Brutus: A Semantic Role Labeling System Incorporating CCG, CFG, and Dependency Features

Stephen A. Boxwell, Dennis Mehay, and Chris Brew
Department of Linguistics
The Ohio State University
{boxwe11,mehay,cbrew}@1ing.ohio-state.edu

Abstract

We describe a semantic role labeling system that makes primary use of CCG-based features. Most previously developed systems are CFG-based and make extensive use of a treepath feature, which suffers from data sparsity due to its use of explicit tree configurations. CCG affords ways to augment treepath-based features to overcome these data sparsity issues. By adding features over CCG word-word dependencies and lexicalized verbal subcategorization frames ("supertags"), we can obtain an F-score that is substantially better than a previous CCG-based SRL system and competitive with the current state of the art. A manual error analysis reveals that parser errors account for many of the errors of our system. This analysis also suggests that simultaneous incremental parsing and semantic role labeling may lead to performance gains in both tasks.

1 Introduction

Semantic Role Labeling (SRL) is the process of assigning semantic roles to strings of words in a sentence according to their relationship to the semantic predicates expressed in the sentence. The task is difficult because the relationship between syntactic relations like "subject" and "object" do not always correspond to semantic relations like "agent" and "patient". An effective semantic role labeling system must recognize the differences between different configurations:

(a) [The man]$_{Arg0}$ opened [the door]$_{Arg1}$ [for him]$_{Arg3}$ [today]$_{ArgM-TMP}$.

(b) [The door]$_{Arg1}$ opened.

(c) [The door]$_{Arg1}$ was opened by [a man]$_{Arg0}$.

We use Propbank (Palmer et al., 2005), a corpus of newswire text annotated with verb predicate semantic role information that is widely used in the SRL literature (Màrquez et al., 2008). Rather than describe semantic roles in terms of "agent" or "patient", Propbank defines semantic roles on a verb-by-verb basis. For example, the verb *open* encodes the OPENER as Arg0, the OPENEE as Arg1, and the beneficiary of the OPENING action as Arg3. Propbank also defines a set of adjunct roles, denoted by the letter M instead of a number. For example, ArgM-TMP denotes a temporal role, like "today". By using verb-specific roles, Propbank avoids specific claims about parallels between the roles of different verbs.

We follow the approach in (Punyakanok et al., 2008) in framing the SRL problem as a two-stage pipeline: identification followed by labeling. During identification, every word in the sentence is labeled either as bearing some (as yet undetermined) semantic role or not . This is done for each verb. Next, during labeling, the precise verb-specific roles for each word are determined. In contrast to the approach in (Punyakanok et al., 2008), which tags constituents directly, we tag headwords and then associate them with a constituent, as in a previous CCG-based approach (Gildea and Hockenmaier, 2003). Another difference is our choice of parsers. Brutus uses the CCG parser of (Clark and Curran, 2007, henceforth the C&C parser), Charniak's parser (Charniak, 2001) for additional CFG-based features, and MALT parser (Nivre et al., 2007) for dependency features, while (Punyakanok et al., 2008) use results from an ensemble of parses from Charniak's Parser and a Collins parser (Collins, 2003; Bikel, 2004). Finally, the system described in (Punyakanok et al., 2008) uses a joint inference model to resolve discrepancies between multiple automatic parses. We do not employ a similar strategy due to the differing notions of constituency represented in our parsers (CCG having a much more fluid notion of constituency and the MALT parser using a different approach entirely).

For the identification and labeling steps, we train a maximum entropy classifier (Berger et al., 1996) over sections 02-21 of a version of the CCGbank corpus (Hockenmaier and Steedman, 2007) that has been augmented by projecting the Propbank semantic annotations (Boxwell and White, 2008). We evaluate our SRL system's argument predictions at the word string level, making our results directly comparable for each argument labeling.[1]

In the following, we briefly introduce the CCG grammatical formalism and motivate its use in SRL (Sections 2–3). Our main contribution is to demonstrate that CCG — arguably a more expressive and lin-

[1]This is guaranteed by our string-to-string mapping from the original Propbank to the CCGbank.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 37–45,
Suntec, Singapore, 2-7 August 2009.

guistically appealing syntactic framework than vanilla CFGs — is a viable basis for the SRL task. This is supported by our experimental results, the setup and details of which we give in Sections 4–10. In particular, using CCG enables us to map semantic roles directly onto verbal categories, an innovation of our approach that leads to performance gains (Section 7). We conclude with an error analysis (Section 11), which motivates our discussion of future research for computational semantics with CCG (Section 12).

