Towards Broad Coverage Surface Realization with CCG

Michael White and **Rajakrishnan Rajkumar** and **Scott Martin**
The Ohio State University
Department of Linguistics
Columbus, OH 43210 USA
{mwhite,raja,scott}@ling.ohio-state.edu

Abstract

This paper reports on progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (CCG). The paper provides initial automatic evaluation results which are roughly comparable to those reported with other formalisms when using a (non-blind) grammar derived from the development section of the CCGbank; the results are worse, though still respectable, when using the standard dev/train/test splits, highlighting the need for better lexical smoothing and more focused search. The paper also shows that factored language models that interpolate word-level n-grams with n-grams over POS tags and supertags provide similar absolute performance improvements over word-level n-grams as have been observed with parsing-inspired log-linear models.

1 Introduction

In this paper, we report on progress towards developing the first broad coverage English surface realizer for Combinatory Categorial Grammar (Steedman, 2000, CCG), using a grammar engineered from the CCGbank (Hockenmaier, 2003)—a corpus of CCG derivations created by transforming the Penn Treebank—together with the OpenCCG (White, 2006b; White, 2006a) realizer, enhanced for robustness. In previous work, OpenCCG has been used with precise, manually developed grammars for dialogue systems. By developing a broad coverage English grammar for OpenCCG, we aim to greatly reduce the effort required to make the realizer work in a new domain, and to enable open domain text-to-text applications. In principle, OpenCCG could also be used to investigate methods of assembling NLP component technologies into a high quality MT system based on semantic transfer, as in (Lønning and Oepen, 2006); since OpenCCG supports disjunctive logical forms as input, such a system could employ transfer rules on packed representations, along the lines of (Emele and Dorna, 1998), rather than making hard disambiguation choices during parsing and transfer.

Our efforts to engineer a grammar from the CCGbank suitable for realization with OpenCCG involve adding semantic representations to the lexical categories and, where feasible, converting the corpus to reflect more precise analyses. While we are finding this process to be a time-consuming and nontrivial one, we expect our efforts to yield a linguistically informed and moderately precise, broad coverage English grammar—suitable for both parsing and realization—in much less time than it would take to scale up a manually written grammar to cover all the phenomena in the Penn Treebank. Our approach may be contrasted with those of (Langkilde-Geary, 2002; Callaway, 2003), who developed converters for the outputs of Treebank parsers to produce inputs for their realizers, rather than pursue parsers and realizers that share a bidirectional grammar. Instead, our approach is more similar to the ones pursued by (Carroll and Oepen, 2005; Nakanishi et al., 2005; Cahill and van Genabith, 2006) with HPSG and LFG grammars, except for our greater focus on engineering a broad coverage grammar where the logical forms more closely resemble those used traditionally in generation.

The rest of the paper is organized as follows. In Section 2, we give an overview of surface realization with OpenCCG. In Section 3, we describe our approach to engineering a grammar from the CCGbank suitable for realization with OpenCCG. In Section 4, we describe our scoring methods, provide initial automatic evaluation results, and discuss concerns about relying too heavily on automatic metrics for judging realization quality. In this section, we also contribute to the debate that has arisen recently over the relative merits of language models and syntactic features in realization ranking, by showing that factored language models that interpolate word-level n-grams with n-grams over part-of-speech tags and supertags provide similar absolute performance improvements over word-level n-grams as have been observed with log-linear models using parsing-inspired features, as in (Velldal and Oepen, 2005). Finally, in Section 5 we conclude with a summary and discussion of future work.

2 Surface Realization with OpenCCG

The OpenCCG open source surface realizer is based on Steedman's (2000) version of CCG elaborated with Baldridge and Kruijff's multi-modal extensions for lexically specified derivation control (Baldridge, 2002; Baldridge and Kruijff, 2003) and hybrid logic dependency semantics (Baldridge and Kruijff, 2002). OpenCCG implements a hybrid symbolic-statistical chart realization algorithm (Kay, 1996; Carroll et al., 1999; White, 2006b) combining (1) a theoretically grounded approach to syntax and semantic composition with (2) factored language models (Bilmes and Kirchhoff, 2003) for making choices among the options left open by the grammar.

