Forest-Based Semantic Role Labeling

Hao Xiong and Haitao Mi and Yang Liu and Qun Liu
Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{xionghao, htmi, yliu, liuqun}@ict.ac.cn

Abstract

Parsing plays an important role in semantic role labeling (SRL) because most SRL systems infer semantic relations from 1-best parses. Therefore, parsing errors inevitably lead to labeling mistakes. To alleviate this problem, we propose to use packed forest, which compactly encodes all parses for a sentence. We design an algorithm to exploit exponentially many parses to learn semantic relations efficiently. Experimental results on the CoNLL-2005 shared task show that using forests achieves an absolute improvement of 1.2% in terms of $\mathbf{F}_1$ **score** over using 1-best parses and 0.6% over using 50-best parses.

Introduction

Semantic role labeling (SRL) is consider to be an important task toward natural language processing, and has been recently used in kinds of natural language applications, such as Information Extraction (Surdeanu et al. 2003), Question and Answering (Shen and Lapata 2007) , Machine Translation (Wu and Fung 2009), Coreference Resolution (Kong et al. 2008) and so on. Given a sentence, the goal of SRL is to assign semantic roles (arguments) to syntactic constituents for each target verb (predicate). Arguments usually include Agent, Patient, Instrument, etc. and also adjuncts such as Locative, Temporal, Manner, Cause, etc. For an overview of semantic role labeling, readers can refer to Màrquez (2009).

Generally, semantic role labeling consists of two steps: identifying and classifying arguments. The former step involves assigning either a semantic argument or non-argument to syntactic element, while the latter includes giving a special semantic role for identified argument. To distinguish the different semantic roles, most previous work map one argument to one syntactic constituent and then extract effective features for the syntactic constituent. Punyakanok, Roth, and tau Yih (2005) shows, most systems rely heavily on the full syntactic parse trees. And because of error propagation and amplification through the chained modules, the overall performance of the system is largely determined by the quality of the automatic syntactic parsers.

However, to our best of knowledge, previous reported work employed only 1-best parses or lists of *k*-best parses, with limited derivations and variations, those syntactic parsing results will inevitably affect the performance of SRL. For example, most of traditional systems firstly map one argument to one syntactic constituent in the parser trees, however, according to our statistics, more than 15% arguments map to many syntactic constituents in the full data of CoNLL-2005[1] shared task(Carreras and Màrquez 2005) because of the parsing errors. Table 1 shows the results of two state of the art Charniak (Charniak 2001) and Collins (Collins 1999) parsers, and the full data still includes more than forty hundred arguments with one-to-many mapping in their corresponding syntactic parser trees.

	Charniak Parser	Collins Parser
AM-MOD	10,099	10,112
A1	9,162	11,327
AM-NEG	3,556	3,560
A0	2,760	3,812
AM-DIS	1,629	1,651

Table 1: The top five arguments, which map to many syntactic constituents obtained with the Charniak and Collins Parsers. (A0 and A1 are two normalized arguments and usually viewed as *subject* and *object* in one sentence. AM-MOD, AM-NEG and AM-DIS are three adjuncts indicating the *modal verbs*,*negation particles* and *clauses*, respectively.)

To alleviate the effects of parsing errors and express more derivations of the parser tree, we employ a packed forest, which almost includes all derivations of parser trees. The employ of packed forest is mainly inspired from the work of (Mi, Huang, and Liu 2008), who also use a packed forest to weaken the impact of parsing errors in a machine translation system. Nevertheless, using packed forest for semantic role labeling, to our knowledge, is the first time. We first extract useful features over forest, and then use a max-entropy classifier to identify and classify the semantic role in one step. [2] Experimental results on the CoNLL-2005 shared task show

[1] http://www.cnts.ua.ac.be/conll2005/

[2] Although support vector machine is more effective for SRL, max-entropy classifier is easier to handle multi-class classification problems and run faster.

that our approach significantly improves the performance of SRL system and achieves an absolute improvement of 1.2% in F_1 score over the 1-best system.

We first briefly describe the previous works of semantic role labeling and review traditional tree-based approach. Then we mainly describe our forest-based model. Finally we present experimental results of different methods and conclude our work.

