Semantic Role Labeling Without Treebanks?

Stephen A. Boxwell[1], **Chris Brew**[2], **Jason Baldridge**[3], **Dennis Mehay**[1], and **Sujith Ravi**[4]

[1]The Ohio State University, {boxwell,mehay}@ling.ohio-state.edu
[2]The Educational Testing Service, cbrew@ets.org
[3]The University of Texas at Austin, jbaldrid@mail.utexas.edu
[4]ISI, sravi@isi.edu

Abstract

We describe a method for training a semantic role labeler for CCG in the absence of gold-standard syntax derivations. Traditionally, semantic role labeling is performed by placing human-annotated semantic roles on gold-standard syntactic parses, identifying patterns in the syntax-semantics relationship, and then predicting roles on novel syntactic analyses. The gold standard syntactic training data can be eliminated from the process by extracting training instances from semantic roles projected onto a packed parse chart. This process can be used to rapidly develop NLP tools for resource-poor languages of interest.

1 Introduction

Semantic role labeling is the process of generating sets of semantic roles from syntactic analyses. The process of training a semantic role labeler, however, is costly in resources. First, it requires gold-standard semantic role data, like Propbank (Palmer et al., 2005). Secondly, it requires a detailed syntactic annotation of the same resource. We are fortunate to have the reasonably-sized Penn Treebank (Marcus et al., 1993) and adaptations for formalisms like Tree Adjoining Grammar (Chen and Shanker, 2004) and Combinatory Categorial Grammar (Hockenmaier and Steedman, 2007) alongside the Propbank data, but for other languages, such resources are unlikely to be available. There has been work in generating semantic role labelers using gold-standard trees in the absence of semantic training data (Fürstenau and Lapata, 2009; Lang and Lapata, 2010). But what if we had semantic training data, but no syntactic training data? If we could develop and train a semantic role labeler without syntactic training data, we could greatly reduce the cost and development time of NLP tools for languages of interest.

One option is to use some automatic means to generate a treebank – instead of creating a corpus of syntax trees by hand, we could use an automatic parser. This, however, leads to a chicken-and-egg problem – we would need a high-quality parse model to choose a single-best analysis for each training sentence, and a parse model needs syntactic training data. No automatic parser can currently generate high quality single-best parses in the absence of a parse model. But a parser can, given word tags and combinatory rules, generate a parse forest – a very large collection of possible analyses – and say little or nothing about their relative merit.

In this paper, we extract SRL features from the entire parse forest, effectively training on every possible parse in the training set simultaneously. This can be done efficiently by representing the parse chart as a hypergraph, enabling us to iterate over every constituent in the parse forest without enumerating every individual parse (which would be computationally infeasible). This, combined with the parsing advantages afforded by Combinatory Categorial Grammar (CCG), enables us to train a semantic role labeler without gold-standard trees.

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2000) is a grammar formalism that describes words in terms of their combinatory potential. For example, determiners belong to the category NP/N, or "the category of words that become noun

phrases when combined with a noun to the right". The rightmost category indicates the argument that the category is seeking, the leftmost category indicates the result of combining this category with its argument, and the slash indicates the direction of combination. Categories can be nested within each other: a transitive verb like *devoured* belongs to the category (S\NP)/NP, or "the category that would become a sentence if it could combine with a noun phrase to the right and another noun phrase to the left". The process of automatically assigning CCG categories to words is called "supertagging", and CCG categories are sometimes informally referred to as "supertags". An example of how categories combine to make sentences is shown in Figure 1.

CCG has many capabilities that go beyond that of a typical context-free grammar. First, it has a sophisticated internal system of managing syntactic heads and dependencies[1]. These dependencies are used to great effect in CCG-based semantic role labeling systems (Gildea and Hockenmaier, 2003; Boxwell et al., 2009), as they do not suffer the same data-sparsity effects encountered with treepath features in CFG-based SRL systems. Secondly, CCG permits these dependencies to be passed through intermediary categories in grammatical structures like relative clauses. In Figure 2, *the steak* is still in the object relation to *devoured*, even though the verb is inside a relative clause. Finally and most importantly, *these dependencies are represented directly on the CCG categories themselves*. This is crucial for the prediction of semantic roles inside a packed parse chart – because the dependency is formed when the two heads combine, it is available to be used as a local feature by the semantic role labeler. This property of CCG and its impact on packed-chart SRL is described extensively in Boxwell et al. (2010). This ability to predict dependencies (and semantic roles) at parse time figures heavily into the process described here.