In OpenCCG, the search for complete realizations makes use of n-gram language models over words represented as vectors of factors, including surface form, part of speech, supertag and semantic class. The search proceeds in one of two modes, *anytime* or *two-stage* (packing/unpacking). In the anytime mode, a best-first search is performed with a configurable time limit: the scores assigned by the n-gram model determine the order of the edges on the agenda, and thus have an impact on realization speed. In the two-stage mode, a packed forest of all possible realizations is created in the first stage; in the second stage, the packed representation is unpacked in bottom-up fashion, with scores assigned to the edge for each sign as it is unpacked, much as in (Langkilde, 2000). Edges are grouped into equivalence classes when they have the same syntactic category and cover the same parts of the input logical form. At present, pruning is only done within equivalence classes of edges, and all lexical category assignments are considered for each input predicate, to ensure that pruning does not prevent a complete realization from being found.

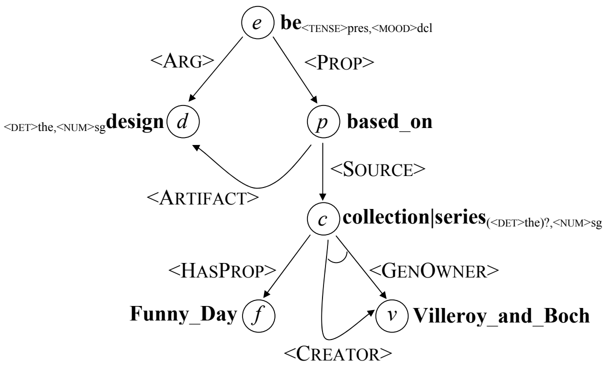

Figure 1: Simplified disjunctive semantic dependency graph from the COMIC dialogue system, for the eight paraphrases *The design (is|'s) based on (the Funny Day (collection|series) by Villeroy and Boch | Villeroy and Boch's Funny Day (collection|series)).*

To realize a wide range of paraphrases, OpenCCG implements an algorithm for efficiently generating from disjunctive logical forms (White, 2006a). This capability has many benefits, such as enabling the selection of realizations according to predicted synthesis quality (Nakatsu and White, 2006), and avoiding repetition in the output of a dialogue system (Foster and White, 2007). The disjunctive logical forms describe semantic dependency graphs with alternative or optional elements, as illustrated in Figure 1; see (White, 2006a) for further explanation.

To better support broad coverage surface realization, we have enhanced OpenCCG for robustness in several ways. The most significant extension implemented so far has been a technique to improve robustness in the event that the realizer fails to find a complete realization, in which case fragments are greedily assembled to cover as much of the input semantics as possible. The algorithm begins with the edge for the best partial realization, i.e. the one that covers the most elementary predications in the input logical form, with ties broken according to the n-gram score. (Larger fragments are preferred under the assumption that they are more likely to be grammatical.) Next, the chart and agenda are greedily searched for the best edge whose semantic coverage is disjoint from those selected so far; this process repeats until no further edges can be added to the set of selected fragments. In the final step, these fragments are concatenated, again in a greedy fashion, this time according to the n-gram score of the concatenated edges: starting with the original best edge, the fragment whose concatenation on the left or right side yields the highest score is chosen as the one to concatenate next, until all the fragments have been concatenated into a single output.

3 Engineering a Grammar for Realization with OpenCCG from the CCGbank

Our process for deriving a grammar suitable for OpenCCG realization from the CCGbank proceeds in two steps. In the first step, the derivations in the CCGbank are revised to reflect the desired syntactic derivations. Changes to the derivations are necessary to reflect the lexicalized treatment of coordination and punctuation assumed by the multi-modal version of CCG that is implemented in OpenCCG. Further changes are necessary to support semantic dependencies rather than surface syntactic ones; in particular, the features and unification constraints in the categories related to semantically empty function words such complementizers, infinitival-*to*, expletive subjects, and case-marking prepositions are adjusted to reflect their purely syntactic status. This step is implemented in a general fashion as a series of XSLT transformations with external Java function calls, following an initial conversion of CCGbank files into an XML representation.

