*



(b) *The design (is|’s) based on Villeroy and Boch’s Funny Day series.*

Their Disjunctive Combination



(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)

$$
\begin{array}{l}
@_e(\textbf{be} \wedge \langle \textsc{tense} \rangle \text{pres} \wedge \langle \textsc{mood} \rangle \text{dcl} \wedge \\
\quad \langle \textsc{Arg} \rangle (d \wedge \textbf{design} \wedge \langle \textsc{det} \rangle \text{the} \wedge \langle \textsc{num} \rangle \text{sg}) \wedge \\
\quad \langle \textsc{Prop} \rangle (p \wedge \textbf{based\_on} \wedge \\
\qquad\qquad \langle \textsc{Artifact} \rangle d \wedge \\
\qquad\qquad \langle \textsc{Source} \rangle (c \wedge \textbf{collection} \wedge \langle \textsc{det} \rangle \text{the} \wedge \langle \textsc{num} \rangle \text{sg} \wedge \\
\qquad\qquad\qquad \langle \textsc{HasProp} \rangle (f \wedge \textbf{Funny\_Day}) \wedge \\
\qquad\qquad\qquad \langle \textsc{Creator} \rangle (v \wedge \textbf{V\&B}))))
\end{array}
$$

(a)

Disjunctive LF in HLDS

$$
\begin{array}{l}
@_e(\textbf{be} \wedge \langle \textsc{tense} \rangle \text{pres} \wedge \langle \textsc{mood} \rangle \text{dcl} \wedge \\
\quad \langle \textsc{Arg} \rangle (d \wedge \textbf{design} \wedge \langle \textsc{det} \rangle \text{the} \wedge \langle \textsc{num} \rangle \text{sg}) \wedge \\
\quad \langle \textsc{Prop} \rangle (p \wedge \textbf{based\_on} \wedge \\
\qquad \langle \textsc{Artifact} \rangle d \wedge \\
\qquad \langle \textsc{Source} \rangle (c \wedge \langle \textsc{num} \rangle \text{sg} \wedge (\langle \textsc{det} \rangle \text{the})? \wedge \\
\qquad\qquad (\textbf{collection} \underline{\vee} \textbf{series}) \wedge \\
\qquad\qquad \langle \textsc{HasProp} \rangle (f \wedge \textbf{Funny\_Day}) \wedge \\
\qquad\qquad (\langle \textsc{Creator} \rangle \boxed{v} \underline{\vee} \langle \textsc{GenOwner} \rangle \boxed{v})))) \\
\wedge\ @_v(\textbf{Villeroy\_and\_Boch})
\end{array}
$$

(c)

Flattening

(2) $0: @_e(\mathbf{be}),\ 1: @_e(\langle \textsc{tense}\rangle \text{pres}),$
$2: @_e(\langle \textsc{mood}\rangle \text{dcl}),\ 3: @_e(\langle \textsc{Arg}\rangle d),$
$4: @_d(\mathbf{design}),\ 5: @_d(\langle \textsc{det}\rangle \text{the}),$
$6: @_d(\langle \textsc{num}\rangle \text{sg}),$
$7: @_e(\langle \textsc{Prop}\rangle p),\ 8: @_p(\mathbf{based\_on}),$
$9: @_p(\langle \textsc{Artifact}\rangle d),\ 10: @_p(\langle \textsc{Source}\rangle c),$
$11: @_c(\langle \textsc{num}\rangle \text{sg}),\ 12: @_c(\langle \textsc{det}\rangle \text{the}),$
$13: @_c(\mathbf{collection}),\ 14: @_c(\mathbf{series}),$
$15: @_c(\langle \textsc{HasProp}\rangle f),\ 16: @_f(\mathbf{Funny\_Day}),$
$17: @_c(\langle \textsc{Creator}\rangle v),\ 18: @_c(\langle \textsc{GenOwner}\rangle v),$
$19: @_v(\mathbf{Villeroy\_and\_Boch})$

(3) $\mathsf{alt}_{0,0} = \{13\};\ \mathsf{alt}_{0,1} = \{14\}$
$\mathsf{alt}_{1,0} = \{17, 19\};\ \mathsf{alt}_{1,1} = \{18, 19\}$
$\mathsf{opt}_0 = \{12\}$