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2000) is a grammatical framework that describes syntactic structure in terms of the combinatory potential of the lexical (word-level) items. Rather than using standard part-of-speech tags and grammatical rules, CCG encodes much of the combinatory potential of each word by assigning a syntactically informative category. For example, the verb *loves* has the category (s\np)/np, which could be read "the kind of word that would be a sentence if it could combine with a noun phrase on the right and a noun phrase on the left". Further, CCG has the advantage of a transparent interface between the way the words combine and their dependencies with other words. Word-word dependencies in the CCGbank are encoded using predicate-argument (PARG) relations. PARG relations are defined by the functor word, the argument word, the category of the functor word and which argument slot of the functor category is being filled. For example, in the sentence *John loves Mary* (figure 1), there are two slots on the verbal category to be filled by NP arguments. The first argument (the subject) fills slot 1. This can be encoded as <loves,john,(s\np)/np,1>, indicating the head of the functor, the head of the argument, the functor category and the argument slot. The second argument (the direct object) fills slot 2. This can be encoded as <loves,mary,(s\np)/np,2>. One of the potential advantages to using CCGbank-style PARG relations is that they uniformly encode both local and long-range dependencies — e.g., the noun phrase *the Mary that John loves* expresses the same set of two dependencies. We will show this to be a valuable tool for semantic role prediction.

3 Potential Advantages to using CCG

There are many potential advantages to using the CCG formalism in SRL. One is the uniformity with which CCG can express equivalence classes of local and long-range (including unbounded) dependencies. CFG-based approaches often rely on examining potentially long sequences of categories (or *treepaths*) between the verb and the target word. Because there are a number of different treepaths that correspond to a single relation (figure 2), this approach can suffer from data sparsity. CCG, however, can encode all treepath-distinct expressions of a single grammatical relation into a single predicate-argument relationship (figure 3). This feature has been shown (Gildea and Hockenmaier, 2003) to be an effective substitute for treepath-based features. But while predicate-argument-based features are very effective, they are still vulnerable both to parser errors and to cases where the semantics of a sentence do not correspond directly to syntactic dependencies. To counteract this, we use both kinds of features with the expectation that the treepath feature will provide low-level detail to compensate for missed, incorrect or syntactically impossible dependencies.

Another advantage of a CCG-based approach (and lexicalist approaches in general) is the ability to encode verb-specific argument mappings. An argument mapping is a link between the CCG category and the semantic roles that are likely to go with each of its arguments. The projection of argument mappings onto CCG verbal categories is explored in (Boxwell and White, 2008). We describe this feature in more detail in section 7.

4 Identification and Labeling Models

As in previous approaches to SRL, Brutus uses a two-stage pipeline of maximum entropy classifiers. In addition, we train an argument mapping classifier (described in more detail below) whose predictions are used as features for the labeling model. The same features are extracted for both treebank and automatic parses. Automatic parses were generated using the C&C CCG parser (Clark and Curran, 2007) with its derivation output format converted to resemble that of the CCGbank. This involved following the derivational bracketings of the C&C parser's output and reconstructing the backpointers to the lexical heads using an in-house implementation of the basic CCG combinatory operations. All classifiers were trained to 500 iterations of L-BFGS training — a quasi-Newton method from the numerical optimization literature (Liu and Nocedal, 1989) — using Zhang Le's maxent toolkit.[2] To prevent overfitting we used Gaussian priors with global variances of 1 and 5 for the identifier and labeler, respectively.[3] The Gaussian priors were determined empirically by testing on the development set.

