That's Not What I Meant! Using Parsers to Avoid Structural Ambiguities in Generated Text

Manjuan Duan and **Michael White**
Department of Linguistics
The Ohio State University
Columbus, OH 43210, USA
{duan,mwhite}@ling.osu.edu

Abstract

We investigate whether parsers can be used for self-monitoring in surface realization in order to avoid egregious errors involving "vicious" ambiguities, namely those where the intended interpretation fails to be considerably more likely than alternative ones. Using parse accuracy in a simple reranking strategy for self-monitoring, we find that with a state-of-the-art averaged perceptron realization ranking model, BLEU scores cannot be improved with any of the well-known Treebank parsers we tested, since these parsers too often make errors that human readers would be unlikely to make. However, by using an SVM ranker to combine the realizer's model score together with features from multiple parsers, including ones designed to make the ranker more robust to parsing mistakes, we show that significant increases in BLEU scores can be achieved. Moreover, via a targeted manual analysis, we demonstrate that the SVM reranker frequently manages to avoid vicious ambiguities, while its ranking errors tend to affect fluency much more often than adequacy.

1 Introduction

Rajkumar & White (2011; 2012) have recently shown that some rather egregious surface realization errors—in the sense that the reader would likely end up with the wrong interpretation—can be avoided by making use of features inspired by psycholinguistics research together with an otherwise state-of-the-art averaged perceptron realization ranking model (White and Rajkumar, 2009), as reviewed in the next section. However, one is apt to wonder: could one use a parser to check whether the intended interpretation is easy to recover, either as an alternative or to catch additional mistakes? Doing so would be tantamount to self-monitoring in Levelt's (1989) model of language production.

Neumann & van Noord (1992) pursued the idea of self-monitoring for generation in early work with reversible grammars. As Neumann & van Noord observed, a simple, brute-force way to generate unambiguous sentences is to enumerate possible realizations of an input logical form, then to parse each realization to see how many interpretations it has, keeping only those that have a single reading; they then went on to devise a more efficient method of using self-monitoring to avoid generating ambiguous sentences, targeted to the ambiguous portion of the output. We might question, however, whether it is really possible to avoid ambiguity entirely in the general case, since Abney (1996) and others have argued that nearly every sentence is potentially ambiguous, though we (as human comprehenders) may not notice the ambiguities if they are unlikely. Taking up this issue, Khan et al. (2008)—building on Chantree et al.'s (2006) approach to identifying "innocuous" ambiguities—conducted several experiments to test whether ambiguity could be balanced against length or fluency in the context of generating referring expressions involving coordinate structures. Though Khan et al.'s study was limited to this one kind of structural ambiguity, they do observe that generating the brief variants when the intended interpretation is clear instantiates Van Deemter's (2004) general strategy of only avoiding **vicious ambiguities**—that is, ambiguities where the intended interpretation fails to be considerably more likely than any other distractor interpretations—rather than trying to avoid all ambiguities.

In this paper, we investigate whether Neumann & van Noord's brute-force strategy for avoiding ambiguities in surface realization can be updated to only avoid vicious ambiguities, extending (and revising) Van Deemter's general strategy to all kinds of structural ambiguity, not just the one investigated by Khan et al. To do so—in a nutshell—we enumerate an n-best list of realizations and rerank them if necessary to avoid vicious ambiguities, as determined by one or more automatic parsers. A potential obstacle, of course, is that automatic parsers may not be sufficiently representative of human readers, insofar as errors that a parser makes may not be problematic for human comprehension; moreover, parsers are rarely successful in fully recovering the intended interpretation for sentences of moderate length, even with carefully edited news text. Consequently, we examine two reranking strategies, one a simple baseline approach and the other using an SVM reranker (Joachims, 2002).