Semantic Role Labeling

Semantic role labeling plays an important role in natural language processing. Given a sentence, the goal of SRL is to identify argument of each target verb and then classify identified argument into different semantic role. For example, given a sentence "The economy 's temperature will be *taken* from several vantage points this week", the goal of SRL is to identify different arguments for the verb *take* which yields the following output:

$$[_{A1} \text{ The economy 's temperature}][_{AM-MOD} \text{ will}] \text{ be } [_{V} \text{ taken}] \; [_{A2} \text{ from several vantage points}] \; [_{AM-TMP} \text{ this week}].$$

where A1 represents the *thing taken*, A2 represents the *entity taken from*, AM-MOD is an adjunct indicating the modal verb, AM-TMP is also an adjunct indicating the timing of the action and V determines the verb. Generally, arguments such as A1, A2, etc. have different semantics for each target verb that have specified in the PropBank(Kingsbury and Palmer 2002) Frame files. Moreover, each argument can find a constituent in the corresponding full syntactic parse tree. For more definitions of PropBank, readers can refer to (Kingsbury and Palmer 2002; Palmer, Gildea, and Kingsbury 2005).

The work (Gildea and Jurafsky 2002), who used some basic features such as Phrase Type, Governing Category, Parse Tree Path, etc. and employed an interpolation method to identify and classify the syntactic constituents in the FrameNet (Baker, Fillmore, and Lowe 1998), can be viewed as the first work of automatic semantic role labeling. Following this work, some excellent works focused on exploiting additional features(Pradhan et al. 2003; Chen and Rambow 2003; Xue and Palmer 2004; Jiang, Li, and Ng 2005), employing effective machine learning models(Nielsen and Pradhan 2004; Punyakanok et al. 2004; Pradhan et al. 2004; Moschitti 2004; Pradhan et al. 2005a; Zhang et al. 2007), using different syntactic views(Gildea and Hockenmaier 2003; Pradhan et al. 2005b), robust labeling(Pradhan, WayneWard, and H.Martin 2008) and finding similar verbs(Andrew and Reid 2007; Pennacchiotti et al. 2008), etc. In addition, some works focused on semi-supervised(Thompson 2004; Deschacht and Moens 2009; Fürstenau and Lapata 2009a; Fürstenau and Lapata 2009b) or unsupervised semantic role labeling(Swier and Stevenson 2004; Rappoport1 2009). Moreover, semantic role labeling became a well-defined shared task at the CoNLL 2004, 2005 and 2008 conferences.

Most of those previous works can be viewed as tree-based SRL, since they only take as input 1-best or *k*-best parse trees, which inevitably affect the performance of SRL due to the parsing errors. So we first review briefly the tree-based approach and show the drawback of this model in next section. Then we extend to our forest-based model.

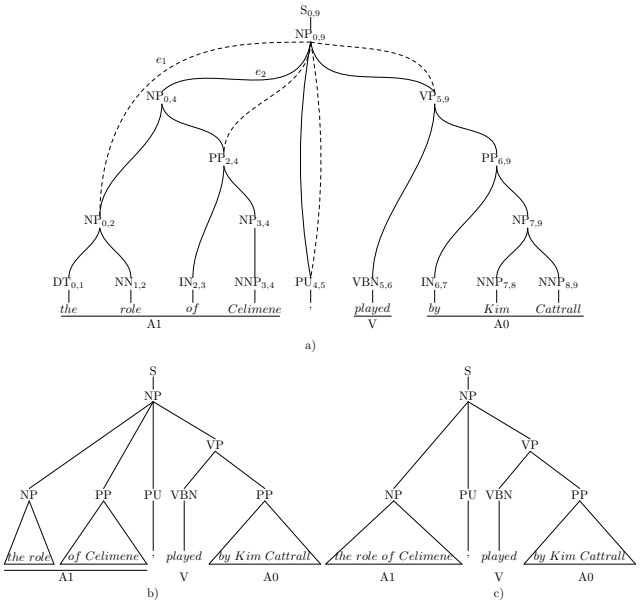

Figure 1: Given the sentence, a) shows its packed forest structure, b) represents the 1-best parse tree and c) is the *k*th-best parse tree.

