Use of Deep Linguistic Features for the Recognition and Labeling of Semantic Arguments

John Chen
Department of Computer Science
Columbia University
New York, NY 10027
jchen@cs.columbia.edu

Owen Rambow
Department of Computer Science
Columbia University
New York, NY 10027
rambow@cs.columbia.edu

Abstract

We use deep linguistic features to predict semantic roles on syntactic arguments, and show that these perform considerably better than surface-oriented features. We also show that predicting labels from a "lightweight" parser that generates deep syntactic features performs comparably to using a full parser that generates only surface syntactic features.

1 Introduction

Syntax mediates between surface word order and meaning. The goal of parsing (syntactic analysis) is ultimately to provide the first step towards giving a semantic interpretation of a string of words. So far, attention has focused on parsing, because the semantically annotated corpora required for learning semantic interpretation have not been available. The completion of the first phase of the PropBank (Kingsbury et al., 2002) represents an important step. The PropBank superimposes an annotation of semantic predicate-argument structures on top of the Penn Treebank (PTB) (Marcus et al., 1993; Marcus et al., 1994). The arc labels chosen for the arguments are specific to the predicate, not universal.

In this paper, we find that the use of deep linguistic representations to predict these semantic labels are more effective than the generally more surface-syntax representations previously employed (Gildea and Palmer (2002)). Specifically, we show that the syntactic dependency structure that results from the extraction of a Tree Adjoining Grammar (TAG) from the PTB, and the features that accompany this structure, form a better basis for determining semantic role labels. Crucially, the same structure is also produced when parsing with TAG. We suggest that the syntactic representation chosen in the PTB is less well suited for semantic processing than the other, deeper syntactic representations. In fact, this deeper representation expresses syntactic notions that have achieved a wide acceptance across linguistic frameworks, unlike the very particular surface-syntactic choices made by the linguists who created the PTB syntactic annotation rules.

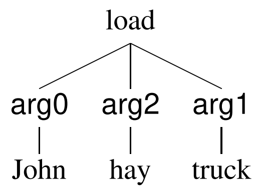

Figure 1: PropBank-style semantic representation for both *John loaded the truck with hay* and *John loaded hay into the truck*

The outline of this paper is as follows. In Section 2 we introduce the PropBank and describe the problem of predicting semantic tags. Section 3 presents an overview of our work and distinguishes it from previous work. Section 4 describes the method used to produce the TAGs that are the basis of our experiments. Section 5 specifies how training and test data that are used in our experiments are derived from the PropBank. Next, we give results on two sets of experiments. Those that predict semantic tags given gold-standard linguistic information are described in Section 6. Those that do prediction from raw text are described in Section 7. Finally, in Section 8 we present concluding remarks.

2 The PropBank and the Labeling of Semantic Roles

The PropBank (Kingsbury et al., 2002) annotates the PTB with dependency structures (or 'predicate-argument' structures), using sense tags for each word and local semantic labels for each argument and adjunct. Argument labels are numbered and used consistently across syntactic alternations for the same verb meaning, as shown in Figure 1. Adjuncts are given special tags such as TMP (for temporal), or LOC (for locatives) derived from the original annotation of the Penn Treebank. In addition to the annotated corpus, PropBank provides a lexicon which lists, for each meaning of each annotated verb, its *roleset*, i.e., the possible arguments in the predicate and their labels. As an example, the entry for the verb *kick*, is given in Figure 2. The notion of "meaning" used is fairly coarse-grained, typically motivated from differing syntactic behavior. Since each verb meaning corresponds to exactly one roleset, these terms are often used interchangeably. The roleset also includes a "descriptor" field which is intended for use during annotation and as documentation, but which does not have any theoretical standing. Each entry also includes examples. Currently there are frames for about 1600 verbs in the corpus, with a total of 2402 rolesets.

Since we did not yet have access to a corpus annotated with rolesets, we concentrate in this paper on predicting the role labels for the arguments. It is only once we have both that we can interpret the relation between predicate and argument at a very fine level (for example, *truck* in *he kicked the truck withhay* as the destination of the loading action). We will turn to the problem of assigning rolesets to predicates once the data is available. We note though that preliminary investigations have shown that for about 65% of predicates (tokens) in the WSJ, there is only one roleset. In a further 7% of predicates (tokens), the set of semantic labels on the arguments of that predicate completely disambiguates the roleset.