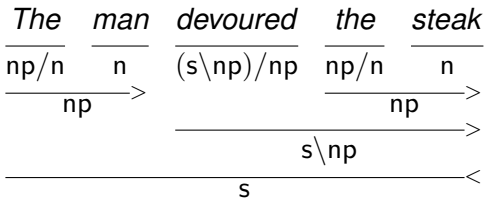

Figure 1: A simple CCG derivation.

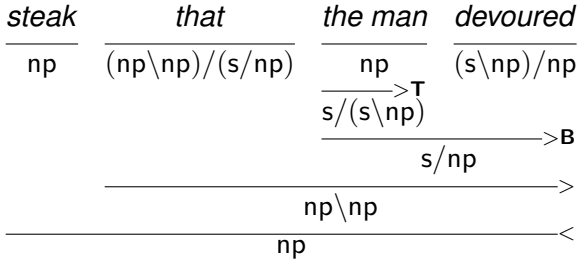

Figure 2: An example of CCG's treatment of relative clauses. The syntactic dependency between *devoured* and *steak* is the same as it was in figure 1.

3 Brutus: A CCG Based Semantic Role Labeler

The Brutus Semantic Role Labeler (Boxwell et al., 2009)[2] is a semantic role labeling system for CCG. It is trained using CCGbank and a version of Propbank that has been aligned to the CCGbank in order to account for discrepancies in terminal indexation (Honnibal and Curran, 2007; Boxwell and White, 2008). The system is organized in a two-stage pipeline of maximum entropy models[3], following the organization of a previous CFG-style approach (Punyakanok et al., 2008). The first stage is the identification stage, where, for each predicate in the sentence, each word is tagged as either a role or a nonrole (figure 3). The second stage is the classification stage, where the roles are sorted into ARG0, ARG1, and so on (figure 4). The identification model and the classification model share the same features, but they are trained and run separately.

For the results presented here, we use a version of Brutus that has been stripped down to only use local features so as to enable us to perform SRL at parse time. Recall from section 1 that we wish to extract training features not from a complete parse tree, but from a packed parse chart. For this reason, global features (those that are inaccessible to a single edge in the parse chart) cannot be used. After removing all global features from the seman-

[1] A complete explanation of CCG predicate-argument dependencies can be found in the CCGbank user manual (Hockenmaier and Steedman, 2005)

[2] Found at `http://www.ling.ohio-state.edu/~boxwell/software/brutus.html`

[3] We use the Zhang Le maxent toolkit, available at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`, using the BFGS training method, trained to 500 iterations with gaussian priors of 1 and 5, for the identification and classification steps, respectively.

tic role labeler, the local features that remain are as follows:

- **Words.** A three-word window surrounding the candidate word.
- **Predicate.** The predicate whose semantic roles the system is looking for.
- **Predicate Category.** The CCG category of the predicate.
- **Result Category Detail.** This indicates the feature on the result category of the predicate. Possible values include DCL (for declarative sentences), PSS (for passive sentences), NG (for present-progressive phrases like "running the race"), etc. These are read trivially off of the verbal category.
- **Syntactic Dependency.** As with a previous approach in CCG semantic role labeling (Gildea and Hockenmaier, 2003), this feature shows the exact nature of the syntactic dependency between the predicate and the word we are considering, if any such dependency exists. This feature is represented by the category of the predicate, the argument slot that this word fits into, and whether or not the predicate is the head of the resultant category, represented with a left or right arrow.
- **Before / After.** A binary indicator feature indicating whether the candidate word is before or after the predicate.