Tree-based Semantic Role Labeling

Conventional tree-based SRL consists two steps: identifying and classifying arguments.

The first step is to map each argument to one syntactic constituent, they use a bottom-up approach to map the argument to the first syntactic constituent that has the same boundaries. For example, Figure 1.c shows a partial syntactic tree of one sentence. In order to map the argument A1 to one syntactic constituent, they firstly find out the boundary of A1 [*the ... Celimene*], and then climb as high as possible across the syntactic derivations, finally they meet the first satisfied syntactic constituent *NP*. Following this way, another argument A0 can be mapped to the syntactic constituent *PP*. In addition, the constituent with no mapping of arguments is viewed as a special argument. e.g. *VP* in Figure 1.c will be assigned to a role as "non-arg".

After mapping all arguments, useful features over those syntactic constituents are extracted to distinguish different arguments, such as phrase type, first word, last word, etc. Machine learning tools are used to classify the training samples consists of extracted flat features. Consequently, in the decoding stage, they firstly generate the 1-best parse tree for the given sentence, and then map each constituent to one argument that has the highest probability predicted by the machine learning tools. Finally, a greedy strategy is performed to obtain the best assignments.

However, the problem will arise when we do SRL on the tree in Figure 1.b, since no syntactic constituent can be

mapped to argument A1. Actually, the tree in Figure 1.b is more likely generated than the tree in Figure 1.c by the most of the-state-of-art parsers. In order to alleviate this problem, an obvious way is to use *k*-best list. However, a *k*-best list, with its limited scope, has too few variations and too many redundancies. For example, 50-best parses can only express five types of combinational ambiguous derivations ($2^5 < 50 < 2^6$). Therefore, encoding ambiguous derivations as much as possible will results in a large *k*-best parses but with a huge *k*. So it is also inefficient.

Forest-based Semantic Role Labeling

Different from previous works, we use packed forest instead of 1-best or *k*-best parses. A packed forest is a compact structure of parsing results that includes almost all the derivations for a given sentence under context-free grammar (Billott and Lang 1989). In this section, we describe the structure of the packed forest and the features extracted over forest, finally we propose an algorithm for decoding.

Packed Forest

A packed forest (forest), is a compact structure of all the derivations given the sentence. Figure 1.a shows the forest of the sentence "*the role of Celimene , played by Kim Cattrall*".

According to different deductive strategies performed by the parser, node $\text{NP}_{0,9}$ in Figure 1.a includes two different derivations

$$\text{NP}_{0,9} \rightarrow \text{NP}_{0,2}\ \text{PP}_{2,4}\ \text{PU}_{4,5}\ \text{VP}_{5,9}$$
and
$$\text{NP}_{0,9} \rightarrow \text{NP}_{0,4}\ \text{PU}_{4,5}\ \text{VP}_{5,9}$$

which results in two different parses shown in Figure 1.b and Figure 1.c, respectively.

More formally, a compact forest consists of a pair $\langle V, E\rangle$, in which V includes all **nodes** and E involves all **hyperedges** (derivations). Generally, for a given sentence $w_0 w_1 ... w_n$, the node in V has the form of $N_{i,j}$, where the indexes denotes the source span $(w_{i,j-1})$. The hyperedge is a pair $\langle head, tails\rangle$, where **head** and **tails** are consequent and antecedent items of hyperedge, respectively. For example, $e1$ and $e2$ in Figure 1.a are two hyperedges, and

$$\textit{head}(e_1) = \{\text{NP}_{0,9}\},\ \textit{tails}(e_1) = \{\text{NP}_{0,2}, \text{PP}_{2,4}, \text{PU}_{4,5}, \text{VP}_{5,9}\},$$
and
$$\textit{head}(e_2) = \{\text{NP}_{0,9}\},\ \textit{tails}(e_2) = \{\text{NP}_{0,4}, \text{PU}_{4,5}, \text{VP}_{5,9}\}$$

Since the packed forest can efficiently express large number of ambiguous derivations, it can be viewed as a *k*-best parses but with a huge *k*.