<table>
<tr><td>ID</td><td colspan="2">kick.01</td></tr>
<tr><td>Name</td><td colspan="2">drive or impel with the foot</td></tr>
<tr><td>VN/Levin classes</td><td colspan="2">11.4-2, 17.1, 18.1, 23.2
40.3.2, 49</td></tr>
<tr><td rowspan="4">Roles</td><td>Number</td><td>Description</td></tr>
<tr><td>0</td><td>Kicker</td></tr>
<tr><td>1</td><td>Thing kicked</td></tr>
<tr><td>2</td><td>Instrument
(defaults to foot)</td></tr>
<tr><td>Example</td><td colspan="2">[John]$_i$ tried [*trace*$_i$]$_{\text{ARG0}}$ to kick [the football]$_{\text{ARG1}}$</td></tr>
</table>

Figure 2: The unique roleset for *kick*

3 Overview

Gildea and Palmer (2002) show that semantic role labels can be predicted given syntactic features derived from the PTB with fairly high accuracy. Furthermore, they show that this method can be used in conjunction with a parser to produce parses annotated with semantic labels, and that the parser outperforms a chunker. The features they use in their experiments can be listed as follows.

Head Word (**HW**.) The predicate's head word as well as the argument's head word is used.

Phrase Type. This feature represents the type of phrase expressing the semantic role. In Figure 3 **phrase type** for the argument *prices* is NP.

Path. This feature captures the surface syntactic relation between the argument's constituent and the predicate. See Figure 3 for an example.

Position. This binary feature represents whether the argument occurs before or after the predicate in the sentence.

Voice. This binary feature represents whether the predicate is syntactically realized in either passive or active voice.

Notice that for the exception of **voice**, the features solely represent surface syntax aspects of the input parse tree. This should not be taken to mean that deep syntax features are not important. For example, in their inclusion of **voice**, Gildea and Palmer (2002) note that this deep syntax feature plays an important role in connecting semantic role with surface grammatical function.

Aside from **voice**, we posit that other deep linguistic features may be useful to predict semantic role. In this work, we explore the use of more general, deeper syntax features. We also experiment with semantic features derived from the PropBank.

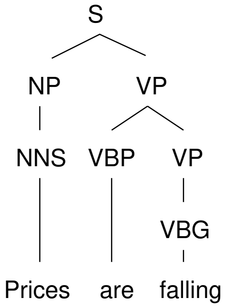

Figure 3: In the predicate argument relationship between the predicate *falling* and the argument *prices*, the **path** feature is VBG↑VP↑VP↑S↓NP.

Our methodology is as follows. The first stage entails generating features representing different levels of linguistic analysis. This is done by first automatically extracting several kinds of TAG from the PropBank. This may in itself generate useful features because TAG structures typically relate closely syntactic arguments with their corresponding predicate. Beyond this, our TAG extraction procedure produces a set of features that relate TAG structures on both the surface-syntax as well as the deep-syntax level. Finally, because a TAG is extracted from the PropBank, we have a set of semantic features derived indirectly from the PropBank through TAG. The second stage of our methodology entails using these features to predict semantic roles. We first experiment with prediction of semantic roles given gold-standard parses from the test corpus. We subsequently experiment with their prediction given raw text fed through a deterministic dependency parser.

4 Extraction of TAGs from the PropBank

Our experiments depend upon automatically extracting TAGs from the PropBank. In doing so, we follow the work of others in extracting grammars of various kinds from the PTB, whether it be TAG (Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2000), combinatory categorial grammar (Hockenmaier and Steedman, 2002), or constraint dependency grammar (Wang and Harper, 2002). We will discuss TAGs and an important principle guiding their formation, the extraction procedure from the PTB that is described in (Chen, 2001) including extensions to extract a TAG from the PropBank, and finally the extraction of deeper linguistic features

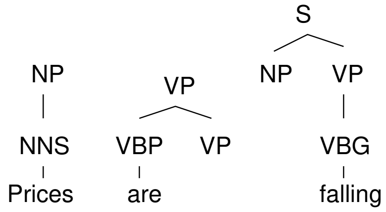

Figure 4: Parse tree associated with the sentence *Prices are falling* has been fragmented into three tree frames.

from the resulting TAG.