Both the identifier and the labeler use the following features:

(1) **Words**. Words drawn from a 3 word window around the target word,[4] with each word associated with a binary indicator feature.

(2) **Part of Speech**. Part of Speech tags drawn from a 3 word window around the target word,

[2] Available for download at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`.

[3] Gaussian priors achieve a smoothing effect (to prevent overfitting) by penalizing very large feature weights.

[4] The size of the window was determined experimentally on the development set – we use the same window sizes throughout.

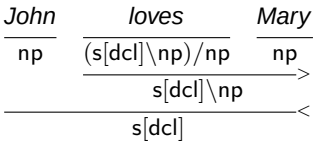

Figure 1: This sentence has two dependencies: <loves,mary,(s\np)/np,2> and <loves,john,(s\np)/np,1>

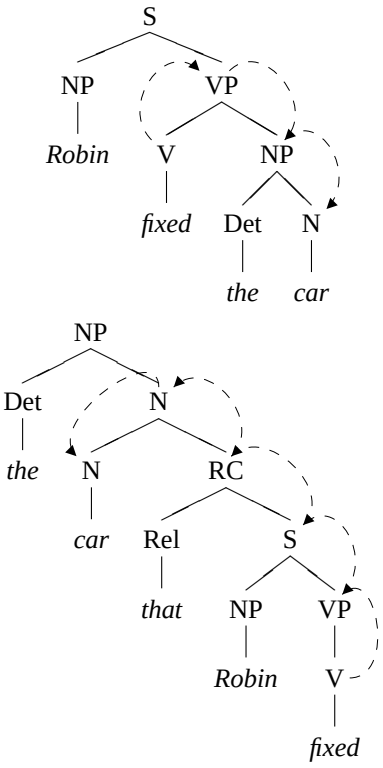

Figure 2: The semantic relation (Arg1) between 'car' and 'fixed' in both phrases is the same, but the treepaths — traced with arrows above — are different: (V>VP<NP<N and V>VP>S>RC>N<N, respectively).

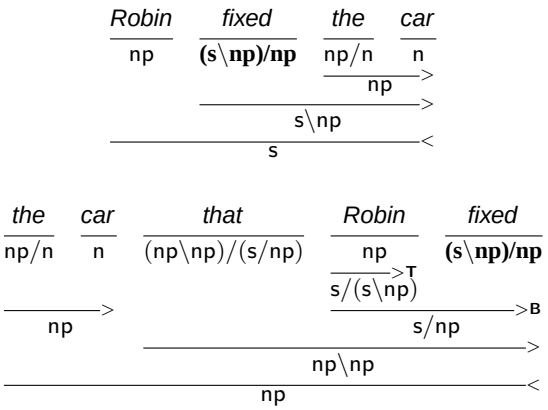

Figure 3: CCG word-word dependencies are passed up through subordinate clauses, encoding the relation between *car* and *fixed* the same in both cases: (s\np)/np.2.→ (Gildea and Hockenmaier, 2003)

with each associated with a binary indicator feature.

(3) **CCG Categories**. CCG categories drawn from a 3 word window around the target word, with each associated with a binary indicator feature.

(4) **Predicate**. The lemma of the predicate we are tagging. E.g. *fix* is the lemma of *fixed*.

(5) **Result Category Detail**. The grammatical feature on the category of the predicate (indicating declarative, passive, progressive, etc). This can be read off the verb category: declarative for *eats*: *(s[dcl]\np)/np* or progressive for *running*: *s[ng]\np*.

(6) **Before/After**. A binary indicator variable indicating whether the target word is before or after the verb.

(7) **Treepath**. The sequence of CCG categories representing the path through the derivation from the predicate to the target word. For the relationship between *fixed* and *car* in the first sentence of figure 3, the treepath is (s[dcl]\np)/np>s[dcl]\np<np<n, with > and < indicating movement up and down the tree, respectively.

(8) **Short Treepath**. Similar to the above treepath feature, except the path stops at the highest node under the least common subsumer that is headed by the target word (this is the *constituent* that the role would be marked on if we identified this terminal as a role-bearing word). Again, for the relationship between *fixed* and *car* in the first sentence of figure 3, the short treepath is (s[dcl]\np)/np>s[dcl]\np<np.