4 Parsing Without Syntactic Training Data

In order to test the performance of our semantic role labeler, we will need automatically generated parses to run the SRL models over. Even though we are able to train SRL models in the absence of syntactic training data, we still need test parses on which to predict roles. So why not use the fast, accurate CCG parser (Clark and Curran, 2004b) used with previous CCG-based SRL systems? It makes sense to use the highest quality parses available. But recall that the reason for this roundabout way of training the semantic role labeler is to enable us to generate SRL models *without syntactic training data*. If we use an off-the-shelf syntactic parser that was trained on gold-standard training data, we introduce a source of additional training data that we wish to exclude. But how will we generate reasonably accurate parses without a trained parse model? Even a simple MLE-style approach requires training data.

Combinator	Penalty
Function Application	0
Function Composition	1
Crossing Composition	1
Type Raising	1
Null Coordination	2
Full Coordination	0
Substitution	∞

Table 1: The complete sub-baseline model, which requires no syntactic training data. The substitution combinator is used to model parasitic gaps in English, which are so rare that we make the pragmatic decision to disallow substitution entirely.

To satisfy this need, we develop a very simple parse model that penalizes any non-normal-form rule applications, effectively relying on the CCG supertags to identify likely grammatical relations. Specifically, combinators like function composition and type raising are penalized by a fixed amount, while function application is allowed to pass without penalty. The candidate analysis with the lowest penalty is chosen as the single-best – in case of a tie, the most right-branching analysis is chosen. The complete parse model is shown in table 1.

5 Experiment 1: Generating Traditional Identification and Classification Models from the Chart

In the first experiment, we use a parser to create a set of parse forests from the training set. The individual parses are not enumerated – we extract features from every possible syntactic derivation simultaneously by iterating over every edge in the packed chart. Local syntactic features are accessible, as are the gold-standard semantic roles from Propbank. The identifier and classifier models are then trained from these features, instead of from features obtained from gold-standard syntactic derivations. We will call this two-part SRL model the CHART model. We compare this model to the more traditional GOLD model, which uses the same features but is generated from gold standard trees. We test the system using both gold-standard parse trees and single-best auto-

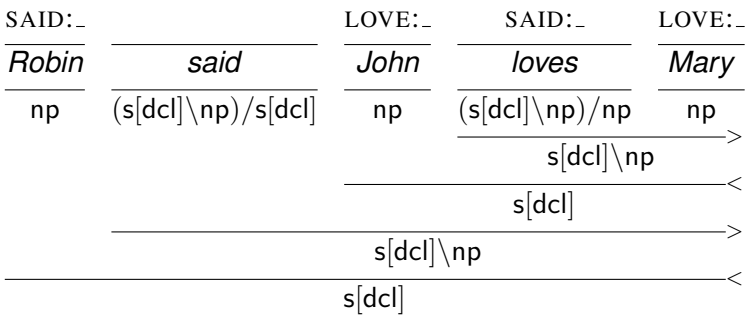

Figure 3: In the first stage of the semantic role labeling process, candidate semantic roles are chosen by the identifier model. We have not yet decided which role (ARG0, ARG1, etc) each word plays, only that there is a role there.

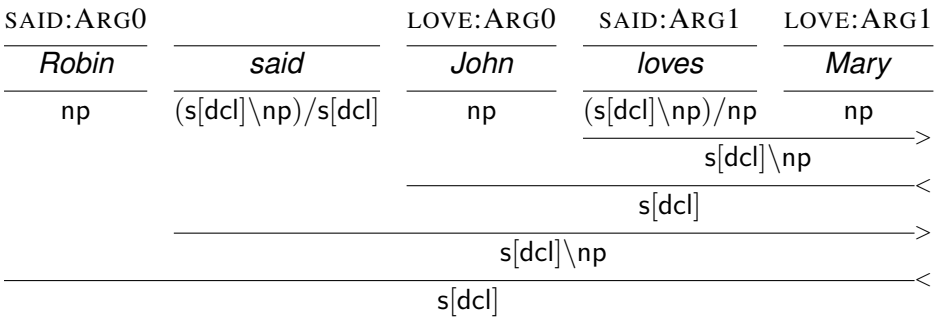

Figure 4: In the second stage of the semantic role labeling process, the classifier model sorts the roles into ARG0, ARG1, etc.

matically generated parse trees (generated from gold-standard supertags by the parser from section 4). Interestingly, SRL performance drops only slightly between gold standard test parses and automatically generated parses when using the chart-based SRL model. Table 2 shows the results for the development set, and table 3 shows the results for the test set.