A *TAG* is defined to be a set of lexicalized elementary trees (Joshi and Schabes, 1991). They may be composed by several well-defined operations to form parse trees. A lexicalized elementary tree where the lexical item is removed is called a *tree frame* or a *supertag*. The lexical item in the tree is called an *anchor*. Although the TAG formalism allows wide latitude in how elementary trees may be defined, various linguistic principles generally guide their formation. An important principle is that dependencies, including long-distance dependencies, are typically localized the same elementary tree by appropriate grouping of syntactically or semantically related elements.

The extraction procedure fragments a parse tree from the PTB that is provided as input into elementary trees. See Figure 4. These elementary trees can be composed by TAG operations to form the original parse tree. The extraction procedure determines the structure of each elementary tree by localizing dependencies through the use of heuristics. Salient heuristics include the use of a head percolation table (Magerman, 1995), and another table that distinguishes between complements and adjunct nodes in the tree. For our current work, we use the head percolation table to determine heads of phrases. Also, we treat a PropBank argument (ARG0 . . . ARG9) as a complement and a PropBank adjunct (ARGM's) as an adjunct when such annotation is available.[1] Otherwise, we basically follow the approach of (Chen, 2001).[2]

Besides introducing one kind of TAG extraction

[1]The version of the PropBank we are using is not fully annotated with semantic role information, although the most common predicates are.

[2]Specifically, CA1.

procedure, (Chen, 2001) introduces the notion of grouping linguistically-related extracted tree frames together. In one approach, each tree frame is decomposed into a feature vector. Each element of this vector describes a single linguistically-motivated characteristic of the tree.

The elements comprising a feature vector are listed in Table 1. Each elementary tree is decomposed into a feature vector in a relatively straightforward manner. For example, the POS feature is obtained from the preterminal node of the elementary tree. There are also features that specify the syntactic transformations that an elementary tree exhibits. Each such transformation is recognized by structural pattern matching the elementary tree against a pattern that identifies the transformation's existence. For more details, see (Chen, 2001).

Given a set of elementary trees which compose a TAG, and also the feature vector corresponding to each tree, it is possible to annotate each node representing an argument in the tree with role information. These are syntactic roles including for example *subject* and *direct object*. Each argument node is labeled with two kinds of roles: a *surface* syntactic role and a *deep* syntactic role. The former is obtained through determining the position of the node with respect to the anchor of the tree using the usually positional rules for determining argument status in English. The latter is obtained from the former and also from knowledge of the syntactic transformations that have been applied to the tree. For example, we determine the deep syntactic role of a *wh*-moved element by "undoing" the *wh*-movement by using the trace information in the PTB.

The PropBank contains all of the notation of the Penn Treebank as well as semantic notation. For our current work, we extract two kinds of TAG from the PropBank. One grammar, SEM-TAG, has elementary trees annotated with the aforementioned syntactic information as well as semantic information. Semantic information includes semantic role as well as semantic subcategorization information. The other grammar, SYNT-TAG, differs from SEM-TAG only by the absence of any semantic role information.

Table 1: List of each feature in a feature vector and some possible values.

Feature	Values
Part of speech	DT, NN, VB, RB, ...
Subcategorization	NP , NP_S , ∅, ...
MaxProj	S, NP, VP, ...
Modifyee	NP, VP, S, ...
Direction	LEFT, RIGHT
Co-anchors	{ of }, { by }, ∅, ...
Declarative	TRUE, FALSE
Empty Subject	TRUE, FALSE
Complementizer	TRUE, FALSE
Passive	TRUE, FALSE
By-Passive	TRUE, FALSE
Topicalized-X	TRUE, FALSE
Wh-movement-X-Y	TRUE, FALSE
Subject-Aux Inversion	TRUE, FALSE
Relative Clause	TRUE, FALSE

5 Corpora

For our experiments, we use a version of the PropBank where the most commonly appearing predicates have been annotated, not all. Our extracted TAGs are derived from Sections 02-21 of the PTB. Furthermore, training data for our experiments are always derived from these sections. Section 23 is used for test data.

The entire set of semantic roles that are found in the PropBank are not used in our experiments. In particular, we only include as semantic roles those instances in the propbank such that in the extracted TAG they are localized in the same elementary tree. As a consequence, adjunct semantic roles (ARGM's) are basically absent from our test corpus. Furthermore, not all of the complement semantic roles are found in our test corpus. For example, cases of subject-control PRO are ignored because the surface subject is found in a different tree frame than the predicate. Still, a large majority of complement semantic roles are found in our test corpus (more than 87%).