(9) **NP Modified**. A binary indicator feature indicating whether the target word is modified by an NP modifier.[5]

[5]This is easily read off of the CCG PARG relationships.

(10) **Subcategorization**. A sequence of the categories that the verb combines with in the CCG derivation tree. For the first sentence in figure 3, the correct subcategorization would be *np,np*. Notice that this is not necessarily a restatement of the verbal category – in the second sentence of figure 3, the correct subcategorization is *s/(s\np),(np\np)/(s[dcl]/np),np*.

(11) **PARG feature**. We follow a previous CCG-based approach (Gildea and Hockenmaier, 2003) in using a feature to describe the PARG relationship between the two words, if one exists. If there is a dependency in the PARG structure between the two words, then this feature is defined as the conjunction of (1) the category of the functor, (2) the argument slot that is being filled in the functor category, and (3) an indication as to whether the functor (→) or the argument (←) is the lexical head. For example, to indicate the relationship between *car* and *fixed* in both sentences of figure 3, the feature is (s\np)/np.2.→.

The labeler uses all of the previous features, plus the following:

(12) **Headship**. A binary indicator feature as to whether the functor or the argument is the lexical head of the dependency between the two words, if one exists.

(13) **Predicate and Before/After**. The conjunction of two earlier features: the predicate lemma and the Before/After feature.

(14) **Rel Clause**. Whether the path from predicate to target word passes through a relative clause (e.g., marked by the word *'that'* or any other word with a relativizer category).

(15) **PP features**. When the target word is a preposition, we define binary indicator features for the word, POS, and CCG category of the head of the topmost NP in the prepositional phrase headed by a preposition (a.k.a. the '*lexical head*' of the PP). So, if *on* heads the phrase *'on the third Friday'*, then we extract features relating to *Friday* for the preposition *on*. This is null when the target word is not a preposition.

(16) **Argument Mappings**. If there is a PARG relation between the predicate and the target word, the argument mapping is the most likely predicted role to go with that argument. These mappings are predicted using a separate classifier that is trained primarily on lexical information of the verb, its immediate string-level context, and its observed arguments in the training data. This feature is null when there is no PARG relation between the predicate and the target word. The Argument Mapping feature can be viewed as a simple prediction about some of the non-modifier semantic roles that a verb is likely to express. We use this information as a feature and not a hard constraint to allow other features to overrule the recommendation made by the argument mapping classifier. The features used in the argument mapping classifier are described in detail in section 7.

5 CFG based Features

In addition to CCG-based features, features can be drawn from a traditional CFG-style approach when they are available. Our motivation for this is twofold. First, others (Punyakanok et al., 2008, e.g.), have found that different parsers have different error patterns, and so using multiple parsers can yield complementary sources of correct information. Second, we noticed that, although the CCG-based system performed well on head word labeling, performance dropped when projecting these labels to the constituent level (see sections 8 and 9 for more). This may have to do with the fact that CCG is not centered around a constituency-based analysis, as well as with inconsistencies between CCG and Penn Treebank-style bracketings (the latter being what was annotated in the original Propbank).

Penn Treebank-derived features are used in the identifier, labeler, and argument mapping classifiers. For automatic parses, we use Charniak's parser (Charniak, 2001). For gold-standard parses, we remove functional tag and trace information from the Penn Treebank parses before we extract features over them, so as to simulate the conditions of an automatic parse. The Penn Treebank features are as follows:

(17) **CFG Treepath**. A sequence of traditional CFG-style categories representing the path from the verb to the target word.

(18) **CFG Short Treepath**. Analogous to the CCG-based short treepath feature.

(19) **CFG Subcategorization**. Analogous to the CCG-based subcategorization feature.

(20) **CFG Least Common Subsumer**. The category of the root of the smallest tree that dominates both the verb and the target word.