Our simple reranking strategy for self-monitoring is to rerank the realizer's n-best list by parse accuracy, preserving the original order in case of ties. In this way, if there is a realization in the n-best list that can be parsed more accurately than the top-ranked realization—even if the intended interpretation cannot be recovered with 100% accuracy—it will become the preferred output of the combined realization-with-self-monitoring system. With this simple reranking strategy and each of three different Treebank parsers, we find that it is possible to improve BLEU scores on Penn Treebank development data with White & Rajkumar's (2011; 2012) baseline generative model, but not with their averaged perceptron model. In inspecting the results of reranking with this strategy, we observe that while it does sometimes succeed in avoiding egregious errors involving vicious ambiguities, common parsing mistakes such as PP-attachment errors lead to unnecessarily sacrificing conciseness or fluency in order to avoid ambiguities that would be easily tolerated by human readers. Therefore, to develop a more nuanced self-monitoring reranker that is more robust to such parsing mistakes, we trained an SVM using dependency precision and recall features for all three parses, their n-best parsing results, and per-label precision and recall for each type of dependency, together with the realizer's normalized perceptron model score as a feature. With the SVM reranker, we obtain a significant improvement in BLEU scores over White & Rajkumar's averaged perceptron model on both development and test data. Additionally, in a targeted manual analysis, we find that in cases where the SVM reranker improves the BLEU score, improvements to fluency and adequacy are roughly balanced, while in cases where the BLEU score goes down, it is mostly fluency that is made worse (with reranking yielding an acceptable paraphrase roughly one third of the time in both cases).

The paper is structured as follows. In Section 2, we review the realization ranking models that serve as a starting point for the paper. In Section 3, we report on our experiments with the simple reranking strategy, including a discussion of the ways in which this method typically fails. In Section 4, we describe how we trained an SVM reranker and report our results using BLEU scores (Papineni et al., 2002). In Section 5, we present a targeted manual analysis of the development set sentences with the greatest change in BLEU scores, discussing both successes and errors. In Section 6, we briefly review related work on broad coverage surface realization. Finally, in Section 7, we sum up and discuss opportunities for future work in this direction.

2 Background

We use the OpenCCG[1] surface realizer for the experiments reported in this paper. The OpenCCG realizer generates surface strings for input semantic dependency graphs (or logical forms) using a chart-based algorithm (White, 2006) for Combinatory Categorial Grammar (Steedman, 2000) together with a "hypertagger" for probabilistically assigning lexical categories to lexical predicates in the input (Espinosa et al., 2008). An example input appears in Figure 1. In the figure, nodes correspond to discourse referents labeled with lexical predicates, and dependency relations between nodes encode argument structure (gold standard CCG lexical categories are also shown); note that semantically empty function words such as infinitival-*to* are missing. The grammar is extracted from a version of the CCGbank (Hockenmaier and Steedman, 2007) enhanced for realization; the enhancements include: better analyses of punctuation (White and Rajkumar, 2008); less error prone handling of named entities (Rajkumar et al., 2009); re-inserting quotes into the CCGbank;

[1]`http://openccg.sf.net`

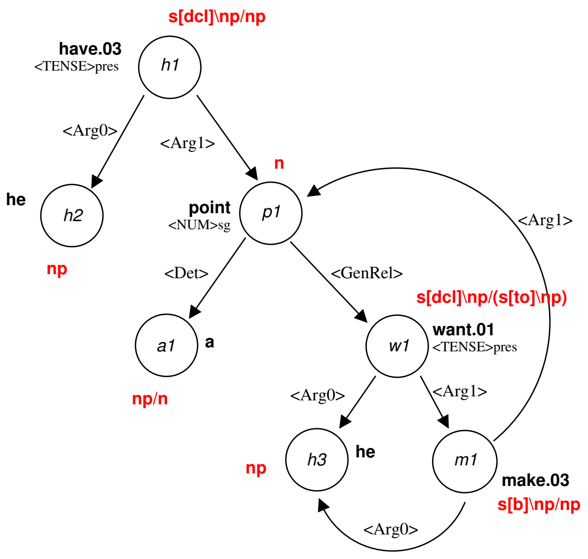


Figure 1: Example OpenCCG semantic dependency input for *he has a point he wants to make*, with gold standard lexical categories for each node

and assignment of consistent semantic roles across diathesis alternations (Boxwell and White, 2008), using PropBank (Palmer et al., 2005).

To select preferred outputs from the chart, we use White & Rajkumar's (2009; 2012) realization ranking model, recently augmented with a large-scale 5-gram model based on the Gigaword corpus. The ranking model makes choices addressing all three interrelated sub-tasks traditionally considered part of the surface realization task in natural language generation research (Reiter and Dale, 2000; Reiter, 2010): inflecting lemmas with grammatical word forms, inserting function words and linearizing the words in a grammatical and natural order. The model takes as its starting point two probabilistic models of syntax that have been developed for CCG parsing, Hockenmaier & Steedman's (2002) generative model and Clark & Curran's (2007) normal-form model. Using the averaged perceptron algorithm (Collins, 2002), White & Rajkumar (2009) trained a structured prediction ranking model to combine these existing syntactic models with several n-gram language models. This model improved upon the state-of-the-art in terms of automatic evaluation scores on held-out test data, but nevertheless an error analysis revealed a surprising number of word order, function word and inflection errors. For each kind of error, subsequent work investigated the utility of employing more linguistically motivated features to improve the ranking model.