In the second step, a grammar is extracted from the converted CCGbank and augmented with logical forms. Categories and unary type changing rules (corresponding to zero morphemes) are sorted by frequency and extracted if they meet the specified frequency thresholds. Categories for function words that have no semantics or introduce only semantic features or relations are marked in an XSLT transformation. A separate transformation then uses around two dozen generalized templates to add logical forms to the categories. The effect of this transformation is illustrated below. Examples (1) and (2) show how numbered semantic roles, as in PropBank (Palmer et al., 2005), are added to the active and passive categories for a transitive verb, where ***pred*** is a placeholder for the lexical predicate; examples (3) and (4) show how more specific relations are introduced in the category for determiners and the category for the possessive *'s*, respectively.

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

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

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

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

After logical form insertion, the extracted and augmented grammar is loaded and used to parse the sentences in the CCGbank according to the gold-standard derivation. If the derivation can be successfully followed, the parse yields a logical form which is saved along with the corpus sentence in order to later test the realizer. The algorithm for constrain-parsing the corpus sentences attempts to continue processing if it encounters a blocked derivation due to sentence-interal punctuation. While we are currently reanalyzing the punctuation categories along the lines of Doran's (1998) TAG-based analysis, many problem cases remain due to the CCGbank's reliance on punctuation-specific binary rules that are not supported in OpenCCG.

Currently, the algorithm succeeds in creating logical forms for 1824, or 95.1%, of the 1917 sentences in the development section (Sect. 00) of the converted CCGbank, and 2182, or 94.3%, of the 2314 sentences in the test section (Sect. 23). Of these, 1293, or 67.4.%, of the development logical forms are semantic dependency graphs with a single root, while 1614, or 69.7%, of the test logical forms have a single root. The remaining cases, with multiple roots, are missing one or more dependencies required to form a fully connected graph. While these missing dependencies usually reflect inadequacies in the logical form templates, we have found several categories that require dependencies beyond those specified in the original CCGbank.

4 Evaluation

In this section, we report initial results on the standard development and test sections (00 and 23, resp.) of the converted CCGbank. We report both non-blind and blind test results, where the non-blind results use a grammar derived from the development section, as the comparison is informative. Note that with both sets of results, n-gram probabilities are not estimated from the test data.

In deriving the grammar, we filtered out any lexical categories that appeared fewer than k_l times, and any unary type changing rules that appeared fewer than k_r times. These cutoff thresholds were determined using informal experiments on the development data; for the grammar derived from the development section, we used $k_l = 3$ and $k_r = 5$, while for the grammar derived from the test sections, we used $k_l = 10$ and $k_r = 10$. We also excluded any lexical categories or unary rules that were not matched by the semantic templates.

To handle out-of-vocabulary (OOV) words in the test sections, we have implemented a basic approach to lexical smoothing, as follows. For each OOV noun, proper name, number or adjective, we added the word to the lexicon with all categories for their respective parts of speech that occurred at least 50 times in the training sections. For each OOV verb, we added it to the lexicon with the 5 most frequent categories for all verbs; likewise, we added each OOV adverb with the 3 most frequent categories for all adverbs.

We ran the realizer in its best-first anytime search mode, with an overall time limit of 15 seconds and an n-best pruning value (i.e., beam width) of 5. When a realization that completely covered the input semantics could not be found within the time limit, the largest compatible fragments were greedily assembled as described in Section 2, so that some output was produced for 100% of the inputs.

4.1 Scoring Methods

To score partial and complete realizations, we experimented with a variety of factored trigram models over words, part-of-speech tags and supertags. As a baseline, we used a null scorer, which assigns a score of zero to all realizations. With this scorer, an arbitrary choice is made among complete realizations; with fragments, the largest compatible fragments are concatenated in an arbitrary order.