6 Dependency Parser Features

Finally, several features can be extracted from a dependency representation of the same sentence. Automatic dependency relations were produced by the MALT parser. We incorporate MALT into our collection of parses because it provides detailed information on the exact syntactic relations between word pairs (subject, object, adverb, etc) that is not found in other automatic parsers. The features used from the dependency parses are listed below:

(21) **DEP-Exists** A binary indicator feature showing whether or not there is a dependency between the target word and the predicate.

(22) **DEP-Type** If there is a dependency between the target word and the predicate, what type of dependency it is (SUBJ, OBJ, etc).

7 Argument Mapping Model

An innovation in our approach is to use a separate classifier to predict an argument mapping feature. An argument mapping is a mapping from the syntactic arguments of a verbal category to the semantic arguments that should correspond to them (Boxwell and White, 2008). In order to generate examples of the argument mapping for training purposes, it is necessary to employ the PARG relations for a given sentence to identify the headwords of each of the verbal arguments. That is, we use the PARG relations to identify the headwords of each of the constituents that are arguments of the verb. Next, the appropriate semantic role that corresponds to that headword (given by Propbank) is identified. This is done by climbing the CCG derivation tree towards the root until we find a semantic role corresponding to the verb in question — i.e., by finding the point where the constituent headed by the verbal category combines with the constituent headed by the argument in question. These semantic roles are then marked on the corresponding syntactic argument of the verb.

As an example, consider the sentence *The boy loves a girl.* (figure 4). By examining the arguments that the verbal category combines with in the treebank, we can identify the corresponding semantic role for each argument that is marked on the verbal category. We then use these tags to train the Argument Mapping model, which will predict likely argument mappings for verbal categories based on their local surroundings and the headwords of their arguments, similar to the supertagging approaches used to label the informative syntactic categories of the verbs (Bangalore and Joshi, 1999; Clark, 2002), except tagging "one level above" the syntax.

The Argument Mapping Predictor uses the following features:

(23) **Predicate**. The lemma of the predicate, as before.

(24) **Words**. Words drawn from a 5 word window around the target word, with each word associated with a binary indicator feature, as before.

(25) **Parts of Speech**. Part of Speech tags drawn from a 5 word window around the target word, with each tag associated with a binary indicator feature, as before.

(26) **CCG Categories**. CCG categories drawn from a 5 word window around the target word, with each category associated with a binary indicator feature, as before.

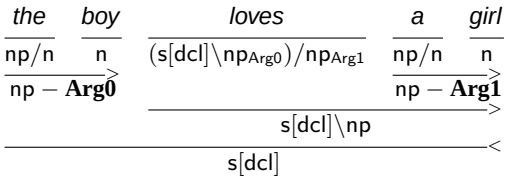

Figure 4: By looking at the constituents that the verb combines with, we can identify the semantic roles corresponding to the arguments marked on the verbal category.

(27) **Argument Data**. The word, POS, and CCG category, and treepath of the headwords of each of the verbal arguments (i.e., PARG dependents), each encoded as a separate binary indicator feature.

(28) **Number of arguments**. The number of arguments marked on the verb.

(29) **Words of Arguments**. The head words of each of the verb's arguments.

(30) **Subcategorization**. The CCG categories that combine with this verb. This includes syntactic adjuncts as well as arguments.

(31) **CFG-Sisters**. The POS categories of the sisters of this predicate in the CFG representation.

(32) **DEP-dependencies**. The individual dependency types of each of the dependencies relating to the verb (SBJ, OBJ, ADV, etc) taken from the dependency parse. We also incorporate a single feature representing the entire set of dependency types associated with this verb into a single feature, representing the set of dependencies as a whole.

Given these features with gold standard parses, our argument mapping model can predict entire argument mappings with an accuracy rate of 87.96% on the test set, and 87.70% on the development set. We found the features generated by this model to be very useful for semantic role prediction, as they enable us to make decisions about entire sets of semantic roles associated with individual lemmas, rather than choosing them independently of each other.