To improve word ordering decisions, White & Rajkumar (2012) demonstrated that incorporating a feature into the ranker inspired by Gibson's (2000) dependency locality theory can deliver statistically significant improvements in automatic evaluation scores, better match the distributional characteristics of sentence orderings, and significantly reduce the number of serious ordering errors (some involving vicious ambiguities) as confirmed by a targeted human evaluation. Supporting Gibson's theory, comprehension and corpus studies have found that the tendency to **minimize dependency length** has a strong influence on constituent ordering choices; see Temperley (2007) and Gildea and Temperley (2010) for an overview.

Table 1 shows examples from White and Rajkumar (2012) of how the dependency length feature (DEPLEN) affects the OpenCCG realizer's output even in comparison to a model (DEPORD) with a rich set of discriminative syntactic and dependency ordering features, but no features directly targeting relative weight. In wsj_0015.7, the dependency length model produces an exact match, while the DEPORD model fails to shift the short temporal adverbial *next year* next to the verb, leaving a confusingly repetitive *this year next year* at the end of the sentence. Note how shifting *next year* from its canonical VP-final position to appear next to the verb shortens its dependency length considerably, while barely lengthening the dependency to *based on*; at the same time, it avoids ambiguity in what *next year* is modifying. In wsj_0020.1 we see the reverse case: the dependency length model produces a nearly exact match with just an equally acceptable inversion of *closely watching*, keeping the direct object in its canonical position. By contrast, the DEPORD model mistakenly shifts the direct object *South Korea, Taiwan and Saudia Arabia* to the end of the sentence where it is difficult to understand following two very long intervening phrases.

With function words, Rajkumar and White (2011) showed that they could improve upon the earlier model's predictions for when to employ *that*-complementizers using features inspired by Jaeger's (2010) work on using the principle of **uniform information density**, which holds that human language use tends to keep information density relatively constant in order to optimize communicative efficiency. In news text, com-

wsj_0015.7	the exact amount of the refund will be determined **next year** based on actual collections made until Dec. 31 of this year .
DEPLEN	[same]
DEPORD	the exact amount of the refund will be determined based on actual collections made until Dec. 31 of this year *next year* .
wsj_0020.1	the U.S. , claiming some success in its trade diplomacy , removed South Korea , Taiwan and Saudi Arabia from a list of countries it is closely watching for allegedly failing to honor U.S. patents , copyrights and other intellectual-property rights .
DEPLEN	the U.S. claiming some success in its trade diplomacy , removed **South Korea , Taiwan and Saudi Arabia** from a list of countries it is *watching closely* for allegedly failing to honor U.S. patents , copyrights and other intellectual-property rights .
DEPORD	the U.S. removed from a list of countries it is *watching closely* for allegedly failing to honor U.S. patents , copyrights and other intellectual-property rights , claiming some success in its trade diplomacy , *South Korea , Taiwan and Saudi Arabia* .

Table 1: Examples of realized output for full models with and without the dependency length feature (White and Rajkumar, 2012)

plementizers are left out two times out of three, but in some cases the presence of *that* is crucial to the interpretation. Generally, inserting a complementizer makes the onset of a complement clause more predictable, and thus less information dense, thereby avoiding a potential spike in information density that is associated with comprehension difficulty. Rajkumar & White's experiments confirmed the efficacy of the features based on Jaeger's work, including information density–based features, in a local classification model.[2] Their experiments also showed that the improvements in prediction accuracy apply to cases in which the presence of a *that*-complementizer arguably makes a substantial difference to fluency or intelligiblity. For example, in (1), the presence of *that* avoids a local ambiguity, helping the reader to understand that *for the second month in a row* modifies the reporting of the shortage; without *that*, it is very easy to mis-parse the sentence as having *for the second month in a row* modifying the saying event.

(1) He said that/∅? for the second month in a row, food processors reported a shortage of nonfat dry milk. (PTB WSJ0036.61)

Finally, to reduce the number of subject-verb agreement errors, Rajkumar and White (2010) extended the earlier model with features enabling it to make correct verb form choices in sentences involving complex coordinate constructions and with expressions such as *a lot of* where the correct choice is not determined solely by the head noun. They also improved animacy agreement with relativizers, reducing the number of errors where *that* or *which* was chosen to modify an animate noun rather than *who* or *whom* (and vice-versa), while also allowing both choices where corpus evidence was mixed.

