Identifying Semantic Roles Using Combinatory Categorial Grammar

Daniel Gildea and **Julia Hockenmaier**
University of Pennsylvania
{dgildea,juliahr}@cis.upenn.edu

Abstract

We present a system for automatically identifying PropBank-style semantic roles based on the output of a statistical parser for Combinatory Categorial Grammar. This system performs at least as well as a system based on a traditional Treebank parser, and outperforms it on core argument roles.

1 Introduction

Correctly identifying the semantic roles of sentence constituents is a crucial part of interpreting text, and in addition to forming an important part of the information extraction problem, can serve as an intermediate step in machine translation or automatic summarization. Even for a single predicate, semantic arguments can have multiple syntactic realizations, as shown by the following paraphrases:

(1) John will meet with Mary.
John will meet Mary.
John and Mary will meet.

(2) The door opened.
Mary opened the door.

Recently, attention has turned to creating corpora annotated with argument structures. The PropBank (Kingsbury and Palmer, 2002) and the FrameNet (Baker et al., 1998) projects both document the variation in syntactic realization of the arguments of predicates in general English text. Gildea and Palmer (2002) developed a system to predict semantic roles (as defined in PropBank) from sentences and their parse trees as determined by the statistical parser of Collins (1999). In this paper, we examine how the syntactic representations used by different statistical parsers affect the performance of such a system. We compare a parser based on Combinatory Categorial Grammar (CCG) (Hockenmaier and Steedman, 2002b) with the Collins parser. As the CCG parser is trained and tested on a corpus of CCG derivations that have been obtained by automatic conversion from the Penn Treebank, we are able to compare performance using both gold-standard and automatic parses for both CCG and the traditional Treebank representation. The Treebank-parser returns skeletal phrase-structure trees without the traces or functional tags in the original Penn Treebank, whereas the CCG parser returns word-word dependencies that correspond to the underlying predicate-argument structure, including long-range dependencies arising through control, raising, extraction and coordination.

2 Predicate-argument relations in PropBank

The Proposition Bank (Kingsbury and Palmer, 2002) provides a human-annotated corpus of semantic verb-argument relations. For each verb appearing in the corpus, a set of semantic roles is defined. Roles for each verb are simply numbered Arg0, Arg1, Arg2, etc. As an example, the entry-specific roles for the verb *offer* are given below:

Arg0	entity offering
Arg1	commodity
Arg2	price
Arg3	benefactive or entity offered to

These roles are then annotated for every instance of the verb appearing in the corpus, including the following examples:

(3) [$_{\text{ARG0}}$ the company] to *offer* [$_{\text{ARG1}}$ a 15% stake] to [$_{\text{ARG2}}$ the public].

(4) [$_{\text{ARG0}}$ Sotheby's] ... *offered* [$_{\text{ARG2}}$ the Dorrance heirs] [$_{\text{ARG1}}$ a money-back guarantee]

(5) [$_{\text{ARG1}}$ an amendment] *offered* by [$_{\text{ARG0}}$ Rep. Peter DeFazio]

(6) [$_{\text{ARG2}}$ Subcontractors] will be *offered* [$_{\text{ARG1}}$ a settlement]

A variety of additional roles are assumed to apply across all verbs. These secondary roles can be thought of as being adjuncts, rather than arguments, although no claims are made as to optionality or other traditional argument/adjunct tests. The secondary roles include:

Location	*in Tokyo, outside*
Time	*last week, on Tuesday, never*
Manner	*easily, dramatically*
Direction	*south, into the wind*
Cause	*due to pressure from Washington*
Discourse	*however, also, on the other hand*
Extent	*15%, 289 points*
Purpose	*to satisfy requirements*
Negation	*not, n't*
Modal	*can, might, should, will*
Adverbial	*(none of the above)*

and are represented in PropBank as "ArgM" with an additional function tag, for example ArgM-TMP for temporal. We refer to PropBank's numbered arguments as "core" arguments. Core arguments represent 75% of the total labeled roles in the PropBank data. Our system predicts all the roles, including core arguments as well as the ArgM labels and their function tags.