8 Enabling Cross-System Comparison

The Brutus system is designed to label headwords of semantic roles, rather than entire constituents. However, because most SRL systems are designed to label constituents rather than headwords, it is necessary to project the roles up the derivation to the correct constituent in order to make a meaningful comparison of the system's performance. This introduces the potential for further error, so we report results on the accuracy of headwords as well as the correct string of words. We deterministically move the role to the highest constituent in the derivation that is headed by the

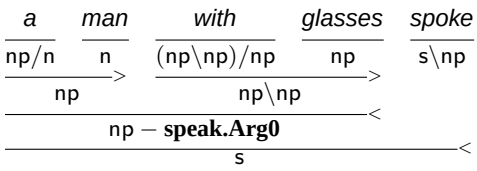

Figure 5: The role is moved towards the root until the original node is no longer the head of the marked constituent.

	P	R	F
G&H (treebank)	67.5%	60.0%	63.5%
Brutus (treebank)	88.18%	85.00%	86.56%
G&H (automatic)	55.7%	49.5%	52.4%
Brutus (automatic)	76.06%	70.15%	72.99%

Table 1: Accuracy of semantic role prediction using only CCG based features.

originally tagged terminal. In most cases, this corresponds to the node immediately dominated by the lowest common subsuming node of the the target word and the verb (figure 5). In some cases, the highest constituent that is headed by the target word is not immediately dominated by the lowest common subsuming node (figure 6).

9 Results

Using a version of Brutus incorporating only the CCG-based features described above, we achieve better results than a previous CCG based system (Gildea and Hockenmaier, 2003, henceforth G&H). This could be due to a number of factors, including the fact that our system employs a different CCG parser, uses a more complete mapping of the Propbank onto the CCGbank, uses a different machine learning approach,[6] and has a richer feature set. The results for constituent tagging accuracy are shown in table 1.

As expected, by incorporating Penn Treebank-based features and dependency features, we obtain better results than with the CCG-only system. The results for gold standard parses are comparable to the winning system of the CoNLL 2005 shared task on semantic role labeling (Punyakanok et al., 2008). Other systems (Toutanova et al., 2008; Surdeanu et al., 2007; Johansson and Nugues, 2008) have also achieved comparable results – we compare our system to (Punyakanok et al., 2008) due to the similarities in our approaches. The performance of the full system is shown in table 2.

Table 3 shows the ability of the system to predict the correct headwords of semantic roles. This is a necessary condition for correctness of the full constituent, but not a sufficient one. In parser evaluation, Carroll, Minnen, and Briscoe (Carroll et al., 2003) have argued

[6] G&H use a generative model with a back-off lattice, whereas we use a maximum entropy classifier.

	P	R	F
P. et al (treebank)	86.22%	87.40%	86.81%
Brutus (treebank)	88.29%	86.39%	87.33%
P. et al (automatic)	77.09%	75.51%	76.29%
Brutus (automatic)	76.73%	70.45%	73.45%

Table 2: Accuracy of semantic role prediction using CCG, CFG, and MALT based features.

	P	R	F
Headword (treebank)	88.94%	86.98%	87.95%
Boundary (treebank)	88.29%	86.39%	87.33%
Headword (automatic)	82.36%	75.97%	79.04%
Boundary (automatic)	76.33%	70.59%	73.35%

Table 3: Accuracy of the system for labeling semantic roles on both constituent boundaries and headwords. Headwords are easier to predict than boundaries, reflecting CCG's focus on word-word relations rather than constituency.

for dependencies as a more appropriate means of evaluation, reflecting the focus on headwords from constituent boundaries. We argue that, especially in the heavily lexicalized CCG framework, headword evaluation is more appropriate, reflecting the emphasis on headword combinatorics in the CCG formalism.

10 The Contribution of the New Features

Two features which are less frequently used in SRL research play a major role in the Brutus system: The PARG feature (Gildea and Hockenmaier, 2003) and the argument mapping feature. Removing them has a strong effect on accuracy when labeling treebank parses, as shown in our feature ablation results in table 4. We do not report results including the Argument Mapping feature but not the PARG feature, because some predicate-argument relation information is assumed in generating the Argument Mapping feature.