[2]Note that the features from the local classification model for *that*-complementizer choice have not yet been incorporated into OpenCCG's global realization ranking model, and thus do not inform the baseline realization choices in this work.

3 Simple Reranking

3.1 Methods

We ran two OpenCCG surface realization models on the CCGbank dev set (derived from Section 00 of the Penn Treebank) and obtained n-best ($n = 10$) realizations. The first one is the baseline generative model (hereafter, generative model) used in training the averaged perceptron model. This model ranks realizations using the product of the Hockenmaier syntax model, n-gram models over words, POS tags and supertags in the training sections of the CCGbank, and the large-scale 5-gram model from Gigaword. The second one is the averaged perceptron model (hereafter, perceptron model), which uses all the features reviewed in Section 2. In order to experiment with multiple parsers, we used the Stanford dependencies (de Marneffe et al., 2006), obtaining gold dependencies from the gold-standard PTB parses and automatic dependencies from the automatic parses of each realization. Using dependencies allowed us to measure parse accuracy independently of word order. We chose the Berkeley parser (Petrov et al., 2006), Brown parser (Charniak and Johnson, 2005) and Stanford parser (Klein and Manning, 2003) to parse the realizations generated by the

	Berkeley	Brown	Stanford
No reranking	87.93	87.93	87.93
Labeled	87.77	87.87	87.12
Unlabeled	87.90	**87.97**	86.97

Table 2: Devset BLEU scores for simple ranking on top of n-best perceptron model realizations

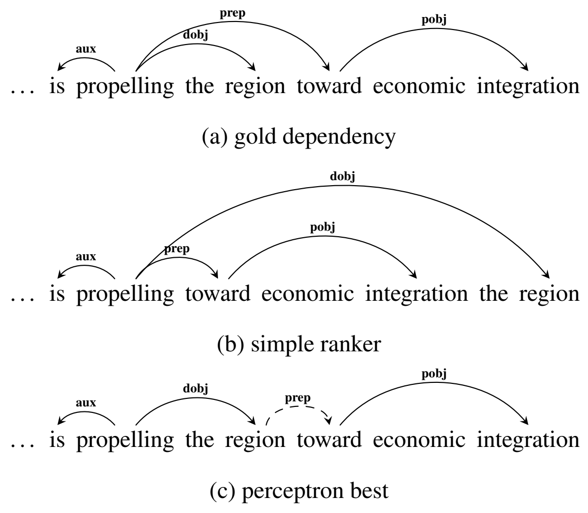

aux
prep
dobj
pobj
… is propelling the region toward economic integration
(a) gold dependency
dobj
aux
prep
pobj
… is propelling toward economic integration the region
(b) simple ranker
aux
dobj
prep
pobj
… is propelling the region toward economic integration
(c) perceptron best

Figure 2: Example parsing mistake in PP-attachment (wsj_0043.1)

two realization models and calculated precision, recall and F_1 of the dependencies for each realization by comparing them with the gold dependencies. We then ranked the realizations by their F_1 score of parse accuracy, keeping the original ranking in case of ties. We also tried using unlabeled (and unordered) dependencies, in order to possibly make better use of parses that were close to being correct. In this setting, as long as the right pair of tokens occur in a dependency relation, it was counted as a correctly recovered dependency.

3.2 Results

Simple ranking with the Berkeley parser of the generative model's n-best realizations raised the BLEU score from 85.55 to 86.07, well below the averaged perceptron model's BLEU score of 87.93. However, as shown in Table 2, none of the parsers yielded significant improvements on the top of the perceptron model.

Inspecting the results of simple ranking revealed that while simple ranking did successfully avoid vicious ambiguities in some cases, parser mistakes with PP-attachments, noun-noun compounds and coordinate structures too often blocked the gold realization from emerging on top. To illustrate, Figure 2 shows an example with a PP-attachment mistake. In the figure, the key gold dependencies of the reference sentence are shown in (a), the dependencies of the realization selected by the simple ranker are shown in (b), and the dependencies of the realization selected by the perceptron ranker (same as gold) appear in (c), with the parsing mistake indicated by the dashed line. The simple ranker ends up choosing (b) as the best realization because it has the most accurate parse compared to the reference sentence, given the mistake with (c).