The language models were created using the SRILM toolkit (Stolcke, 2002) on the standard training sections (2–21) of the CCGbank, with sentence-initial words (other than proper names) uncapitalized. While these models are considerably smaller than the ones used in (Langkilde-Geary, 2002; Velldal and Oepen, 2005), the training data does have the advantage of being in the same domain and genre. For the word-based models, perplexity measures were much lower for trigram models than bigram models, but 4-gram models showed no improvement. In the end, we experimented with two word-based models, a trigram model using Good-Turing smoothing (SRILM's default method), and one using interpolated Kneser-Ney smoothing. Both models used the default frequency cutoffs.

In addition to the usual word-based trigram models, we also created trigram models over part-of-speech tags and supertags (using Kneser-Ney smoothing). In the latter model, the probability of the current word's supertag (category label) is conditioned on the previous two part-of-speech tags, inspired by work on CCG supertagging (Clark and Curran, 2004; Curran et al., 2006). This is illustrated in (1)-(2). Equation (1) shows how the chain rule and the Markov assumption is used to approximate the probability of a sentence or phrase consisting of a sequence of factor vectors $\vec{F}_i$, for words 1 to n. In (2), $p^W(\vec{F}_i \mid \vec{F}_{i-2}^{i-1})$, $p^P(\vec{F}_i \mid \vec{F}_{i-2}^{i-1})$ and $p^S(\vec{F}_i \mid \vec{F}_{i-2}^{i-1})$ are defined to be approximations of the conditional probability $p(\vec{F}_i \mid \vec{F}_{i-2}^{i-1})$ that pay attention only to the word (W), part-of-speech (P) and supertag (S) factors of $\vec{F}$, as indicated.

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

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

Table 1: Non-blind results on development data (CCGbank Section 00), using a grammar extracted from the same section. Factored trigram models over words, part-of-speech (pos) tags and supertags improve BLEU scores and exact matches, with the interpolated supertag model showing a substantial increase over word-only trigram models.

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

Additionally, we created a model that chains the part-of-speech trigram model with a supertag model that uses the current part-of-speech tag as an additional parent upon which to condition the probability of the supertag (S), as shown in (3).

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

Finally, we built two additional models which linearly interpolate the interpolated Kneser-Ney word model with the part-of-speech model and the combined part-of-speech and supertag model, as shown in (4). Assuming that the word model was the more informative one, we set the interpolation weights using the rank order centroid formula, which in this case yields 0.75 for λ_1 and 0.25 for λ_2.

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

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

Table 2: Initial results on development (Section 00) and test (Section 23) data, using a grammar extracted from the training sections (Sections 02-21) and a smoothed lexicon (see text). Results show the same pattern as in Table 1.

4.2 Results

Tables 1 and 2 show initial results for the different scoring methods, using grammars extracted from the development section (non-blind) and training sections (blind), respectively. Results are given in terms of percentage of exact matches and BLEU (Papineni et al., 2001) n-gram precision scores using the corpus sentence as a reference. While realizations that exactly match the corpus sentence can be presumed to be of high quality, this metric is overly harsh in that it does not recognize acceptable variants. BLEU scores have been shown to correlate with human judgments of adequacy and fluency, but must be interpreted with caution, as discussed below.

As the tables indicate, the best performing models interpolate the word-level trigram model with the chained part-of-speech and supertag trigram model. In Table 1, this model scores nearly four BLEU points[1] better than the word-level trigram model (using interpolated Kneser-Ney smoothing) by itself, and two points better than the word-level model interpolated with just the part-of-speech trigram model. By comparison, the impact of using interpolated Kneser-Ney smoothing instead of Good-Turing smoothing is small, less than one third of a BLEU point. As the next line shows, the word-level trigram models both perform better than the chained part-of-speech and supertag trigram model by itself; this model in turn performs better than the plain supertag trigram model, and considerably better than the plain part-of-speech trigram model, whose performance is not that much better than the null scorer

[1] 1 BLEU point = 0.01 change in BLEU score.

ref.1	Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 .
ref.2	Mr. Vinken is chairman of Elsevier N.V. , the Dutch publishing group .
top.1	61 years old Pierre Vinken will join the board as a nonexecutive director Nov. 29 .
top.2	Mr. Vinken is chairman of Elsevier N.V. , the publishing Dutch group .
word.1	61 years old will join the board as a nonexecutive director Nov. 29 . Pierre Vinken
word.2	Mr. Vinken is in chairman of N.V. Elsevier , the Dutch publishing group .
null.1	will join Nov. 29 the board as a nonexecutive director 61 years old Pierre Vinken .
null.2	Mr. Vinken is chairman of Elsevier N.V. , the publishing Dutch group .