6 Semantic Roles from Gold-Standard Linguistic Information

This section is devoted towards evaluating different features obtained from a gold-standard corpus in the task of determining semantic role. We use the feature set mentioned in Section 3 as well as features derived from TAGs mentioned in Section 4. In this section, we detail the latter set of features. We then describe the results of using different feature sets. These experiments are performed using the C4.5 decision tree machine learning algorithm. The standard settings are used. Furthermore, results are always given using unpruned decision trees because we find that these are the ones that performed the best on a development set.

These features are determined during the extraction of a TAG:

Supertag Path. This is a path in a tree frame from its preterminal to a particular argument node in a tree frame. The **supertag path** of the subject of the rightmost tree frame in Figure 4 is VBG↑VP↑S↓NP.

Supertag. This can be the tree frame corresponding to either the predicate or the argument.

Srole. This is the surface-syntactic role of an argument. Example of values include 0 (subject) and 1 (direct object).

Ssubcat. This is the surface-syntactic subcategorization frame. For example, the **ssubcat** corresponding to a transitive tree frame would be NP0_NP1. PPs as arguments are always annotated with the preposition. For example, the **ssubcat** for the passive version of *hit* would be NP1_NP2(by).

Drole. This is the deep-syntactic role of an argument. Example of values include 0 (subject) and 1 (direct object).

Dsubcat. This is the deep-syntactic subcategorization frame. For example, the **dsubcat** corresponding to a transitive tree frame would be NP0_NP1. Generally, PPs as arguments are annotated with the preposition. For example, the **dsubcat** for *load* is NP0_NP1_NP2(into). The exception is when the argument is not realized as a PP when the predicate is realized in a non-syntactically transformed way. For example, the **dsubcat** for the passive version of *hit* would be NP0_NP1.

Semsubcat. This is the semantic subcategorization frame.

We first experiment with the set of features described in Gildea and Palmer (2002): **Pred HW**, **Arg HW**, **Phrase Type**, **Position**, **Path**, **Voice**. Call this feature set `GP0`. The error rate, 10.0%, is lower than that reported by Gildea and Palmer (2002), 17.2%. This is presumably because our training and test data has been assembled in a different manner as mentioned in Section 5.

Our next experiment is on the same set of features, with the exception that **Path** has been replaced with **Supertag Path**. (Feature set `GP1`). The error rate is reduced from 10.0% to 9.7%. This is statistically significant (*t-test*, $p < 0.05$), albeit a small improvement. One explanation for the improvement is that **Path** does not generalize as well as **Supertag path** does. For example, the **path** feature value VBG↑VP↑VP↑S↓NP reflects surface subject position in the sentence *Prices are falling* but so does VBG↑VP↑S↓NP in the sentence *Sellers regret prices falling.* Because TAG localizes dependencies, the corresponding values for **Supertag path** in these sentences would be identical.

We now experiment with our surface syntax features: **Pred HW**, **Arg HW**, **Ssubcat**, and **Srole**. (Feature set `SURFACE`.) Its performance on SEM-TAG is 8.2% whereas its performance on SYNT-TAG is 7.6%, a tangible improvement over previous models. One reason for the improvement could be that this model is assigning semantic labels with knowledge of the other roles the predicate assigns, unlike previous models.

Our next experiment involves using deep syntax features: **Pred HW**, **Arg HW**, **Dsubcat**, and **Drole**. (Feature set `DEEP`.) Its performance on both SEM-TAG and SYNT-TAG is 6.5%, better than previous models. Its performance is better than `SURFACE` presumably because syntactic transformations are taken to account by deep syntax features. Note also that the transformations which are taken into account are a superset of the transformations taken into account by Gildea and Palmer (2002).

This experiment considers use of semantic features: **Pred HW**, **Arg HW**, **Semsubcat**, and **Drole**. (Feature set `SEMANTIC`.) Of course, there are only results for SEM-TAG, which turns out to be 1.9%. This is the best performance yet.

In our final experiment, we use supertag features: **Pred HW**, **Arg HW**, **Pred Supertag**, **Arg Su-**

Table 2: Error rates of models which label semantic roles on gold-standard parses. Each model is based on its own feature sets, with features coming from a particular kind of extracted grammar.