3 Predicate-argument relations in CCG

Combinatory Categorial Grammar (CCG) (Steedman, 2000), is a grammatical theory which provides a completely transparent interface between surface syntax and underlying semantics, such that each syntactic derivation corresponds directly to an interpretable semantic representation which includes long-range dependencies that arise through control, raising, coordination and extraction.

In CCG, words are assigned atomic categories such as NP, or functor categories like $(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}$ (transitive declarative verb) or $\mathsf{S/S}$ (sentential modifier). Adjuncts are represented as functor categories such as $\mathsf{S/S}$ which expect and return the same type. We use indices to number the arguments of functor categories, eg. $(\mathsf{S[dcl]\backslash NP_1})/\mathsf{NP_2}$, or $\mathsf{S/S_1}$, and indicate the word-word dependencies in the predicate-argument structure as tuples $\langle w_h, c_h, i, w_a\rangle$, where c_h is the lexical category of the head word w_h, and w_a is the head word of the constituent that fills the ith argument of c_h.

Long-range dependencies can be projected through certain types of lexical categories or through rules such as coordination of functor categories. For example, in the lexical category of a relative pronoun, $(\mathsf{NP\backslash NP_i})/(\mathsf{S[dcl]/NP_i})$, the head of the NP that is missing from the relative clause is unified with (as indicated by the indices i) the head of the NP that is modified by the entire relative clause.

Figure 1 shows the derivations of an ordinary sentence, a relative clause and a right-node-raising construction. In all three sentences, the predicate-argument relations between *London* and *denied* and *plans* and *denied* are the same, which in CCG is expressed by the fact that *London* fills the first (ie. subject) argument slot of the lexical category of *denied*, $(\mathsf{S[dcl]\backslash NP_1})/\mathsf{NP_2}$, and *plans* fills the second (object) slot. The relations extracted from the CCG derivation for the sentence *"London denied plans on Monday"* are shown in Table 1.

The CCG parser returns the local and long-range word-word dependencies that express the predicate-argument structure corresponding to the derivation. These relations are recovered with an accuracy of around 83% (labeled recovery) or 91% (unlabeled recovery) (Hockenmaier, 2003). By contrast, standard Treebank parsers such as (Collins, 1999) only return phrase-structure trees, from which non-local dependencies are difficult to recover.