Table 3: Sample outputs from the first two sentences in the Penn Treebank, for the top scoring model, the best word-only n-gram model, and the null scoring (arbitrary choice) model from Table 1. The first sentence requires joining fragments.

baseline. Sample outputs using the non-blind grammar appear in Table 3; these sentences illustrate how the top-scoring model often yields decent (but certainly not perfect) realizations that are generally better than ones produced with the baseline models.

Table 2 shows the same pattern as Table 1, with an even greater boost provided by the top scoring model over the word-level model, though with considerably lower scores overall. To investigate the reasons for the drop-off, we compared the percentage of complete realizations (versus fragmentary ones), as well as the percentage of test items for which realization finished within the time limit. With the grammar derived from the development section, the realizer was able to find complete realizations for 55% of the development sentences with logical forms (with the remaining 45% realized as concatenated fragments), and with the search exceeding the time limit in less than 1% of the cases. In contrast, with the grammar derived from the training sections, complete realizations were found for only 22% of these development sentences, with the search exceeding the time limit 68% of the time. These differences suggested that the drop-off in scores was caused not only by sparse data and an overly simple lexical smoothing strategy, but also by search errors arising with the larger grammar derived from the training sections.

To confirm that search errors had become a significant issue, we compared the percentage of complete realizations with the top scoring model against an oracle model that uses a simplified BLEU score based on the target string, which we have found useful for regression testing in previous work. We made the comparison on the first normal-sized file in the development section (`wsj_0003`), using both the grammar extracted from the training sections and a grammar extracted just from this file, with no frequency cutoffs. Using the latter, file-specific grammar, both the oracle and top scoring models found complete realizations for 73% of the test sentences. (With a perfect grammar extraction process, we would expect 100% complete realizations.) By contrast, with the grammar derived from the training sections, the percentage of complete realizations using the oracle model dropped to 50%, while the percentage using the top model fell to 13%, indicating that search errors occurred frequently.

4.3 Comparison to Previous Work

It is perhaps not surprising that interpolating the chained part-of-speech and supertag trigram model with the word-level model boosts performance, given that n-grams of supertags are not unlike the n-grams of lexical types used as features in Velldal & Oepen's (2005) maximum entropy model. What is remarkable though is that the increase in BLEU points is comparable to the improvement over an n-gram model achieved by Velldal & Oepen's complete model, which additionally contains features based on the syntactic derivation trees. Since Velldal & Oepen do not report scores for a model using just word-level n-grams and supertag-like n-grams, one cannot tell whether and to what extent the derivational features provide complementary information. Of course, additional factors complicate the comparison, in particular that Velldal & Oepen use a much smaller corpus of 864 sentence and logical form pairs with a much larger language model trained on the BNC, which however differs in domain and genre from their corpus. It is also worth keeping in mind that Nakanishi et al.'s (2005) log-linear syntactic model outperformed an n-gram model on the Penn Treebank, and showed no improvement (in fact, worsened) when the n-gram score was added as a feature. Again, however, their n-gram model was based on the BNC, rather than an in-domain and in-genre corpus.

Looking beyond realization ranking, Birch et al. (2007) have recently shown that n-gram models over CCG supertags can be combined with word-level language models to yield improved word orderings in phrase-based statistical machine translation. They also show that CCG sequence models work better than part-of-speech sequence models, but do not investigate chaining a part-of-speech model with a supertag model. Note that in their factored, phrase-based statistical MT approach, the CCG supertags do not actually play a role in syntactic derivations. Given that in our approach, the CCG categories must actually combine into a derivation (except where fragments are employed), it is interesting to observe that the supertag models still provide a considerable boost in determining preferred word orders.