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. $\{11,13,14\}$ *collection* $\vdash \mathsf{n}_c$:
$@_c(\textbf{collection}) \wedge @_c(\langle \text{NUM} \rangle \text{sg})$

b. $\{11,13,14\}$ *series* $\vdash \mathsf{n}_c$:
$@_c(\textbf{series}) \wedge @_c(\langle \text{NUM} \rangle \text{sg})$

c. $\{17\}$ $\mathsf{alt}_{1,0}$ *by* $\vdash \mathsf{n}_c \backslash \mathsf{n}_c / \mathsf{np}_v$:
$@_c(\langle \text{CREATOR} \rangle v)$

d. $\{18\}$ $\mathsf{alt}_{1,1}$ *'s* $\vdash \mathsf{np}_c / \mathsf{n}_c \backslash \mathsf{np}_v$:
$@_c(\langle \text{GENOWNER} \rangle v)$

e. $\{19\}$ $\mathsf{alt}_{1,0}; \mathsf{alt}_{1,1}$ *Villeroy_and_Boch* $\vdash \mathsf{np}_v$:
$@_v(\textbf{V\&B})$

Derivation

1. $\{8\text{-}10\}$ *based_on* $\vdash$ $\mathsf{s}_p\backslash\mathsf{np}_d/\mathsf{np}_c$

2. $\{12\}$ *the* $\vdash$ $\mathsf{np}_c/\mathsf{n}_c$

3. $\{15, 16\}$ *Funny_Day* $\vdash$ $\mathsf{n}_c/\mathsf{n}_c$

4. $\{11, 13, 14\}$ *collection* $\vdash$ n_c
   $\{11, 13, 14\}$ *series* $\vdash$ n_c

5. $\{17\}\,\mathsf{alt}_{1,0}$ *by* $\vdash$ $\mathsf{n}_c\backslash\mathsf{n}_c/\mathsf{np}_v$

6. $\{18\}\,\mathsf{alt}_{1,1}$ *'s* $\vdash$ $\mathsf{np}_c/\mathsf{n}_c\backslash\mathsf{np}_v$

7. $\{19\}\,\mathsf{alt}_{1,0}; \mathsf{alt}_{1,1}$ *Villeroy_and_Boch* $\vdash$ np_v

Derivation (2)

8. $\{11, 13\text{-}16\}$ *FD [collection]* $\vdash$ n_c $(3\ 4 >)$
9. $\{17\text{-}19\}$ *by V&B* $\vdash$ $\mathsf{n}_c\backslash\mathsf{n}_c$ $(5\ 7 >)$
10. $\{17\text{-}19\}$ *V&B 's* $\vdash$ $\mathsf{np}_c/\mathsf{n}_c$ $(7\ 6 <)$
11. $\{11, 13\text{-}19\}$ *FD [coll.] by V&B* $\vdash$ n_c $(8\ 9 <)$
12. $\{11, 13\text{-}19\}$ *V&B 's FD [coll.]* $\vdash$ np_c $(10\ 8 >)$
13. $\{11\text{-}19\}$ *the FD [coll.] by V&B* $\vdash$ np_c $(2\ 11 >)$
    $\{11\text{-}19\}$ *V&B 's FD [coll.]* $\vdash$ np_c $(12\ \mathrm{optC})$
14. $\{8\text{-}19\}$ *b._on [the FD [coll.] . . .]* $\vdash$ $\mathsf{s}_p\backslash\mathsf{np}_d$ $(1\ 13 >)$

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, 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)

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

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

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) $\mathsf{s}_{\mathit{1:dcl}}\backslash\mathsf{np}_{\mathit{2}}/\mathsf{np}_{\mathit{3}} \Longrightarrow$
$\mathsf{s}_{\mathit{1:dcl,x1}}\backslash\mathsf{np}_{\mathit{2:x2}}/\mathsf{np}_{\mathit{3:x3}} \;:\; @_{x1}(\textbf{*pred*} \wedge \langle \textsc{Arg0} \rangle x2 \wedge \langle \textsc{Arg1} \rangle x3)$

(2) $\mathsf{s}_{\mathit{1:pss}}\backslash\mathsf{np}_{\mathit{2}} \Longrightarrow \mathsf{s}_{\mathit{1:pss,x1}}\backslash\mathsf{np}_{\mathit{2:x2}} \;:\; @_{x1}(\textbf{*pred*} \wedge \langle \textsc{Arg1} \rangle x2)$

(3) $\mathsf{np}_{\mathit{1}}/\mathsf{n}_{\mathit{1}} \Longrightarrow \mathsf{np}_{\mathit{1:x1}}/\mathsf{n}_{\mathit{1:x1}} \;:\; @_{x1}(\langle \textsc{Det} \rangle(d \wedge \textbf{*pred*}))$