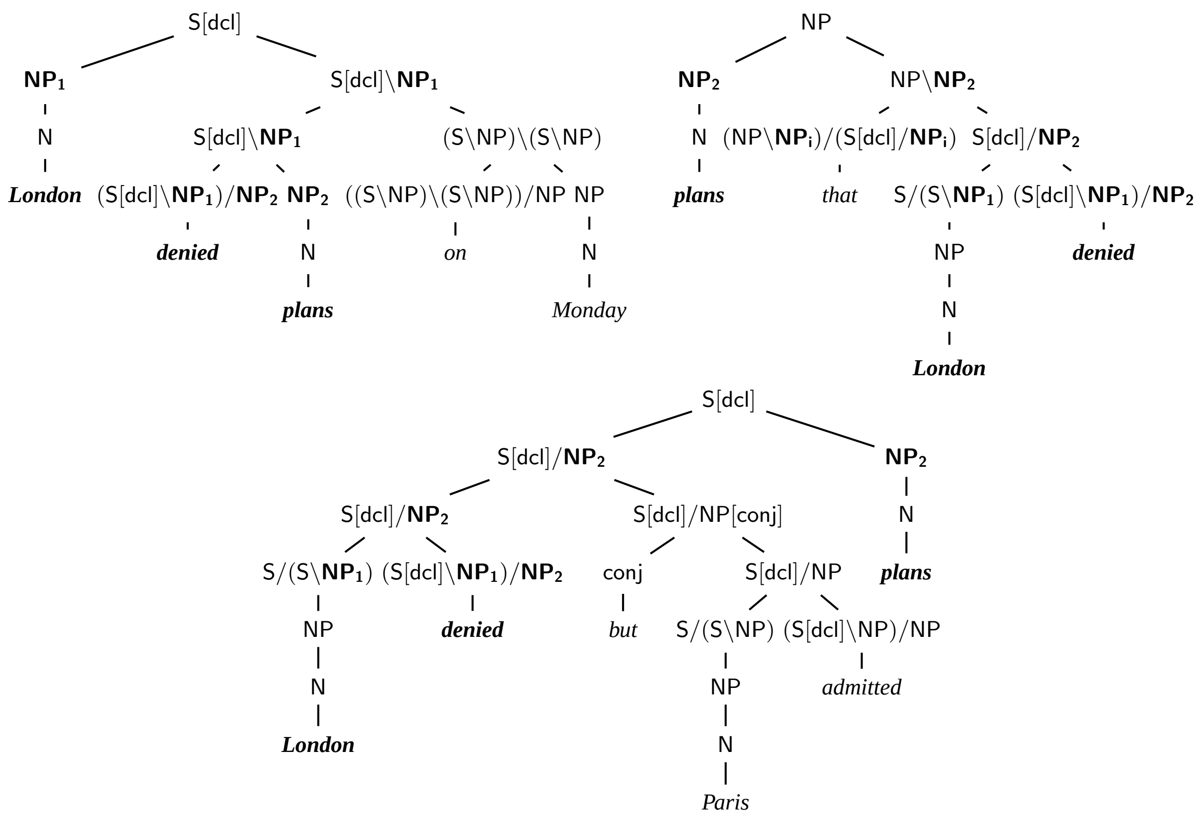

Figure 1: CCG derivation trees for three clauses containing the same predicate-argument relations.

w_h	c_h	i	w_a
denied	$(\mathsf{S[dcl]}\backslash\mathbf{NP}_1)/\mathsf{NP}_2$	1	*London*
denied	$(\mathsf{S[dcl]}\backslash\mathsf{NP}_1)/\mathbf{NP}_2$	2	*plans*
on	$((\mathsf{S}\backslash\mathsf{NP}_1)\backslash(\mathbf{S}\backslash\mathbf{NP})_2)/\mathsf{NP}_3$	2	*denied*
on	$((\mathsf{S}\backslash\mathsf{NP}_1)\backslash(\mathsf{S}\backslash\mathsf{NP})_2)/\mathbf{NP}_3$	3	*Monday*

Table 1: CCG predicate-argument relations for the sentence *"London denied plans on Monday"*

The CCG parser has been trained and tested on CCGbank (Hockenmaier and Steedman, 2002a), a treebank of CCG derivations obtained from the Penn Treebank, from which we also obtain our training data.

4 Mapping between PropBank and CCGbank

Our aim is to use CCG derivations as input to a system for automatically producing the argument labels of PropBank. In order to do this, we wish to correlate the CCG relations above with PropBank arguments. PropBank argument labels are assigned to nodes in the syntactic trees from the Penn Treebank. While the CCGbank is derived from the Penn Treebank, in many cases the constituent structures do not correspond. That is, there may be no constituent in the CCG derivation corresponding to the same sequence of words as a particular constituent in the Treebank tree. For this reason, we compute the correspondence between the CCG derivation and the PropBank labels at the level of head words. For each role label for a verb's argument in PropBank, we first find the head word for its constituent according to the the head rules of (Collins, 1999). We then look for the label of the CCG relation between this head word and the verb itself.

5 The Experiments

In previous work using the PropBank corpus, Gildea and Palmer (2002) developed a system to predict semantic roles from sentences and their parse trees as determined by the statistical parser of Collins (1999). We will briefly review their probability model before adapting the system to incorporate features from the CCG derivations.

5.1 The model of Gildea and Palmer (2002)

For the Treebank-based system, we use the probability model of Gildea and Palmer (2002). Probabilities of a parse constituent belonging to a given semantic role are calculated from the following features:

The **phrase type** feature indicates the syntactic type of the phrase expressing the semantic roles: examples include noun phrase (NP), verb phrase (VP), and clause (S).