Features

Most features in our system are mainly inspired from (Pradhan et al. 2005a) and incorporated with some modification to adapt the forest structure. Here, we only represent the features that should be specially treated, which include

- **Path** records the traversal path through the parser tree from the constituent to the predicate. This feature is unique in 1-best parser tree while in forest structure results in multiple solutions. Taking constituent $NP_{0,2}$ in Figure 1.a for example, this feature results in two values: NP↑NP↓VP↓VBN and NP↑NP↑NP↓VP↓VBN. To avoid causing ambiguity, here we choose the shortest path from the constituent to the predicate. We hence use the former as the value of this feature while drop the latter, where the length is longer than former.[3]
- **Parent POS tag** indicates the POS tag of the parent node. Since one constituent might includes several parents node in the forest, we reserve the parent node appeared in the **Path** feature.
- **Partial path** indicates the partial path, which through the parser tree from the constituent to the lower ancestor of constituent and its predicate. As it also meets ambiguity in forest structure, we use the shortest path the same as the **Path** feature.

Decoding

Given the labeled predicates for a sentence, our searching algorithm first identify the arguments, and then classify them into semantic roles for each predicate.

We first parse the input sentence into a forest. Then, according to the rules of SRL, a constituent in the forest is considered to be a candidate argument when its span does not cross the predicate. For each candidate constituent c_i in the forest f, we attempt to assign an argument with the highest probability from the set of arguments, $A^{1,T}$, which denotes T types of arguments. However, as the different constituents in the forest might have the similar span and one constituent might embed in another, we keep only one argument with the largest probability, since it is easy for a max-entropy classifier to output the probability $p(c_i = a_j)$ of each prediction. We maximize the following objective function:

$$\hat{P} = \arg\max_{a_j \in A^{1,T}} \sum_{c_i \in f} \{\log p(c_i = a_j)\} \quad (1)$$

Since the packed forest consists of huge number of parses, the probability of a constituent assigned to an argument a_j can not be viewed as the value of $p(c_i = a_j)$ predicted by the classifier. Formally, we note that t_k is the score of the k-th parser result generated by the parser and f_k is the k-th parser tree. Therefore, the object function yields the following output.

$$\hat{P} = \arg\max_{a_j \in A^{1,T}} \sum_{c_i \in f_k} \{\delta \cdot \log p(t_k) + \log p(c_i = a_j)\} \quad (2)$$

where δ is a parameter and $(0 \leq \delta \leq 1)$.

In a compact forest, it is not easy to compute the score of k-th parser result. However, we can use the Inside-Outside viterbi algorithm to compute the fractional value $frac(c_i)$ of the constituent instead of the score of k-th parser result.

$$frac(c_i) = \frac{\alpha\beta(c_i)}{\alpha\beta(TOP)} \quad (3)$$

[3] Another alternative is considering the best scored path computed by inside-outside algorithm, however, it gets no improvement in our experiments.

Algorithm 1 Packed forest decoding algorithm.

```
1: Input: Predicate-tagged word sequence
2: Generate packed forest by modified Charniak parser
3: for predicate p_i in the sentence do
4:    for constituent c_j in the packed forest do
5:       if p_i.span ⊈ c_j.span then
6:          ⟨score,type⟩:=GetFeatureScore(p_i,c_j)
7:          st := ⟨score,type⟩
8:          st.score := st.score + δ · c_j.frac
9:          results[c_j.span] := GetMax(results,st)
10:   Sort results in undescended order by score
11:   for ⟨score,type⟩ in results do
12:      span := ⟨score,type⟩.span
13:      type := ⟨score,type⟩.type
14:      if labeled[span] ≠ true then
15:         labeled[span] := true
16:         sentence[span] := type
```

where TOP is the root node of the packed forest. β and α are inside and outside probability, respectively. The fractional value can be viewed as the distance to the 1-best parser. Finally, the object function has the following form.

$$\hat{P} = \arg\max_{a_j \in A^{1,T}} \sum_{c_i \in f} \{\delta \cdot \log \frac{\alpha\beta(c_i)}{\alpha\beta(TOP)} + \log p(c_i = a_j)\} \quad (4)$$

Algorithm 1 shows the whole procedure of the decoding, where in line 6, a function is used to get the basic predicted score generated by the max-entropy classifier given the extracted features, and, in line 9, a function is used to return the higher score. In line 10, we sort the candidates, and then greedily output the best assignments.