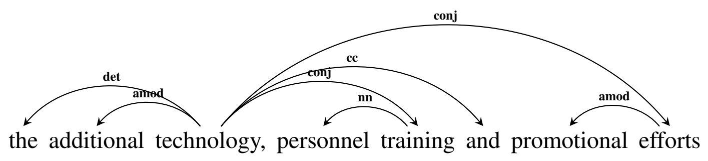
Other common parse errors are illustrated in Figure 3. Here, (b) ends up getting chosen by the simple ranker as the realization with the most accurate parse given the failures in (c), where *the additional technology, personnel training* is mistakenly analyzed as one noun phrase, a reading unlikely to be considered by human readers.

In sum, although simple ranking helps to avoid vicious ambiguity in some cases, the overall results of simple ranking are no better than the perceptron model (according to BLEU, at least), as parse failures that are not reflective of human intepretive tendencies too often lead the ranker to choose dispreferred realizations. As such, we turn now to a more nuanced model for combining the results of multiple parsers in a way that is less sensitive to such parsing mistakes, while also letting the perceptron model have a say in the final ranking.

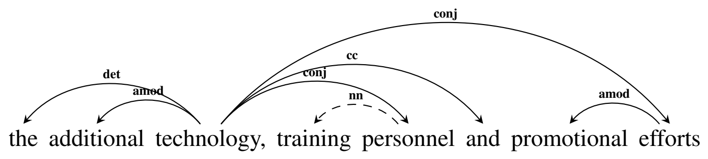
4 Reranking with SVMs

4.1 Methods

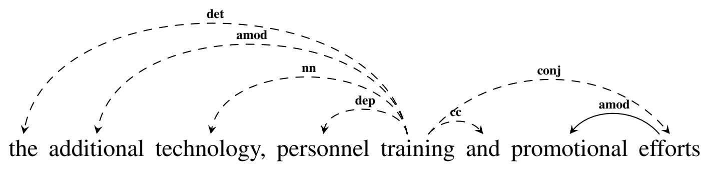
Since different parsers make different errors, we conjectured that dependencies in the intersection of the output of multiple parsers may be more reliable and thus may more reliably reflect human comprehension preferences. Similarly, we conjectured that large differences in the realizer's perceptron model score may more reliably reflect human fluency preferences than small ones, and thus we combined this score with features for parser accuracy in an SVM ranker. Additionally, given that parsers may more reliably recover some kinds of dependencies than others, we included features for each dependency type, so that the SVM ranker might learn how to weight them appropriately. Finally, since the differences among the n-best parses reflect the least certain parsing decisions,


conj
cc
det
conj
amod
nn
amod
the additional technology, personnel training and promotional efforts

(a) gold dependency


conj
cc
det
conj
amod
nn
amod
the additional technology, training personnel and promotional efforts

(b) simple ranker


det
amod
nn
conj
dep
amod
cc
the additional technology, personnel training and promotional efforts

(c) perceptron best

Figure 3: Example parsing mistakes in a noun-noun compound and a coordinate structure (wsj_0085.45)

and thus ones that may require more common sense inference that is easy for humans but not machines, we conjectured that including features from the n-best parses may help to better match human performance. In more detail, we made use of the following feature classes for each candidate realization:

perceptron model score the score from the realizer's model, normalized to [0,1] for the realizations in the n-best list

precision and recall labeled and unlabeled precision and recall for each parser's best parse

per-label precision and recall (*dep*) precision and recall for each type of dependency obtained from each parser's best parse (using zero if not defined for lack of predicted or gold dependencies with a given label)

n-best precision and recall (*nbest*) labeled and unlabeled precision and recall for each parser's top five parses, along with the same features for the most accurate of these parses

In training, we used the BLEU scores of each realization compared with its reference sentence to establish a preference order over pairs of candidate realizations, assuming that the original corpus sentences are generally better than related alternatives, and that BLEU can somewhat reliably predict human preference judgments.

We trained the SVM ranker (Joachims, 2002) with a linear kernel and chose the hyper-parameter c, which tunes the trade-off between training error and margin, with 6-fold cross-validation on the devset. We trained different models to investigate the contribution made by different parsers and different types of features, with the perceptron model score included as a feature in all models. For each parser, we trained a model with its overall precision and recall features, as shown at the top of Table 3. Then we combined these three models to get a new model (Bkl+Brw+St in the table) . Next, to this combined model we separately added (i) the per-label precision and recall features from all the parsers (BBS+dep), and (ii) the n-best features from the parsers (BBS+nbest). The full model (BBS+dep+nbest) includes all the features listed above. Finally, since the Berkeley parser yielded the best results on its own, we also tested models using all the feature classes but only using this parser by itself.