	Gold Parse			Auto Parse		
Train	P	R	F	P	R	F
GOLD	88.4	85.7	87.0	84.8	80.4	82.5
CHART	83.5	70.8	76.6	83.0	69.4	75.6

Table 2: SRL performance on gold-standard parses and automatic parses from the development set (section 00). The models are defined in section 5.

Manual inspection of the results reveals that the CHART model frequently fails to identify modifier roles, contributing to the very low recall score. This was traced to a consistent weakening of the adjunct dependency feature, resulting largely from the ambiguous attachment of auxiliary verbs. Consider a simple sentence *Jon will visit tomorrow*. Syntactically, there are two possible attachments for *tomorrow*. It can be attached low, to *visit* (figure 5), or it can be attached high, to *will visit* (figure 6). The former will result in a dependency between *visit* and *tomorrow*, while the latter will result in a dependency between *will* and *tomorrow*. Now, imagine training over this sentence's chart. For both analyses, we notice that a role should be placed on *tomorrow*. In one case, there is a dependency between *visit* and *tomorrow*, and in one case there is not. Our simple parsing model does not necessarily do a good job of discriminating in favor of the analysis that we want, so the SRL components may see both options with nearly equal weight. Empirically, the identification model learns that the dependency feature is not a good predictor of modifier roles. This is in-

	Gold Parse			Auto Parse		
Train	P	R	F	P	R	F
GOLD	89.7	84.8	87.2	85.8	80.0	82.8
CHART	84.6	70.4	76.9	83.0	67.7	74.6

Table 3: SRL performance on the test set (section 23) using the same models as table 2.

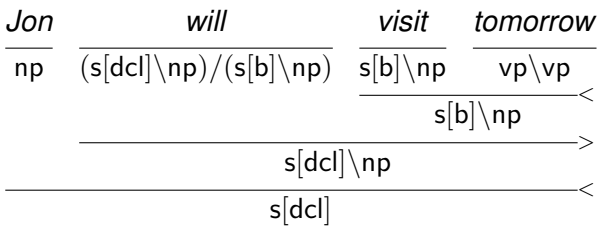

Figure 5: The correct analysis for *Jon will visit tomorrow*. In this case, there is a syntactic dependency between *tomorrow* and *visit*. Note that vp is an abbreviation for s\np.

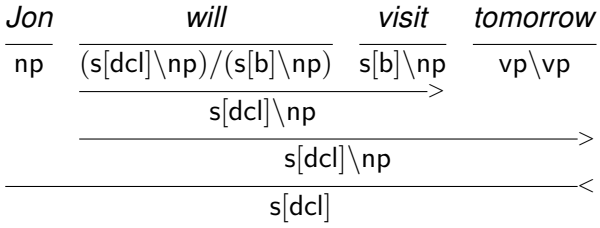

Figure 6: An erroneous analysis for *Jon will visit tomorrow*. There is no syntactic dependency between *tomorrow* and *visit*.

correct – in fact, the presence of a syntactic dependency between a predicate and a target word is almost always a dead giveaway to the presence of a semantic role. In our effort to downplay the role of syntax we may have set up a situation in which a sophisticated machine-learning based argument identifier does exactly the wrong thing. It could be that a less sophisticated argument identifier will be better suited to the task that our system requires.

6 Experiment 2: Improving Argument Identification with a Simpler Model

The CHART identification model performs poorly because it does not recognize syntactic dependencies as good predictors of semantic roles. Suppose that instead of using that identification model, we used a simple heuristic: if there is a syntactic dependency between a word and the predicate, then label that word with a semantic role – otherwise, do not. This simple identification "model" requires no training – it simply relies on the pattern that semantic role bearing units tend to be joined to their predicates by syntactic dependencies. We will refer to this as the chart-dependency model, or C-DEP. In addition to this model, we propose another model that similarly identifies roles with dependencies, but enumerates certain exceptional dependencies that do not predict semantic roles (like those originating from auxiliary verbs like *to* and *has*) according to the Propbank guidelines. We will refer to this as the improved chart-dependency model, or C-DEP+[4]. In both cases, the classification model is identical to that of the CHART model.