	P	R	F
+PARG +AM	88.77%	86.15%	87.44%
+PARG -AM	88.42%	85.78%	87.08%
-PARG -AM	87.92%	84.65%	86.26%

Table 4: The effects of removing key features from the system on gold standard parses.

The same is true for automatic parses, as shown in table 5.

11 Error Analysis

Many of the errors made by the Brutus system can be traced directly to erroneous parses, either in the automatic or treebank parse. In some cases, PP attachment

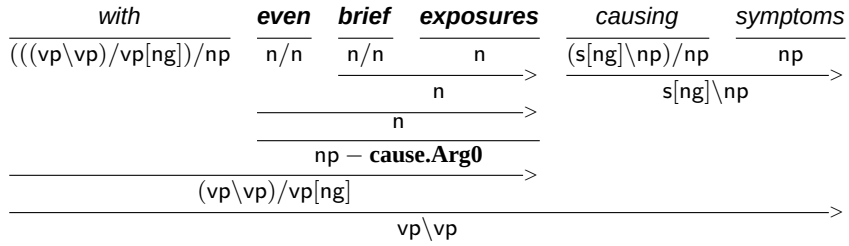

Figure 6: In this case, *with* is the head of *with even brief exposures*, so the role is correctly marked on *even brief exposures* (based on wsj_0003.2).

	P	R	F
+PARG +AM	74.14%	62.09%	67.58%
+PARG -AM	70.02%	64.68%	67.25%
-PARG -AM	73.90%	61.15%	66.93%

Table 5: The effects of removing key features from the system on automatic parses.

ambiguities cause a role to be marked too high in the derivation. In the sentence *the company stopped using asbestos in 1956* (figure 7), the correct Arg1 of *stopped* is *using asbestos*. However, because *in 1956* is erroneously modifying the verb *using* rather than the verb *stopped* in the treebank parse, the system trusts the syntactic analysis and places Arg1 of *stopped* on *using asbestos in 1956*. This particular problem is caused by an annotation error in the original Penn Treebank that was carried through in the conversion to CCGbank.

Another common error deals with genitive constructions. Consider the phrase *a form of asbestos used to make filters*. By CCG combinatorics, the relative clause could either attach to *asbestos* or to *a form of asbestos*. The gold standard CCG parse attaches the relative clause to *a form of asbestos* (figure 8). Propbank agrees with this analysis, assigning Arg1 of *use* to the constituent *a form of asbestos*. The automatic parser, however, attaches the relative clause low – to *asbestos* (figure 9). When the system is given the automatically generated parse, it incorrectly assigns the semantic role to *asbestos*. In cases where the parser attaches the relative clause correctly, the system is much more likely to assign the role correctly.

Problems with relative clause attachment to genitives are not limited to automatic parses – errors in gold-standard treebank parses cause similar problems when Treebank parses disagree with Propbank annotator intuitions. In the phrase *a group of workers exposed to asbestos* (figure 10), the gold standard CCG parse attaches the relative clause to *workers*. Propbank, however, annotates *a group of workers* as Arg1 of *exposed*, rather than following the parse and assigning the role only to *workers*. The system again follows the parse and incorrectly assigns the role to *workers* instead of *a group of workers*. Interestingly, the C&C parser opts for high attachment in this instance, resulting in the correct prediction of *a group of workers* as Arg1 of *exposed* in the automatic parse.

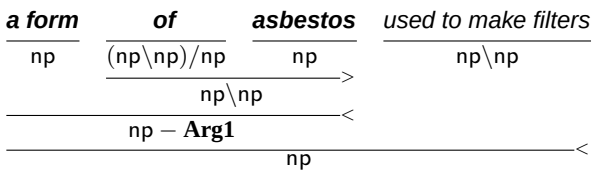

Figure 8: CCGbank gold-standard parse of a relative clause attachment. The system correctly identifies *a form of asbestos* as Arg1 of *used*. (wsj_0003.1)