Feature Set	SEM-TAG	SYNT-TAG
GP0	10.0	10.0
GP1	9.7	9.7
SURFACE	8.2	7.6
DEEP	6.5	6.5
SEMANTIC	1.9	
SUPERTAG	2.8	7.4

pertag, **Drole**. (Feature set SUPERTAG.) The error rates are 2.8% for SEM-TAG and 7.4% for SYNT-TAG. Considering SEM-TAG only, this model performs better than its corresponding DEEP model, probably because supertag for SEM-TAG include crucial semantic information. Considering SYNT-TAG only, this model performs worse than its corresponding DEEP model, presumably because of sparse data problems when modeling supertags. This sparse data problem is also apparent by comparing the model based on SEM-TAG with the corresponding SEM-TAG SEMANTIC model.

7 Semantic Roles from Raw Text

In this section, we are concerned with the problem of finding semantic arguments and labeling them with their correct semantic role given raw text as input. In order to perform this task, we parse this raw text using a combination of supertagging and LDA, which is a method that yields partial dependency parses annotated with TAG structures. We perform this task using both SEM-TAG and SYNT-TAG. For the former, after supertagging and LDA, the task is accomplished because the TAG structures are already annotated with semantic role information. For the latter, we use the best performing model from Section 6 in order to find semantic roles given syntactic features from the parse.

7.1 Supertagging

Supertagging (Bangalore and Joshi (1999)) is the task of assigning a single supertag to each word given raw text as input. For example, given the sentence *Prices are falling*, a supertagger might return the supertagged sentence in Figure 4. Supertagging returns an *almost-parse* in the sense that it is performing much parsing disambiguation. The typical technique to perform supertagging is the trigram model, akin to models of the same name for part-of-speech tagging. This is the technique that we use here.

Data sparseness is a significant issue when supertagging with extracted grammar (Chen and Vijay-Shanker (2000)). For this reason, we smooth the emit probabilities $P(w|t)$ in the trigram model using distributional similarity following Chen (2001). In particular, we use Jaccard's coefficient as the similarity metric with a similarity threshold of 0.04 and a radius of 25 because these were found to attain optimal results in Chen (2001).

Training data for supertagging is Sections 02-21 of the PropBank. A supertagging model based on SEM-TAG performs with 76.32% accuracy on Section 23. The corresponding model for SYNT-TAG performs with 80.34% accuracy. Accuracy is measured for all words in the sentence including punctuation. The SYNT-TAG model performs better than the SEM-TAG model, understandably, because SYNT-TAG is the simpler grammar.

7.2 LDA

LDA is an acronym for Lightweight Dependency Analyzer (Srinivas (1997)). Given as input a supertagged sequence of words, it outputs a partial dependency parse. It takes advantage of the fact that supertagging provides an almost-parse in order to dependency parse the sentence in a simple, deterministic fashion. Basic LDA is a two step procedure. The first step involves linking each word serving as a modifier with the word that it modifies. The second step involves linking each word serving as an argument with its predicate. Linking always only occurs so that grammatical requirements as stipulated by the supertags are satisfied. The version of LDA that is used in this work differs from Srinivas (1997) in that there are other constraints on the linking process.[3] In particular, a link is not established if its existence would create crossing brackets or cycles in the dependency tree for the sentence.

We perform LDA on two versions of Section 23,

[3] We thank Srinivas for the use of his LDA software.

Table 3: Accuracy of dependency parsing using LDA on supertagged input for different kinds of extracted grammar.

Grammar	Recall	Precision	F
SEM-TAG	66.16	74.95	70.58
SYNT-TAG	74.79	80.35	77.47

Table 4: Evaluation of semantic argument recognition on SEM-TAG corpus via supertag and LDA.

Task: determine			Recall	Precision	F
base +		arg	0.39	0.84	0.53
base +	bnd		0.28	0.61	0.38
base +	bnd +	arg	0.28	0.61	0.38

one supertagged with SEM-TAG and the other with SYNT-TAG. The results are shown in Table 3. Evaluation is performed on dependencies excluding leaf-node punctuation. Each dependency is evaluated according to both whether the correct head and dependent is related as well as whether they both receive the correct part of speech tag. The F-measure scores, in the 70% range, are relatively low compared to Collins (1999) which has a corresponding score of around 90%. This is perhaps to be expected because Collins (1999) is based on a full parser. Note also that the accuracy of LDA is highly dependent on the accuracy of the supertagged input. This explains, for example, the fact that the accuracy on SEM-TAG supertagged input is lower than the accuracy with SYNT-TAG supertagged input.