The **parse tree path** feature is designed to capture the syntactic relation of a constituent to the predicate. It is defined as the path from the predicate through the parse tree to the constituent in question, represented as a string of parse tree nonterminals linked by symbols indicating upward or downward movement through the tree, as shown in Figure 2. Although the path is composed as a string of symbols, our systems will treat the string as an atomic value. The path includes, as the first element of the string, the part of speech of the predicate, and, as the last element, the phrase type or syntactic category of the sentence constituent marked as an argument.

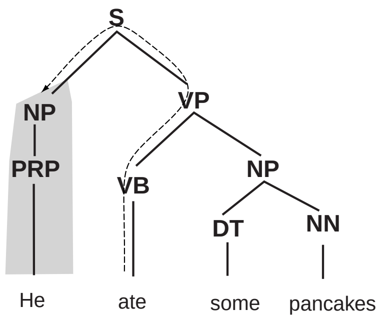

Figure 2: In this example, the **path** from the predicate *ate* to the argument NP *He* can be represented as VB↑VP↑S↓NP, with ↑ indicating upward movement in the parse tree and ↓ downward movement.

The **position** feature simply indicates whether the constituent to be labeled occurs before or after the predicate. This feature is highly correlated with grammatical function, since subjects will generally appear before a verb, and objects after. This feature may overcome the shortcomings of reading grammatical function from the parse tree, as well as errors in the parser output.

The **voice** feature distinguishes between active and passive verbs, and is important in predicting semantic roles because direct objects of active verbs correspond to subjects of passive verbs. An instance of a verb was considered passive if it is tagged as a past participle (e.g. *taken*), unless it occurs as a descendent verb phrase headed by any form of *have* (e.g. *has taken*) without an intervening verb phrase headed by any form of *be* (e.g. *has been taken*).

The **head word** is a lexical feature, and provides information about the semantic type of the role filler. Head words of nodes in the parse tree are determined using the same deterministic set of head word rules used by Collins (1999).

The system attempts to predict argument roles in new data, looking for the highest probability assignment of roles r_i to all constituents i in the sentence, given the set of features $F_i = \{pt_i, path_i, pos_i, v_i, h_i\}$ at each constituent in the parse tree, and the predicate p:

$$argmax_{r_{1..n}} P(r_{1..n}|F_{1..n}, p)$$

We break the probability estimation into two parts, the first being the probability $P(r_i|F_i, p)$ of a constituent's role given our five features for the consituent, and the predicate p. Due to the sparsity of the data, it is not possible to estimate this probability from the counts in the training data. Instead, probabilities are estimated from various subsets of the features, and interpolated as a linear combination of the resulting distributions. The interpolation is performed over the most specific distributions for which data are available, which can be thought of as choosing the topmost distributions available from a backoff lattice, shown in Figure 3.

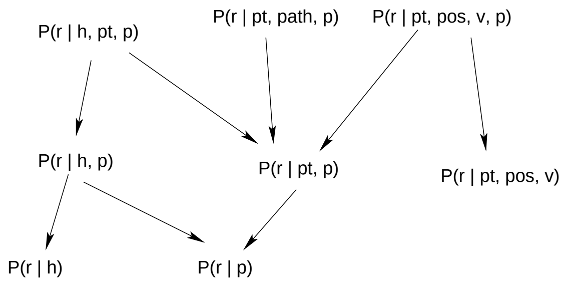

Figure 3: Backoff lattice with more specific distributions towards the top.

The probabilities $P(r_i|F_i, p)$ are combined with the probabilities $P(\{r_{1..n}\}|p)$ for a set of roles appearing in a sentence given a predicate, using the following formula:

$$P(r_{1..n}|F_{1..n}, p) \approx P(\{r_{1..n}\}|p) \prod_i \frac{P(r_i|F_i, p)}{P(r_i|p)}$$

This approach, described in more detail in Gildea and Jurafsky (2002), allows interaction between the role assignments for individual constituents while making certain independence assumptions necessary for efficient probability estimation. In particular, we assume that sets of roles appear independent of their linear order, and that the features F of a constituents are independent of other constituents' features given the constituent's role.