Experiments

Experimental Setup

We perform our experiments on CoNLL-2005 shared task, which used sections 02-21 of PropBank for training, section 24 for development and section 23 for test. The whole data consists of 43594 sentences and 262281 arguments, in which there are 35 roles including A0-A5, AA, 14 Adjunct (AM-) and 14 Reference (R-) arguments.

In last section, we used inside-outside algorithm to compute the fractional value for each node. Similarly, we compute the $\alpha\beta(e)$ for each hyperedge:

$$\alpha\beta(e) = \alpha(head(e)) \times \prod_{n_i \in tails(e)} \beta(n_i) \times p(e) \quad (5)$$

where $p(e)$ is the probability of the hyperedge.

Intuitively, this merit can be viewed as the cost of the best derivation that traverses e, and the difference $\eta(e) = \log \alpha\beta(e) - \log \beta(TOP)$ is the distance away from the globally best derivation. Following Huang (2008), we prune away all hyperedges that have $\eta(e) > p$ for a threshold p. Since huge forest involves large number of invalid derivations which potentially lead to misclassification, we can use the threshold p to limit the size of forest and test the impact on performance given different values for p. Compared to 1-best parse tree, pruned forest still includes more constituents, which enlarge the search space. Table 2 lists the number of constituents in the 1-best parse tree and the forest with different pruning threshold p.

	Development	**Test**
1-best (p <0.01)	27,010	46,740
forest (p=3)	41,751	70,603
forest (p=5)	56,342	93,712
forest (p=7)	76,980	122,564

Table 2: Constituents in different syntactic structure

	Precision (%)	**Recall (%)**	**F (%)**
1-best	**79.27**	71.26	75.05
50-best	79.18	72.10	75.47
forest (p=3)	78.13	73.52	**75.75**
forest (p=5)	77.89	73.60	75.68
forest (p=7)	77.42	**73.72**	75.52

Table 3: Performance comparison on develop set

We use the classifier from Le Zhang's Maximum Entropy Modeling Toolkit[4] and use the L-BFGS parameter estimation algorithm with gaussian prior smoothing(Chen and Rosenfeld 1999). We set the gaussian prior to 2 and train the model in 1000 iterations according to the previous experience.

Experimental Results

We firstly perform our experiments on the shared task applied develop set with different methods. The first baseline uses the shared task applied parser tree as the input. And we modified the Charniak parser to generate a 50-best parses, which is another contrast system. In order to get the best pruning threshold, we run three forest-based systems with different pruning threshold but with the similar value of parameter δ (δ=0.5). Table 3 reports the performance of different methods. The performance on forest is higher than two baselines. We can also see that the performance on different pruned forest is different. Since the larger p indicating more derivations are encoded and more ambiguities are represented (for English, a forest pruned with p=5 encodes double size of derivations than that of p=3), we can map more arguments to corresponding constituents, which are absent in 1-best parse tree. However, as the number of constituents increased, the probability of misclassification also increased. Hence, the system with larger p achieves higher recall but with a large decrease in precision causing by its large number of valid derivations.

We also do the experiments given the different value of parameter δ. Figure 2 shows that the best performance is achieved when δ is given to 0.8.

We also investigate in whether the performance of SRL is sensitive to parsing quality? So we vary our system by dividing training data into different size. We train our parser

[4] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

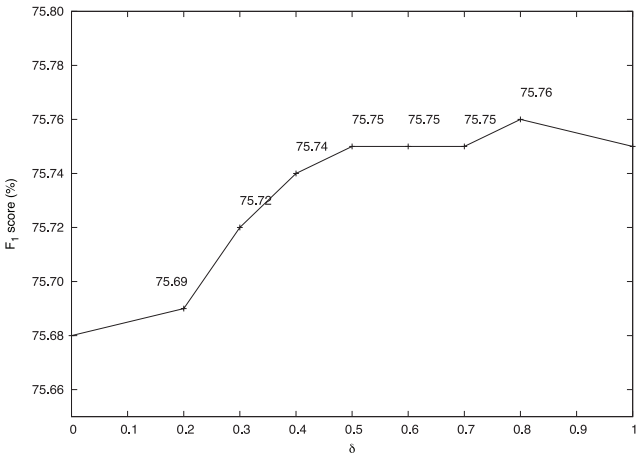

Figure 2: Performance of our system with different value of parameter (δ)