(4) $\mathsf{np}_{\mathit{1}}/\mathsf{n}_{\mathit{1}}\backslash\mathsf{np}_{\mathit{2}} \Longrightarrow \mathsf{np}_{\mathit{1:x1}}/\mathsf{n}_{\mathit{1:x1}}\backslash\mathsf{np}_{\mathit{2:x2}} \;:\; @_{x1}(\langle \textsc{GenOwn} \rangle x2)$

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:

test set	LF created	single root
dev (00)	95.1% / 98.0%	67.4.% / 76.4%
test (23)	94.3% / 96.0%	69.7% / 77.2%

Coverage

Paper/Current:

test set	LF created	single root
dev (00)	95.1% / 98.0%	67.4.% / 76.4%
test (23)	94.3% / 96.0%	69.7% / 77.2%

- LFs with multiple roots have missing dependencies

Coverage

Paper/Current:

test set	LF created	single root
dev (00)	95.1% / 98.0%	67.4.% / 76.4%
test (23)	94.3% / 96.0%	69.7% / 77.2%

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

Exact regeneration

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

test set	grammar	complete	exact
wsj_0003	wsj_0003	86.7%	86.7%
	dev	76.7%	70.0%
	train	63.3%	56.7%
dev (00)	dev	59.1%	53.4%
	train	46.6%	37.7%

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

$$p(\vec{F}_1^n) \approx \prod_{i=1}^{n} p(\vec{F}_i \mid \vec{F}_{i-2}, \vec{F}_{i-1}) \tag{1}$$

$$\begin{aligned} p^W(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) &= p(W_i \mid W_{i-2}, W_{i-1}) \\ p^P(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) &= p(P_i \mid P_{i-2}, P_{i-1}) \\ p^S(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) &= p(S_i \mid P_{i-2}, P_{i-1}) \end{aligned} \tag{2}$$

Chained, Interpolated Models

$$p^{PS}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) = p(P_i \mid P_{i-2}^{i-1})p(S_i \mid P_{i-2}^{i}) \tag{3}$$

$$\begin{aligned} p^{W+P}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) &= \lambda_1 p^{W}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) + \\ &\quad \lambda_2 p^{P}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) \\ p^{W+PS}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) &= \lambda_1 p^{W}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) + \\ &\quad \lambda_2 p^{PS}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) \end{aligned} \tag{4}$$

Initial Non-Blind Development Results

scoring model	exact	BLEU
word 3g + pos 3g * stag 3g	14.8%	0.6615
word 3g + pos 3g	13.7%	0.6407
word 3g, interp. Kneser-Ney	12.2%	0.6247
word 3g, Good-Turing	11.7%	0.6219
pos 3g * supertag 3g	10.6%	0.6042
supertag 3g	10.0%	0.5886
pos 3g	8.0%	0.5413
null	5.1%	0.5251

Initial Results With Usual Splits

test set	scoring model	exact	BLEU
dev	w3g + pos3g * stag3g	8.1%	0.5578
	word 3g + pos 3g	7.1%	0.5210
	word 3g, Kneser-Ney	6.5%	0.4872
	null	2.2%	0.3697
test	w3g + pos3g * stag3g	9.8%	0.5768
	word 3g, Kneser-Ney	6.9%	0.5178

Updated Results With Best Model

Paper/Current:

test set	condition	exact	BLEU
dev	non-blind	14.8% / 24.3%	0.6615 / 0.7317
	usual	8.1% / 12.4%	0.5578 / 0.6101
test	usual	9.8% / 13.0%	0.5768 / 0.6223

BLEU Comparison (PTB 23)

(N.B.: direct comparison difficult!)

	coverage	BLEU
OpenCCG (07)	96.0%	0.6223
Cahill & van Genabith (06)	98.5%	0.6651
Langkilde-Geary (02), ‘Permute, no dir’	83%	0.757
Nakanishi et al. (05), $\leq$ 20w	90.8%	0.7733

Need a Supertagger for Realization!

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

Oracle/Best:

test set	grammar	complete
wsj_0003	wsj_0003	86.7% / 86.7%
	dev	76.7% / 76.7%
	train	63.3% / 10.0%
dev (00)	dev	59.1% / 57.0%
	train	46.6% / 21.3%

Continued Relevance of BLEU Scores?

⇒ Once realizations are generally satisfactory, BLEU scores may no longer be useful in measuring progress

Continued Relevance of BLEU Scores?

⇒ 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, punctation)
4. PropBank integration
5. Perceptron tree models
6. Targeted human evaluations