Tables 4 and 5 show the effect of using the two chart-dependency identification models compared to the GOLD and CHART models from section 5. Performance using the C-DEP and C-DEP+ models greatly improves on the disappointing recall of the CHART model, while retaining the favorable property of eschewing gold-standard syntactic training data. Instead of systematically weakening the most predictive identification feature available, the simple identification models use *only* that feature. This results in a major improvement in recall at the cost of an acceptable drop in precision.

	Gold Parse			Auto Parse		
Train	P	R	F	P	R	F
GOLD	88.4	85.7	87.0	84.8	80.4	82.5
CHART	83.5	70.8	76.6	83.0	69.4	75.6
C-DEP	**75.9**	**83.0**	**79.3**	**72.5**	**78.1**	**75.1**
C-DEP+	**81.6**	**82.8**	**82.2**	**77.9**	**77.8**	**77.9**

Table 4: SRL performance on the development set (section 00) using the four models. The GOLD and CHART models are defined in section 5. The C-DEP and C-DEP+ models are defined in section 6.

	Gold Parse			Auto Parse		
Train	P	R	F	P	R	F
GOLD	89.7	84.8	87.2	85.8	80.0	82.8
CHART	84.6	70.4	76.9	83.0	67.7	74.6
C-DEP	**76.7**	**83.2**	**79.8**	**73.0**	**77.7**	**75.2**
C-DEP+	**82.8**	**82.9**	**82.8**	**78.7**	**77.4**	**78.1**

Table 5: SRL performance on the test set (section 23), using the same models as in table 4.

[4]The C-DEP+ model ignores all dependencies originating from the following categories: (s[to]\np)/(s[b]\np), (s[dcl]\np)/(s[pt]\np), (s[dcl]\np)/(s[pss]\np), (s[b]\np)/(s[pss]\np), and (s[dcl]\np)/(s[ng]\np).

7 Experiment 3: Generating an SRL Model Without Gold-Standard Supertags

In the experiments described in sections 5 and 6, we used four different training methods to generate semantic role labeling models, which are then tested on gold standard syntactic parses and parses that were automatically generated from gold-standard supertags. But much of the challenge of parsing in CCG comes down to the choice of supertags – choosing the correct supertag for a preposition, for example, makes the difference between attaching the prepositional phrase high or low. But surely using gold-standard supertags gives us an unfair advantage; in real-world applications, these supertags would have to be predicted. We could use an off-the-shelf CCG supertagger (Clark and Curran, 2004a), but this would open us to the same chicken-and-egg problem already encountered with automatic parsers – the C&C supertagger is trained on gold-standard syntactic data. Doing without a supertagger and using every possible supertag in a tag dictionary is computationally infeasible; most words have at least two or three possible tags; the most ambiguous word has 133 supertags (*as*). Therefore, we have no choice but to investigate ways to get supertags that do not rely on gold-standard syntactic annotation.

We use a weakly supervised approach to supertagging that augments an HMM with an oracle CCG tag dictionary and a set of broad grammar-informed constraints (Baldridge, 2008; Ravi et al., 2010). The tag dictionary provides only a simple mapping from word to supertag – it does not use any kind of cutoff, nor does it give a prior probability on individual supertags. Using an HMM that has been initialized with grammar-based transition probabilities, combined with a two-stage integer programming strategy, this approach can achieve single-best accuracy of 64.3% on ambiguous supertags. These supertags are then used to generate parse forests, which are used to train the CHART, C-DEP, and C-DEP+ models. Notice that, although most of the tag sequences do not produce spanning analyses, we can still produce a packed chart and generate SRL training features.