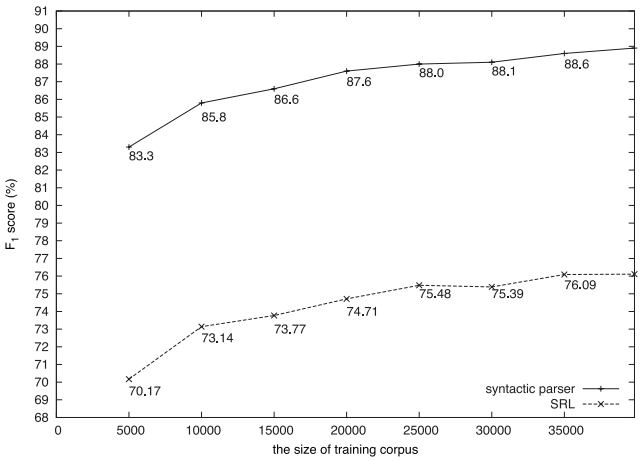

Figure 3: The performance of syntactic parser and SRL when using different size of training corpus to train the parser.

with different size of training corpus, and then parse the test sets using retrained parser. Figure 3 shows, when the size of training corpus increased, the performance of parser and labeler improved significantly, which indicating that the performance of SRL is largely depending on the performance of parser.

Finally, we perform our system on test sets using the best parameters. Table 4 shows the results of different systems on the WSJ corpus. Our methods achieves an absolute improvement of 1.2 percentage in terms of F_1 score. Consequently, from above experiments, we conclude that 1) the performance of SRL is largely depending on the performance of automatic syntactic parsing. 2) whatever using *k*-best parses or forest, the precision of the system increased but the recall decreased. 3) compared to generating large *k*-best parses, using forest is more effective. 4) using a forest can largely alleviate the impaction of parsing errors and further improve the robustness of the SRL. Because of different

	Precision (%)	**Recall (%)**	**F (%)**
1-best	**79.87**	72.69	76.11
50-best	79.49	74.23	76.77
forest($p3$)	78.79	**75.96**	**77.35**

Table 4: Performance comparison on test sets

training method and features, our baseline achieves lower result than the CoNLL-2005 official reported single system, which is 76.46 in F_1 score, however, our forest-based methods still outperform its performance. And we think that its performance should be still largely improved when using our reported methods.

Conclusions and Future Work

In this paper, we propose a novel forest-based method for semantic role labeling. The new method uses a packed forest, which exploits exponentially many parses to learn semantic relations efficiently. The experimental results show that the forest-based system have an absolutely improvement of 1.2% in terms of F_1 score over the 1-best system and 0.6 % over the 50-best parses.

The extension of this work aims to increase the precision of the system. Since the large forest includes more valid derivations, we should exploit more effective features and use heuristic rules to prune the valid derivations. And since some features involves ambiguity in forest,such as **Path**, we just take the shortest one as the value of this feature, which may ignores some discriminative values. Thus, in the future, we should construct a feature forest and design a training algorithm over the feature forest. Moreover, compared to some languages, such as Chinese, English sentence includes less ambiguities relatively. So we attempt to improve the performance of SRL for Chinese or other languages.

Acknowledgements

The authors were supported by National Natural Science Foundation of China, Contracts 90920004 and 60736014, and 863 State Key Project No. 2006AA010108. We would also like to thank Wenbin Jiang for inspirations and generous help, and the anonymous reviewers for suggestions.

References

Andrew, G., and Reid, S. 2007. Generalizing semantic role annotations across syntactically similar verbs. In *Proceedings of ACL-2007*, 192–199.

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The berkeley framenet project. In *Proceedings of COLING-ACL-1998*.

Billott, S., and Lang, B. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of ACL-1989*, 143–151.

Carreras, X., and Màrquez, L. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*.