5.2 The model for CCG derivations

In the CCG version, we replace the features above with corresponding features based on both the sentence's CCG derivation tree (shown in Figure 1) and the CCG predicate-argument relations extracted from it (shown in Table 1).

The **parse tree path** feature, designed to capture grammatical relations between constituents, is replaced with a feature defined as follows: If there is a dependency in the predicate-argument structure of the CCG derivation between two words w and w', the path feature from w to w' is defined as the lexical category of the functor, the argument slot i occupied by the argument, plus an arrow ($\leftarrow$ or $\rightarrow$) to indicate whether w or w' is the categorial functor. For example, in our sentence *"London denied plans on Monday"*, the relation connecting the verb *denied* with *plans* is $(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}.2.\leftarrow$, with the left arrow indicating the lexical category included in the relation is that of the verb, while the relation connecting *denied* with *on* is $((\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP}))/\mathsf{NP}.2.\rightarrow$, with the right arrow indicating the the lexical category included in the relation is that of the modifier.

If the CCG derivation does not define a predicate-argument relation between the two words, we use the parse tree path feature described above, defined over the CCG derivation tree. In our training data, 77% of PropBank arguments corresponded directly to a relation in the CCG predicate-argument representation, and the path feature was used for the remaining 23%. Most of these mismatches arise because the CCG parser and PropBank differ in their definition of head words. For instance, the CCG parser always assumes that the head of a PP is the preposition, whereas PropBank roles can be assigned to the entire PP (7), or only to the NP argument of the preposition (8), in which case the head word comes from the NP:

(7) ... will be *offered* [PP$_{\text{ARGM-LOC}}$ *in the U.S*].

(8) to *offer* ...[PP to [NP$_{\text{ARG2}}$ *the public*]].

In embedded clauses, CCG assumes that the head is the complementizer, whereas in PropBank, the head comes from the embedded sentence itself. In complex verb phrases (eg. *"might not have gone"*), the CCG parser assumes that the first auxiliary (*might*) is head, whereas PropBank assumes it is the main verb (*gone*). Therefore, CCG assumes that *not* modifies *might*, whereas PropBank assumes it modifies *gone*. Although the head rules of the parser could in principle be changed to reflect more directly the dependencies in PropBank, we have not attempted to do so yet. Further mismatches occur because the predicate-argument structure returned by the CCG parser only contains syntactic dependencies, whereas the PropBank data also contain some anaphoric dependencies, eg.:

(9) [$_{\text{ARG0}}$ Realist 's] negotiations to *acquire* Ammann Laser Technik AG...

(10) When properly *applied*, [$_{\text{ARG0}}$ the adhesive] is designed to...

Such dependencies also do not correspond to a relation in the predicate-argument structure of the CCG derivation, and cause the path feature to be used.

The **phrase type** feature is replaced with the lexical category of the maximal projection of the PropBank argument's head word in the CCG derivation tree. For example, the category of *plans* is N, and the category of *denied* is $(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}$.

The **voice** feature can be read off the CCG categories, since the CCG categories of past participles carry different features in active and passive voice (eg. *sold* can be $(\mathsf{S[pt]\backslash NP})/\mathsf{NP}$ or $\mathsf{S[pss]\backslash NP}$).

The **head word** of a constituent is indicated in the derivations returned by the CCG parser.