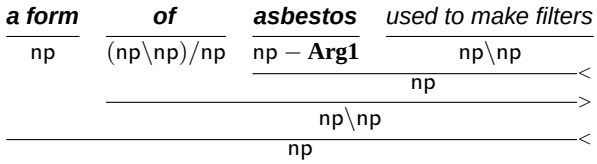

Figure 9: Automatic parse of the noun phrase in figure 8. Incorrect relative clause attachment causes the misidentification of *asbestos* as a semantic role bearing unit. (wsj_0003.1)

12 Future Work

As described in the error analysis section, a large number of errors in the system are attributable to errors in the CCG derivation, either in the gold standard or in automatically generated parses. Potential future work may focus on developing an improved CCG parser using the revised (syntactic) adjunct-argument distinctions (guided by the Propbank annotation) described in (Boxwell and White, 2008). This resource, together with the reasonable accuracy ($\approx$ 90%) with which argument mappings can be predicted, suggests the possibility of an integrated, simultaneous syntactic-semantic parsing process, similar to that of (Musillo and Merlo, 2006; Merlo and Musillo, 2008). We expect this would improve the reliability and accuracy of both the syntactic and semantic analysis components.

13 Acknowledgments

This research was funded by NSF grant IIS-0347799. We are deeply indebted to Julia Hockenmaier for the

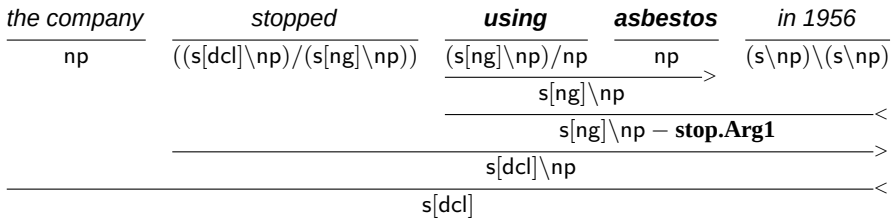

Figure 7: An example of how incorrect PP attachment can cause an incorrect labeling. Stop.Arg1 should cover *using asbestos* rather than *using asbestos in 1956*. This sentence is based on wsj_0003.3, with the structure simplified for clarity.

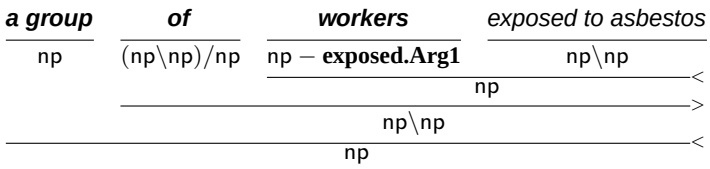

Figure 10: Propbank annotates *a group of workers* as Arg1 of *exposed*, while CCGbank attaches the relative clause low. The system incorrectly labels *workers* as a role bearing unit. (Gold standard – wsj_0003.1)

use of her PARG generation tool.

References

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Adam L. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

D.M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Stephen A. Boxwell and Michael White. 2008. Projecting propbank roles onto the ccgbank. In *Proceedings of the Sixth International Language Resources and Evaluation Conference (LREC-08)*, Marrakech, Morocco.

J. Carroll, G. Minnen, and T. Briscoe. 2003. Parser evaluation. *Treebanks: Building and Using Parsed Corpora*, pages 299–316.

E. Charniak. 2001. Immediate-head parsing for language models. In *Proc. ACL-01*, volume 39, pages 116–123.

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

Stephen Clark. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, pages 19–24, Venice, Italy.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *Proc. EMNLP-03*.

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.

R. Johansson and P. Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. *Proceedings of CoNLL–2008*.

D C Liu and Jorge Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3).

Lluís Màrquez, Xavier Carreras, Kenneth C. Litowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

Paola Merlo and Gabrile Musillo. 2008. Semantic parsing for high-precision semantic role labelling. In *Proceedings of CONLL-08*, Manchester, UK.

Gabriele Musillo and Paola Merlo. 2006. Robust parsing of the proposition bank. In *Proceedings of the EACL 2006 Workshop ROMAND*, Trento.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257–287.

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

M. Surdeanu, L. Màrquez, X. Carreras, and P. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

K. Toutanova, A. Haghighi, and C.D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.