4.2 Results

Table 3 shows the results of different SVM ranking models on the devset. We calculated significance using paired bootstrap resampling (Koehn, 2004).[3] Both the per-label precision & recall fea-

[3]Kudos to Kevin Gimpel for making his implementation available: `http://www.ark.cs.cmu.edu/MT/paired_bootstrap_v13a.tar.gz`

	BLEU	sig.
perceptron baseline	87.93	–
Berkeley	88.45	*
Brown	88.34	
Stanford	88.18	
Bkl+Brw+St	88.44	*
BBS+dep	88.63	**
BBS+nbest	88.60	**
BBS+dep+nbest	**88.73**	**
Bkl+dep	88.63	**
Bkl+nbest	88.48	*
Bkl +dep+nbest	88.68	**

Table 3: Devset results of SVM ranking on top of perceptron model. Significance codes: $**$ for $p < 0.05$, $*$ for $p < 0.1$.

	BLEU	sig.
perceptron baseline	86.94	–
BBS+dep+nbest	**87.64**	**

Table 4: Final test results of SVM ranking on top of perceptron model. Significance codes: $**$ for $p < 0.05$, $*$ for $p < 0.1$.

tures and the n-best parse features contributed to achieving a significant improvement compared to the perceptron model. Somewhat surprisingly, the Berkeley parser did as well as all three parsers using just the overall precision and recall features, but not quite as well using all features. The complete model, BBS+dep+nbest, achieved a BLEU score of 88.73, significantly improving upon the perceptron model ($p < 0.02$). We then confirmed this result on the final test set, Section 23 of the CCGbank, as shown in Table 4 ($p < 0.02$ as well).

5 Analysis and Discussion

5.1 Targeted Manual Analysis

In order to gain a better understanding of the successes and failures of our SVM ranker, we present here a targeted manual analysis of the development set sentences with the greatest change in BLEU scores, carried out by the second author (a native speaker). In this analysis, we consider whether the reranked realization improves upon or detracts from realization quality—in terms of adequacy, fluency, both or neither—along with a linguistic categorization of the differences between the reranked realization and the original top-ranked realization according to the averaged perceptron model. Unlike the broad-based and objective evaluation in terms of BLEU scores presented above, this analysis is narrowly targeted and subjective, though the interested reader is invited to review the complete set of analyzed examples that accompany the paper as a supplement. We leave a more broad-based human evaluation by naive subjects for future work.

Table 5 shows the results of the analysis, both overall and for the most frequent categories of changes. Of the 50 sentences where the BLEU score went up the most, 15 showed an improvement in adequacy (i.e., in conveying the intended meaning), 22 showed an improvement in fluency (with 3 cases also improving adequacy), and 16 yielded no discernible change in fluency or adequacy. By contrast, with the 50 sentences where the BLEU score went down the most, adequacy was only affected 4 times, though fluency was affected 32 times, and 15 remained essentially unchanged.[4] The table also shows that differences in the order of VP constituents usually led to a change in adequacy or fluency, as did ordering changes within NPs, with noun-noun compounds and named entities as the most frequent subcategories of NP-ordering changes. Of the cases where adequacy and fluency were not affected, contractions and subject-verb inversions were the most frequent differences.

Examples of the changes yielded by the SVM ranker appear in Table 6. With wsj_0036.54, the averaged perceptron model selects a realization that regrettably (though amusingly) swaps *purchasing* and *more than 250*—yielding a sentence that suggests that the executives have been purchased!—while the SVM ranker succeeds in ranking the original sentence above all competing realizations. With wsj_0088.25, self-monitoring with the SVM ranker yields a realization nearly identical to the original except for an extra comma, where it is clear that *in public* modifies *do this*; by contrast, in the perceptron-best realization, *in public* mistakenly appears to modify *be disclosed*. With wsj_0041.18, the SVM ranker unfortunately prefers a realization where *presumably* seems to modify *shows* rather than *of two politicians* as

[4]The difference in the distribution of adequacy change, fluency change and no change counts between the two conditions is highly significant statistically ($\chi^2 = 9.3, df = 2, p < 0.01$). In this comparison, items where both fluency and adequacy were affected were counted as adequacy cases.