Stepping back from the effect of factored language models, we observe that it is not necessarily very meaningful to compare overall results on realization ranking with the Penn Treebank, since as Langkilde-Geary (2002) has shown, results can vary widely depending on how specific the inputs to realization are. Moreover, Callison Burch et al. (2006) have shown that with machine translation outputs (which are generally much worse than realizer outputs), improved BLEU scores are neither necessary nor sufficient to achieve better human evaluation scores, and thus they caution against relying on BLEU scores when comparing systems that employ substantially different methods. Stent et al. (2005) also point out that BLEU and other automatic metrics do not take discourse context into account, and suggest that these metrics are poor judges of fluency with generators that aim to produce desirable variation. Finally, as Callaway (2003) points out, it is unclear how to compare systems whose coverage varies considerably.

With these caveats in mind, we may observe that our non-blind results of 0.6615 BLEU score at 94.5% coverage (including failures in logical form creation) are not too far off those of (Cahill and van Genabith, 2006), who report BLEU scores of 0.6651 on all inputs in PTB Section 23 at 98.5% coverage, and 0.6979 at 89.5% coverage. Meanwhile, we note that their LFG f-structure inputs contain somewhat more precise specifications than our logical forms, including surface syntactic roles, features for certain function words, and marking fronted constituents as topics. Of course, our BLEU score of 0.5768 at 94.3% coverage on Section 23, using the standard dev/train/test splits, is considerably worse, though still in the respectable range.

Compared to (Nakanishi et al., 2005), our numbers appear considerably lower than their BLEU score of 0.7733 at 90.8% coverage, however their results only include sentences up to length 20. In (Langkilde-Geary, 2002), a BLEU score of 0.514 is reported for minimally specified inputs, while a score of 0.757 is reported for the 'Permute, no dir' case (which perhaps most closely resembles our inputs), and a score of 0.924 is reported for the most fully specified inputs. However, coverage is less than 83%, and a generation-only (i.e., non-reversible) set of rewrite rules are used in the symbolic half of her system. Finally, in (Callaway, 2003), string accuracy results comparable to those of (Langkilde-Geary, 2002) are reported at 98.7% coverage, using the older FUF/SURGE symbolic realizer; again, however, this approach uses a generation-only grammar, cannot vary its output according to different statistical scoring methods, and is not designed to produce n-best outputs.

4.4 Discussion

As we refine our corpus-based grammar engineering process, extending coverage and producing fewer fragmentary outputs, we expect realization quality to improve. Additionally, we have begun to investigate using a realization analogue of CCG supertagging, where lexical categories are predicted based on the context of predicates in the input semantic dependency graph, in a fashion reminiscent of the LTAG tree models of (Bangalore and Rambow, 2000). We anticipate that a supertagger for realization will more accurately handle unseen words, and reduce search errors by focusing the search space on the most likely lexical categories.

Once we reach the point where realizations are generally satisfactory, we suspect that BLEU scores

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 .
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 .

Table 4: Example outputs from PTB file `wsj_0003` for the top scoring model in Table 2, where the simplified BLEU scores do not mirror relative quality.

may no longer be useful in measuring progress, given the concerns raised by (Callison-Burch et al., 2006) and (Stent et al., 2005) discussed earlier, and especially our goal of producing desirable variation. As such, we expect that targeted human evaluations will become essential. In support of this view, we note that it is easy to find pairs of sentences whose n-gram precision scores are both fairly high where the difference in their scores does not mirror relative quality. For example, Table 4 shows two sentences from PTB file `wsj_0003`, along with the realizations generated using our top scoring model and the grammar derived from the training sections. The scores listed are calculated using a simplified BLEU metric where 1- to 4-gram precision scores are combined via rank order centroid weights, instead of the geometric mean, to make the metric suitable for single sentences. With the first sentence, the realizer output is quite good, if slightly less fluent than the original. With the second sentence, despite its considerably higher score, the realizer output is difficult (if not impossible) to interpret.