We also train a secondary discriminative supertagger using the induced tags that are the output of the tag-dictionary-based HMM supertagger for the training set, and use this supertagger with the parser from section 4 to generate single-best parses to test the SRL models on. It is necessary to train a secondary supertagger over the induced tags because the induced tags by themseleves are unlikely to produce a spanning analysis. The induced supertags from the HMM are only given for the most probable sequence from the HMM; using the beta-best tag predictions of the secondary supertagger produces acceptable coverage. This supertagger is a simple Maxent tagger conditioned on a 5-word window surrounding the target word and trained using a gaussian prior of 5. SRL performance over automatic parses generated with these predicted supertags is not as strong as with gold standard supertags, but is reasonable considering the absence of syntactic training data. Results for the development set are shown in table 6, and results for the test set are shown in table 7.

	Gold Supertags Auto Parse			Auto Supertags Auto Parse		
Train	P	R	F	P	R	F
CHART	83.0	69.4	75.6	64.1	60.0	61.9
C-DEP	72.5	78.1	75.1	65.5	60.5	62.9
C-DEP+	77.9	77.8	77.9	68.8	60.2	64.2

Table 6: SRL performance on the development set (section 00) comparing automatic parses generated using gold-standard supertags and automatically induced supertags.

	Gold Supertags Auto Parse			Auto Supertags Auto Parse		
Train	P	R	F	P	R	F
CHART	83.0	67.7	74.6	63.8	61.7	62.7
C-DEP	73.0	77.7	75.2	66.8	61.1	63.8
C-DEP+	78.7	77.4	78.1	70.0	60.7	65.0

Table 7: SRL performance on the test set (section 23), using the same models as table 7.

Manual inspection of the induced supertag data reveals some unusual predictions of supertags. For example, it is difficult to think of a valid category for the determiner *the* besides NP/N. The word *the* almost always has the category NP/N in CCGbank. CCGbank does assign other categories for *the*, though most, if not all, of them are errors. Table 8 shows the token frequencies of select cate-

gories for *the* from the training set of CCGbank. Even though there are 46 possible categories in all, only one of them is really worth considering (18 of the categories appear only once). The automatic method for inducing supertags from the tag dictionary, however, frequently predicts categories for *the* that are extremely rare in the English CCGbank. This is because the tag dictionary is generated with no cutoff and provides no prior probability across tags – each tag in the dictionary is given equal consideration by the Markov Model, which ranks them according to how well they interact with their neighbors.

For this reason, we revisit our earlier decision to generate a tag dictionary with no cutoff. Instead, we generate a tag dictionary of categories that make up at least 10% of the word tokens. For example, suppose the word *direct* appears in the corpus 100 times. For a category to be listed for the word *direct* in the tag dictionary, it must appear as the category for *direct* no fewer than 10 times. This can effectively eliminate a large number of very rare categories that overwhelm the HMM. It also more closely simulates a hand-written tag dictionary for closed-class words, or a tag dictionary that was generated automatically from a traditional part-of-speech dictionary. Using a tag dictionary with a 10% cutoff greatly improves performance on semantc role labeling, coming to within 7% accuracy of using gold-standard supertags. The results for the development set are shown in table 9, and the results for the test set are shown in table 10.

Category	Frequency
NP/N	47255
N/N	99
((S\NP)\(S\NP))/NP	78
⋮	⋮
(S/S)/(S/S)	1
(S[adj]\NP)/N	1
(N\N)/N	1

Table 8: The frequencies of select categories for *the* from sections 02-21 of the CCGbank (there are 46 in all). Some categories, like NP/N, are extremely common, whereas others, like (N\N)/N, appear only once.

We have shown that simple and easy syntactic processing is still beneficial for SRL.

	Gold Supertags Auto Parse			Auto Supertags Auto Parse		
Train	P	R	F	P	R	F
CHART	83.0	69.4	75.6	67.5	66.3	66.9
C-DEP	72.5	78.1	75.1	69.3	69.5	69.4
C-DEP+	77.9	77.8	77.9	74.0	69.2	71.5

Table 9: SRL performance on the development set (section 00) using cutoff of 10% on tag dictionary.