7.3 Semantic Roles from LDA Output

The output of LDA is a partial dependency parse annotated with TAG structures. We can use this output to predict semantic roles of arguments. The manner in which this is done depends on the kind of grammar that is used. The LDA output using SEM-TAG is already annotated with semantic role information because it is encoded in the grammar itself. On the other hand, the LDA output using SYNT-TAG contains strictly syntactic information. In this case, we use the highest performing model from Section 6 in order to label arguments with semantic roles.

Evaluation of prediction of semantic roles takes the following form. Each argument labeled by a semantic role in the test corpus is treated as one trial. Certain aspects of this trial are always checked for correctness. These include checking that the semantic role and the dependency-link are correct. There are other aspects which may or may not be checked, depending on the type of evaluation. One aspect, “bnd,” is whether or not the argument's bracketing as specified in the dependency tree is correct. Another aspect, “arg,” is whether or not the headword of the argument is chosen to be correct.

Table 4 show the results when we use SEM-TAG in order to supertag the input and perform LDA. When the boundaries are found, finding the head word additionally does not result in a decrease of performance. However, correctly identifying the head word instead of the boundaries leads to an important increase in performance. Furthermore, note the low recall and high precision of the “base + arg” evaluation. In part this is due to the nature of the PropBank corpus that we are using. In particular, because not all predicates in our version of the PropBank are annotated with semantic roles, the supertagger for SEM-TAG will sometimes annotate text without semantic roles when in fact it should contain them.

Table 5 shows the results of first supertagging the input with SYNT-TAG and then using a model trained on the `DEEP` feature set to annotate the resulting syntactic structure with semantic roles. This two-step approach greatly increases performance over the corresponding SEM-TAG based approach. These results are comparable to the results from Gildea and Palmer (2002), but only roughly because of differences in corpora. Gildea and Palmer (2002) achieve a recall of 0.50, a precision of 0.58, and an F-measure of 0.54 when using the full parser of Collins (1999). They also experiment with using a chunker which yields a recall of 0.35, a precision of 0.50, and an F-measure of 0.41.

8 Conclusions

We have presented various alternative approaches to predicting PropBank role labels using forms of linguistic information that are deeper than the PTB's surface-syntax labels. These features may either be directly derived from a TAG, such as **Supertag path**, or indirectly via aspects of supertags, such

Table 5: Evaluation of semantic argument recognition on SYNT-TAG corpus via supertag and LDA.

Task: determine			Recall	Precision	F
base +		arg	0.65	0.75	0.70
base +	bnd		0.48	0.55	0.51
base +	bnd +	arg	0.48	0.55	0.51

as deep syntactic features like **Drole**. These are found to produce substantial improvements in accuracy. We believe that such improvement is due to these features better capturing the syntactic information that is relevant for the task of semantic labeling. Also, these features represent syntactic categories about which there is a broad consensus in the literature. Therefore, we believe that our results are portable to other frameworks and differently annotated corpora such as dependency corpora.

We also show that predicting labels from a "lightweight" parser that generates deep syntactic features performs comparably to using a full parser that generates only surface syntactic features. Improvements along this line may be attained by use of a full TAG parser, such as Chiang (2000) for example.

Acknowledgments

This paper is based upon work supported by the National Science Foundation under the KDD program through a supplement to Grant No. IIS-98-17434. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–266.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of tags from the penn treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 65–76.

John Chen. 2001. *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. Ph.D. thesis, University of Delaware.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463, Hong Kong.

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

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *acl02*, pages 239–246, Philadelphia, PA.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas.

Aravind K. Joshi and Yves Schabes. 1991. Tree-adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Definability and Recognizability of Sets of Trees*. Elsevier.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

David Magerman. 1995. Statistical decision-tree models for parsing. In *33rd Meeting of the Association for Computational Linguistics (ACL'95)*.

Mitchell M. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19.2:313–330, June.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*.

B. Srinivas. 1997. Performance evaluation of supertagging for partial parsing. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 187–198, Cambridge, MA.

Wen Wang and Mary P. Harper. 2002. The superarv language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 238–247, Philadelphia, PA.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Fifth Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.