5 Summary and Future Work

In this paper, we have described our approach to developing the first broad coverage English surface realizer for CCG, using a grammar derived from the CCGbank together with a robustness-enhanced version of the OpenCCG realizer. We have also provided initial automatic evaluation results, where we obtained BLEU scores that are roughly comparable to those reported with other formalisms when using a (non-blind) grammar derived from the development section, and scores in the respectable range using the standard dev/train/test splits. In our evaluation, we have shown that factored language models that interpolate word-level n-grams with n-grams over part-of-speech tags and supertags can provide similar absolute performance improvements over word-level n-grams as have been observed with log-linear models using parsing-inspired features. We also found that better lexical smoothing and more focused search using a subset of initial lexical category assignments are necessary for further progress.

We are currently working on incorporating a more refined analysis of punctuation into our grammar conversion process, and intend to report on the resulting improvements in performance at the workshop. We are also investigating language models using the 5-gram counts that Google has made available,[2] generated from 1 trillion tokens of text from web pages. The SRILM toolkit now has count-based language models that can read this data, but they require inordinate amounts of RAM. We have experimented with filtering the data using the vocabulary from the CCGbank, which appears promising.

In future work, we intend to refine our corpus-based grammar engineering process by stemming open class words to serve as semantic predicates, introducing features for tense and number, implementing subject-verb agreement, and enhancing the treatment of semantic roles by integrating PropBank roles more directly. We also intend to develop a supertagger for generation to improve lexical smoothing and speed up processing, investigate the impact of features used in log-linear parsing models on realization quality in combination with rich language models and supertag models, and perform targeted human evaluations of sentences in context to better judge resulting improvements.

Acknowledgements

We thank Julia Hockenmaier for providing a version of the CCGbank with the heads of PPs marked in the categories. We also thank the anonymous reviewers

[2]`http://www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt`

for helpful comments and suggestions.

References

Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. ACL-02*.

Jason Baldridge and Geert-Jan Kruijff. 2003. Multi-Modal Combinatory Categorial Grammar. In *Proc. ACL-03*.

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proce. COLING-00*.

Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and general parallelized backoff. In *Proc. HLT-03*.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proc. SMT Workshop at ACL-07*.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proc. COLING-ACL '06*.

Charles Callaway. 2003. Evaluating coverage for large symbolic NLG grammars. In *Proc. IJCAI-03*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proc. EACL-06*.

John Carroll and Stefan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proc. IJCNLP-05*.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznański. 1999. An efficient chart generator for (semi-) lexicalist grammars. In *Proc. ENLG-99*.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proc. COLING-04*.

James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proc. COLING-ACL '06*.

Christine Doran. 1998. *Incorporating Punctuation Into The Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective*. Ph.D. thesis, University of Pennsylvania. Technical Report IRCS-98-24.

Martin C. Emele and Michael Dorna. 1998. Ambiguity preserving machine translation using packed representations. In *Proc. of COLING-ACL '98*.

Mary Ellen Foster and Michael White. 2007. Avoiding repetition in generated text. In *Proc. ENLG-07*.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Martin Kay. 1996. Chart generation. In *Proc. ACL-96*.

Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proc. INLG-02*.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proc. NAACL-00*.

Jan Tore Lønning and Stephan Oepen. 2006. Re-usable tools for precision machine translation. In *Proc. COLING-ACL '06 Demo Sessions*.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic methods for disambiguation of an HPSG-based chart generator. In *Proc. IWPT-05*.

Crystal Nakatsu and Michael White. 2006. Learning to say it well: Reranking realizations by predicted synthesis quality. In *Proc. COLING-ACL '06*.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176, IBM.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Proc. CICLing-05*.

Andreas Stolcke. 2002. SRILM — An extensible language modeling toolkit. In *Proc. ICSLP-02*.

Erik Velldal and Stephan Oepen. 2005. Maximum entropy models for realization ranking. In *Proc. MT Summit X*.

Michael White. 2006a. CCG chart realization from disjunctive inputs. In *Proceedings, INLG 2006*.

Michael White. 2006b. Efficient realization of coordinate structures in Combinatory Categorial Grammar. *Research on Language and Computation*, 4(1):39–75.