	±adq	±flu	=eq	±vpord	±npord	±nn	±ne	=vpord	=sbjinv	=cntrc
BLEU wins	15	22	16	10	9	7	3	4	-	11
BLEU losses	4	32	15	8	13	5	5	4	7	-

Table 5: Manual analysis of devset sentences where the SVM ranker achieved the greatest increase/decrease in BLEU scores (50 each of wins/losses) compared to the averaged perceptron baseline model in terms of positive or negative changes in adequacy (±adq), fluency (±flu) or neither (=eq); changes in VP ordering (±vpord), NP ordering (±npord), noun-noun compound ordering (±nn) and named entities (±ne); and neither positive nor negative changes in VP ordering (=vpord), subject-inversion (=sbjinv) and contractions (=cntrc). In all but one case (counted as =eq here), the BLEU wins saw positive changes and the BLEU losses saw negative changes.

wsj_0036.54	the purchasing managers ' report is based on data provided by more than 250 **purchasing** executives .
SVM RANKER	[same]
PERCEP BEST	the purchasing managers ' report is based on data provided by *purchasing* more than 250 executives .
wsj_0088.25	Markey said we could have done this **in public** because so little sensitive information was disclosed , the aide said .
SVM RANKER	Markey said , we could have done this **in public** because so little sensitive information was disclosed , the aide said .
PERCEP BEST	Markey said , we could have done this because so little sensitive information was disclosed *in public* , the aide said .
wsj_0041.18	the screen shows two distorted , unrecognizable photos , **presumably** of two politicians .
SVM RANKER	the screen shows two distorted , unrecognizable photos *presumably* , of two politicians .
PERCEP BEST	[same as original]
wsj_0044.111	" I was dumbfounded " , Mrs. Ward **recalls** .
SVM RANKER	" I was dumbfounded " , *recalls* Mrs. Ward .
PERCEP BEST	[same as original]

Table 6: Examples of devset sentences where the SVM ranker improved adequacy (top), made it worse (middle) or left it the same (bottom)

in the original, which the averaged perceptron model prefers. Finally, wsj_0044.111 is an example where a subject-inversion makes no difference to adequacy or fluency.

5.2 Discussion

The BLEU evaluation and targeted manual analysis together show that the SVM ranker increases the similarity to the original corpus of realizations produced with self-monitoring, often in ways that are crucial for the intended meaning to be apparent to human readers.

A limitation of the experiments reported in this paper is that OpenCCG's input semantic dependency graphs are not the same as the Stanford dependencies used with the Treebank parsers, and thus we have had to rely on the gold parses in the PTB to derive gold dependencies for measuring accuracy of parser dependency recovery. In a realistic application scenario, however, we would need to measure parser accuracy relative to the realizer's input. We initially tried using OpenCCG's parser in a simple ranking approach, but found that it did not improve upon the averaged perceptron model, like the three parsers used subsequently. Given that with the more refined SVM ranker, the Berkeley parser worked nearly as well as all three parsers together using the complete feature set, the prospects for future work on a more realistic scenario using the OpenCCG parser in an SVM ranker for self-monitoring now appear much more promising, either using OpenCCG's reimplementation of Hockenmaier & Steedman's generative CCG model, or using the Berkeley parser trained on OpenCCG's enhanced version of the CCG-bank, along the lines of Fowler and Penn (2010).

6 Related Work

Approaches to surface realization have been developed for LFG, HPSG, and TAG, in addition to CCG, and recently statistical dependency-based approaches have been developed as well; see the report from the first surface realization shared task (Belz et al., 2010; Belz et al., 2011) for an overview. To our knowledge, however, a comprehensive investigation of avoiding vicious structural ambiguities with broad coverage statistical parsers has not been previously explored. As our SVM ranking model does not make use of CCG-specific features, we would expect our self-monitoring method to be equally applicable to realizers using other frameworks.

7 Conclusion

In this paper, we have shown that while using parse accuracy in a simple reranking strategy for self-monitoring fails to improve BLEU scores over a state-of-the-art averaged perceptron realization ranking model, it is possible to significantly increase BLEU scores using an SVM ranker that combines the realizer's model score together with features from multiple parsers, including ones designed to make the ranker more robust to parsing mistakes that human readers would be unlikely to make. Additionally, via a targeted manual analysis, we showed that the SVM reranker frequently manages to avoid egregious errors involving "vicious" ambiguities, of the kind that would mislead human readers as to the intended meaning.