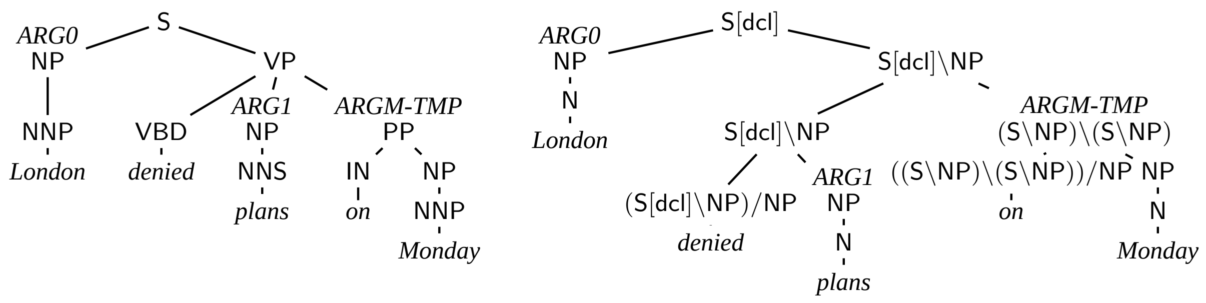

Figure 4: A sample sentence as produced by the Treebank parser (left) and by the CCG parser (right). Nodes are annotated with PropBank roles *ARG0*, *ARG1* and *ARGM-TMP*.

		Treebank-based			*CCG-based*		
Features extracted from	*Args*	*Precision*	*Recall*	*F-score*	*Precision*	*Recall*	*F-score*
Automatic parses	core	75.9	69.6	72.6	76.1	73.5	74.8
	all	72.6	61.2	66.4	71.0	63.1	66.8
Gold-standard parses	core	85.5	81.7	83.5	82.4	78.6	80.4
	all	78.8	69.9	74.1	76.3	67.8	71.8
Gold-standard w/o traces	core	77.6	75.2	76.3			
	all	74.4	66.5	70.2			

Table 2: Accuracy of semantic role prediction

5.3 Data

We use data from the November 2002 release of PropBank. The dataset contains annotations for 72,109 predicate-argument structures with 190,815 individual arguments (of which 75% are core, or numbered, arguments) and has includes examples from 2462 lexical predicates (types). Annotations from Sections 2 through 21 of the Treebank were used for training; Section 23 was the test set. Both parsers were trained on Sections 2 through 21.

6 Results

Because of the mismatch between the constituent structures of CCG and the Treebank, we score both systems according to how well they identify the head words of PropBank's arguments. Table 2 gives the performance of the system on both PropBank's core, or numbered, arguments, and on all PropBank roles including the adjunct-like ArgM roles. In order to analyze the impact of errors in the syntactic parses, we present results using features extracted from both automatic parser output and the gold standard parses in the Penn Treebank (without functional tags) and in CCGbank. Using the gold standard parses provides an upper bound on the performance of the system based on automatic parses. Since the Collins parser does not provide trace information, its upper bound is given by the system tested on the gold-standard Treebank representation with traces removed. In Table 2, "core" indicates results on PropBank's numbered arguments (ARG0...ARG5) only, and "all" includes numbered arguments as well as the ArgM roles. Most of the numbered arguments (in particular ARG0 and ARG1) correspond to arguments that the CCG category of the verb directly subcategorizes for. The CCG-based system outperforms the system based on the Collins parser on these core arguments, and has comparable performance when all PropBank labels are considered. We believe that the superior performance of the CCG system on this core arguments is due to its ability to recover long-distance dependencies, whereas we attribute its lower performance on non-core arguments mainly to the mismatches between PropBank and CCGbank.

The importance of long-range dependencies for our task is indicated by the fact that the performance on the Penn Treebank gold standard without traces

		Treebank-based			*CCG-based*		
	Scoring	*Precision*	*Recall*	*F-score*	*Precision*	*Recall*	*F-score*
Automatic parses	Head word	72.6	61.2	66.4	71.0	63.1	66.8
	Boundary	68.6	57.8	62.7	55.7	49.5	52.4
Gold-standard parses	Head word	77.6	75.2	76.3	76.3	67.8	71.8
(Treebank: w/o traces)	Boundary	74.4	66.5	70.2	67.5	60.0	63.5

Table 3: Comparison of scoring regimes, using automatic parser output and gold standard parses. The first row in this table corresponds to the second row in Table 2.

is significantly lower than that on the Penn Treebank with trace information. Long-range dependencies are especially important for core arguments, shown by the fact that removing trace information from the Treebank parses results in a bigger drop for core arguments (83.5 to 76.3 F-score) than for all roles (74.1 to 70.2). The ability of the CCG parser to recover these long-range dependencies accounts for its higher performance, and in particular its higher recall, on core arguments.