Charniak, E. 2001. Immediate-head parsing for language models. In *Proceedings of ACL-2001*.

Chen, J., and Rambow, O. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of EMNLP-2003*.

Chen, S. F., and Rosenfeld, R. 1999. A gaussian prior for smoothing maximum entropy models. Technical report, CMU-CS-99-108.

Collins, M. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. Dissertation, Computer Science Department, University of Pennsylvenia.

Deschacht, K., and Moens, M.-F. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of EMNLP-2009*, 21–29.

Fürstenau, H., and Lapata, M. 2009a. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of EMNLP-2009*, 11–20.

Fürstenau, H., and Lapata, M. 2009b. Semi-supervised semantic role labeling. In *Proceedings of EACL-2009*.

Gildea, D., and Hockenmaier, J. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of EMNLP-2003*.

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

Huang, L. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, 586–594.

Jiang, Z. P.; Li, J.; and Ng, H. T. 2005. Semantic argument classification exploiting argument interdependence. In *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, 1067–1072. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kingsbury, P., and Palmer, M. 2002. From treebank to propbank. In *Proceedings of LREC-2002, Spain*.

Kong, F.; Li, Y.; Zhou, G.; Zhu, Q.; and Qian, P. 2008. Using semantic roles for coreference resolution. *Advanced Language Processing and Web Information Technology, International Conference on* 0:150–155.

Màrquez, L. 2009. Semantic role labeling past, present and future. Technical report, TALP Research Center Tecnhical University of Catalonia.

Mi, H.; Huang, L.; and Liu, Q. 2008. Forest-based translation. In *Proceedings of ACL-2008*.

Moschitti, A. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of ACL-2004*.

Nielsen, R. D., and Pradhan, S. 2004. Mixing weak learners in semantic parsing. In *Proceedings of EMNLP-2004*. EMNLP.

Palmer, M.; Gildea, D.; and Kingsbury, P. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1).

Pennacchiotti, M.; Cao, D. D.; Basili, R.; Croce, D.; and Roth, M. 2008. Automatic induction of framenet lexical units. In *Proceedings of EMNLP-2008*.

Pradhan, S.; Hacioglu, K.; Ward, W.; Martin, J. H.; and Jurafsky, D. 2003. Semantic role parsing: adding semantic structure to unstructured text. In *Proceedings of ICDM-2003*.

Pradhan, S.; Ward, W.; Hacioglu, K.; Martin, J. H.; and Jurafsky, D. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of NAACL-HLT 2004*.

Pradhan, S.; Hacioglu, K.; Krugler, V.; Ward, W.; Martin, J. H.; and Jurafsky, D. 2005a. Support vector learning for semantic argument classification. *Journal of Machine Learning*.

Pradhan, S.; Ward, W.; Hacioglu, K.; and Martin, J. H. 2005b. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*.

Pradhan, S.; WayneWard; and H.Martin, J. 2008. Towards robust semantic role labeling. *Computational Linguistics* 34(2):289C310.

Punyakanok, V.; Roth, D.; tau Yih, W.; and Zimak, D. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of COLING-2004*.

Punyakanok, V.; Roth, D.; and tau Yih, W. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI-2005*.

Rappoport1, O. A. R. R. A. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of ACL-2009*.

Shen, D., and Lapata, M. 2007. Using semantic roles to improve question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning*, 12–21.

Surdeanu, M.; Harabagiu, S.; JohnWilliams; and Aarseth, P. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL-2003*.

Swier, R. S., and Stevenson, S. 2004. Unsupervised semantic role labelling. In *Proceedings of EMNLP-2004*.

Thompson, C. A. 2004. Semi-supervised semantic role labeling. In *Proceedings of AAAI-2004*.

Wu, D., and Fung, P. 2009. Semantic roles for smt: A hybrid two-pass model. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, 13–16. Boulder, Colorado: Association for Computational Linguistics.

Xue, N., and Palmer, M. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*.

Zhang, M.; Che, W.; Aw, A. T.; Tan, C. L.; Zhou, G.; Liu, T.; and Li, S. 2007. A grammar-driven convolution tree kernel for semantic role classification. In *Proceedings of ACL-2007*.