As noted in Reiter's (2010) survey, many NLG systems use surface realizers as off-the-shelf components. In this paper, we have focused on broad coverage surface realization using widely-available PTB data—where there are many sentences of varying complexity with gold-standard annotations—following the common assumption that experiments with broad coverage realization are (or eventually will be) relevant for NLG applications. Of course, the kinds of ambiguity that can be problematic in news text may or may not be the same as the ones encountered in particular applications. Moreover, for certain applications (e.g. ones with medical or legal implications), it may be better to err on the side of ambiguity avoidance, even at some expense to fluency, thereby requiring training data reflecting the desired trade-off to adapt the methods described here. We leave these application-centered issues for investigation in future work.

The current approach is primarily suitable for offline use, for example in report generation where there are no real-time interaction demands. In future work, we also plan to investigate ways that self-monitoring might be implemented more efficiently as a combined process, rather than running independent parsers as a post-process following realization.

Acknowledgments

We thank Mark Johnson, Micha Elsner, the OSU Clippers Group and the anonymous reviewers for helpful comments and discussion. This work was supported in part by NSF grants IIS-1143635 and IIS-1319318.

References

S. Abney. 1996. Statistical methods and linguistics. In Judith Klavans and Philip Resnik, editors, *The balancing act: Combining symbolic and statistical approaches to language*, pages 1–26. MIT Press, Cambridge, MA.

Anja Belz, Mike White, Josef van Genabith, Deirdre Hogan, and Amanda Stent. 2010. Finding common ground: Towards a surface realisation shared task. In *Proceedings of INLG-10, Generation Challenges*, pages 267–272.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France, September. Association for Computational Linguistics.

Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*.

F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis. 2006. Identifying nocuous ambiguities in natural language requirements. In *Requirements Engineering, 14th IEEE International Conference*, pages 59–68. IEEE.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proc. EMNLP-02*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio, June. Association for Computational Linguistics.

Timothy A. D. Fowler and Gerald Penn. 2010. Accurate context-free parsing with Combinatory Categorial Grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344, Uppsala, Sweden, July. Association for Computational Linguistics.

Edward Gibson. 2000. Dependency locality theory: A distance-based theory of linguistic complexity. In Alec Marantz, Yasushi Miyashita, and Wayne O'Neil, editors, *Image, Language, brain: Papers from the First Mind Articulation Project Symposium*. MIT Press, Cambridge, MA.

Daniel Gildea and David Temperley. 2010. Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proc. ACL-02*.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

T. Florian Jaeger. 2010. Redundancy and reduction: Speakers manage information density. *Cognitive Psychology*, 61(1):23–62, August.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. KDD*.

I.H. Khan, K. Van Deemter, and G. Ritchie. 2008. Generation of referring expressions: Managing structural ambiguities. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Willem J. M. Levelt. 1989. *Speaking: From Intention to Articulation*. MIT Press.

Günter Neumann and Gertjan van Noord. 1992. Self-monitoring with reversible grammars. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 700–706, Stroudsburg, PA, USA. Association for Computational Linguistics.

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. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL-02*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

Rajakrishnan Rajkumar and Michael White. 2010. Designing agreement features for realization ranking. In *Proc. Coling 2010: Posters*, pages 1032–1040, Beijing, China, August.

Rajakrishnan Rajkumar and Michael White. 2011. Linguistically motivated complementizer choice in surface realization. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, pages 39–44, Edinburgh, Scotland, July. Association for Computational Linguistics.

Rajakrishnan Rajkumar, Michael White, and Dominic Espinosa. 2009. Exploiting named entity classes in CCG surface realization. In *Proc. NAACL HLT 2009 Short Papers*.

Ehud Reiter and Robert Dale. 2000. *Building natural generation systems*. Studies in Natural Language Processing. Cambridge University Press.

Ehud Reiter. 2010. Natural language generation. In Alexander Clark, Chris Fox, and Shalom Lappin, editors, *The Handbook of Computational Linguistics and Natural Language Processing (Blackwell Handbooks in Linguistics)*, Blackwell Handbooks in Linguistics, chapter 20. Wiley-Blackwell, 1 edition.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

David Temperley. 2007. Minimization of dependency length in written English. *Cognition*, 105(2):300–333.

K. Van Deemter. 2004. Towards a probabilistic version of bidirectional OT syntax and semantics. *Journal of Semantics*, 21(3):251–280.

Michael White and Rajakrishnan Rajkumar. 2008. A more precise analysis of punctuation for broad-coverage surface realization with CCG. In *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, pages 17–24.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore, August. Association for Computational Linguistics.

Michael White and Rajakrishnan Rajkumar. 2012. Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255, Jeju Island, Korea, July. Association for Computational Linguistics.

Michael White. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language & Computation*, 4(1):39–75.