The CCG gold standard performance is below that of the Penn Treebank gold standard with traces. We believe this performance gap to be caused by the mismatches between the CCG analyses and the PropBank annotations described in Section 5.2. For the reasons described, the head words of the constituents that have PropBank roles are not necessarily the head words that stand in a predicate-argument relation in CCGbank. If two words do not stand in a predicate-argument relation, the CCG system takes recourse to the path feature. This feature is much sparser in CCG: since CCG categories encode subcategorization information, the number of categories in CCGbank is much larger than that of Penn Treebank labels. Analysis of our system's output shows that the system trained on the Penn Treebank gold standard obtains 55.5% recall on those relations that require the CCG path feature, whereas the system using CCGbank only achieves 36.9% recall on these. Also, in CCG, the complement-adjunct distinction is represented in the categories for the complement (eg. PP) or adjunct (eg. $\mathsf{(S\backslash NP)\backslash(S\backslash NP)}$ and in the categories for the head (eg. $\mathsf{(S[dcl]\backslash NP)/PP}$ or $\mathsf{S[dcl]\backslash NP}$). In generating the CCGbank, various heuristics were used to make this distinction. In particular, for PPs, it depends on the "closely-related" (CLR) function tag, which is known to be unreliable. The decisions made in deriving the CCGbank often do not match the hand-annotated complement-adjunct distinctions in PropBank, and this inconsistency is likely to make our CCGbank-based features less predictive. A possible solution is to regenerate the CCGbank using the Propbank annotations.

The impact of our head-word based scoring is analyzed in Table 3, which compares results when only the head word must be correctly identified (as in Table 2) and to results when both the beginning and end of the argument must be correctly identified in the sentence (as in Gildea and Palmer (2002)). Even if the head word is given the correct label, the boundaries of the entire argument may be different from those given in the PropBank annotation. Since constituents in CCGbank do not always match those in PropBank, even the CCG gold standard parses obtain comparatively low scores according to this metric. This is exacerbated when automatic parses are considered.

7 Conclusion

Our CCG-based system for automatically labeling verb arguments with PropBank-style semantic roles outperforms a system using a traditional Treebank-based parser for core arguments, which comprise 75% of the role labels, but scores lower on adjunct-like roles such as temporals and locatives. The CCG parser returns predicate-argument structures that include long-range dependencies; therefore, it seems inherently better suited for this task. However, the performance of our CCG system is lowered by the fact that the syntactic analyses in its training corpus differ from those that underlie PropBank in important ways (in particular in the notion of heads and the complement-adjunct distinction). We would expect a higher performance for the CCG-based system if the analyses in CCGbank resembled more closely those in PropBank.

Our results also indicate the importance of recovering long-range dependencies, either through the trace information in the Penn Treebank, or directly, as in the predicate-argument structures returned by the CCG parser. We speculate that much of the performance improvement we show could be obtained with traditional (ie. non-CCG-based) parsers if they were designed to recover more of the information present in the Penn Treebank, in particular the trace co-indexation. An interesting experiment would be the application of our role-labeling system to the output of the trace recovery system of Johnson (2002). Our results also have implications for parser evaluation, as the most frequently used constituent-based precision and recall measures do not evaluate how well long-range dependencies can be recovered from the output of a parser. Measures based on dependencies, such as those of Lin (1995) and Carroll et al. (1998), are likely to be more relevant to real-world applications of parsing.

Acknowledgments This work was supported by the Institute for Research in Cognitive Science at the University of Pennsylvania, the Propbank project (DoD Grant MDA904-00C-2136), an EPSRC studentship and grant GR/M96889, and NSF ITR grant 0205 456. We thank Mark Steedman, Martha Palmer and Alexandra Kinyon for their comments on this work.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING/ACL*, pages 86–90, Montreal.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain.

Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring Compact Lexicalized Grammars from a Cleaner Treebank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1974–1981, Las Palmas.

Julia Hockenmaier and Mark Steedman. 2002b. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA.

Julia Hockenmaier. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1420–1425, Montreal.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge Mass.