	Gold Supertags Auto Parse			Auto Supertags Auto Parse		
Train	P	R	F	P	R	F
CHART	83.0	67.7	74.6	67.8	65.7	66.7
C-DEP	73.0	77.7	75.2	70.1	68.4	69.2
C-DEP+	78.7	77.4	78.1	75.2	68.2	71.5

Table 10: SRL performance on the test set (section 23) using cutoff of 10% on tag dictionary and the same models as table 9.

For completeness, we briefly explore another option, even simpler than this: we trained SRL models that relied on no syntactic features at all. Specifically, we included the word, predicate, and before/after features (described in detail in section 3). Unsurprisingly, the performance was unacceptably low (P=.73, R-.31, F=.44), most of that coming from successful identification of the predicate itself. This method makes the identifier exceptionally timid, and on the rare occasion that a word *is* identified as a role-bearing unit, it is often assigned roles corresponding to every predicate in the sentence. We conclude that it is necessary to include syntactic features, but that these can be rough and ready.

8 Conclusions and Future Work

In the three experiments presented, we demonstrated that an effective SRL model can be trained without a corpus of parse trees. This can be achieved by using a simple baseline parser to generate a parse forest for a large amount of unannotated newspaper text, then extracting training instances from all possible syntactic analyses simultaneously. This approach is most effective when we have some syntactic knowledge of the sentence in the form of supertags, but is still effective when only a tagging dictionary is available.

In the future, we hope to expand this work into

other languages. Armed only with a Propbank-like corpus of semantic roles and a tag dictionary, we can train a surprisingly effective semantic role labeler. To this end, we hope to further investigate issues surrounding the generation of supertags – particularly, minimally supervised approaches to generating CCG tag dictionaries. Recall that performance actually improved when very rare categories were excluded from the tag dictionary; one option to achieve similar results is to annotate by hand, say, the 200 most common words (almost certainly syntactically interesting closed class words), then using these to guide the generation of a comprehensive tag dictionary, or perhaps by bootstrapping from traditional part-of-speech tags. It would also be beneficial to investigate alternate methods of inducing tags from the tag dictionary that produce n-best tag predictions, as this would improve coverage over the training set.

Another avenue of future research could be the generation of semantic predictions without committing single-best test sentences. Recall that in order to test the semantic role labeler, we needed to generate parse trees for the target sentences. Because we assume that gold-standard syntactic training data is not available, we use a sub-baseline model that requires no training data. But is it really necessary to choose a single best parse at all? Because the version of Brutus used here can extract features from inside the chart, it can also predict semantic roles at parse time (Boxwell et al., 2010). We could therefore predict all possible roles in the chart and explore ways of identifying likely rolesets, using a mechanism for the enforcement of global constraints, such as the integer linear programming solution of Punyakanok et al (2008).

References

J. Baldridge. 2008. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 57–64. Association for Computational Linguistics.

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.

Stephen A. Boxwell, Dennis N. Mehay, and Chris Brew. 2009. Brutus: A semantic role labeling system incorporating CCG, CFG, and Dependency features. In *Proc. ACL-09*.

Stephen A Boxwell, Dennis N Mehay, and Chris Brew. 2010. What a parser can learn from a semantic role labeler and vice versa. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 736–744, Cambridge, MA, October. Association for Computational Linguistics.

J. Chen and V. Shanker. 2004. Automated extraction of TAGs from the Penn Treebank. *New developments in parsing technology*, pages 73–89.

S. Clark and J.R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING*, volume 4, pages 282–288.

Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and Log-Linear Models. In *Proc. ACL-04*.

H. Fürstenau and M. Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–228. Association for Computational Linguistics.

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

J. Hockenmaier and M. Steedman. 2005. CCGbank manual. Technical report, MS-CIS-05-09, University of Pennsylvania.

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.

M. Honnibal and J.R. Curran. 2007. Improving the complement/adjunct distinction in CCGbank. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-07)*, pages 210–217. Citeseer.

J. Lang and M. Lapata. 2010. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947. Association for Computational Linguistics.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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.

S. Ravi, J. Baldridge, and K. Knight. 2010. Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 495–503. Association for Computational Linguistics.

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