Joint Learning Improves Semantic Role Labeling

Kristina Toutanova
Dept of Computer Science
Stanford University
Stanford, CA, 94305
kristina@cs.stanford.edu

Aria Haghighi
Dept of Computer Science
Stanford University
Stanford, CA, 94305
aria42@stanford.edu

Christopher D. Manning
Dept of Computer Science
Stanford University
Stanford, CA, 94305
manning@cs.stanford.edu

Abstract

Despite much recent progress on accurate semantic role labeling, previous work has largely used independent classifiers, possibly combined with separate label sequence models via Viterbi decoding. This stands in stark contrast to the linguistic observation that a core argument frame is a *joint* structure, with strong dependencies between arguments. We show how to build a joint model of argument frames, incorporating novel features that model these interactions into discriminative log-linear models. This system achieves an error reduction of 22% on all arguments and 32% on core arguments over a state-of-the art independent classifier for gold-standard parse trees on PropBank.

1 Introduction

The release of semantically annotated corpora such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2003) has made it possible to develop high-accuracy statistical models for automated semantic role labeling (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Xue and Palmer, 2004). Such systems have identified several linguistically motivated features for discriminating arguments and their labels (see Table 1). These features usually characterize aspects of individual arguments and the predicate.

It is evident that the labels and the features of arguments are highly correlated. For example, there are hard constraints – that arguments cannot overlap with each other or the predicate, and also *soft* constraints – for example, is it unlikely that a predicate will have two or more AGENT arguments, or that a predicate used in the active voice will have a THEME argument prior to an AGENT argument. Several systems have incorporated such dependencies, for example, (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Thompson et al., 2003) and several systems submitted in the CoNLL-2004 shared task (Carreras and Màrquez, 2004). However, we show that there are greater gains to be had by modeling joint information about a verb's argument structure.

We propose a discriminative log-linear joint model for semantic role labeling, which incorporates more global features and achieves superior performance in comparison to state-of-the-art models. To deal with the computational complexity of the task, we employ dynamic programming and reranking approaches. We present performance results on the February 2004 version of PropBank on gold-standard parse trees as well as results on automatic parses generated by Charniak's parser (Charniak, 2000).

2 Semantic Role Labeling: Task Definition and Architectures

Consider the pair of sentences,

- [The GM-Jaguar pact]$_{\text{AGENT}}$ *gives* [the car market]$_{\text{RECIPIENT}}$ [a much-needed boost]$_{\text{THEME}}$
- [A much-needed boost]$_{\text{THEME}}$ was *given* to [the car market]$_{\text{RECIPIENT}}$ by [the GM-Jaguar pact]$_{\text{AGENT}}$

Despite the different syntactic positions of the labeled phrases, we recognize that each plays the same *role* – indicated by the label – in the meaning of this sense of the verb *give*. We call such phrases fillers of semantic roles and our task is, given a sentence and a target verb, to return all such phrases along with their correct labels. Therefore one subtask is to group the words of a sentence into phrases or constituents. As in most previous work on semantic role labeling, we assume the existence of a separate parsing model that can assign a parse tree t to each sentence, and the task then is to label each node in the parse tree with the semantic role of the phrase it dominates, or NONE, if the phrase does not fill any role. We do stress however that the joint framework and features proposed here can also be used when only a shallow parse (chunked) representation is available as in the CoNLL-2004 shared task (Carreras and Màrquez, 2004).

In the February 2004 version of the PropBank corpus, annotations are done on top of the Penn TreeBank II parse trees (Marcus et al., 1993). Possible labels of arguments in this corpus are the *core* argument labels ARG[0-5], and the *modifier* argument labels. The core arguments ARG[3-5] do not have consistent global roles and tend to be verb specific. There are about 14 modifier labels such as ARGM-LOC and ARGM-TMP, for location and temporal modifiers respectively.[1] Figure 1 shows an example parse tree annotated with semantic roles.

We distinguish between models that learn to label nodes in the parse tree independently, called *local models*, and models that incorporate dependencies among the labels of multiple nodes, called *joint models*. We build both local and joint models for semantic role labeling, and evaluate the gains achievable by incorporating joint information. We start by introducing our local models, and later build on them to define joint models.

3 Local Classifiers

In the context of role labeling, we call a classifier local if it assigns a probability (or score) to the label of an individual parse tree node n_i independently of the labels of other nodes.

We use the standard separation of the task of semantic role labeling into *identification* and *classification* phases. In *identification*, our task is to classify nodes of t as either ARG, an argument (including modifiers), or NONE, a non-argument. In *classification*, we are given a set of arguments in t and must label each one with its appropriate semantic role. Formally, let L denote a mapping of the nodes in t to a label set of semantic roles (including NONE) and let $Id(L)$ be the mapping which collapses L's non-NONE values into ARG. Then we can decompose the probability of a labeling L into probabilities according to an identification model P_{ID} and a classification model P_{CLS}.

$$\begin{aligned} P_{SRL}(L|t,v) \quad &= \quad P_{ID}(Id(L)|t,v) \times \\ & \qquad P_{CLS}(L|t,v,Id(L)) \quad (1) \end{aligned}$$

This decomposition does not encode any independence assumptions, but is a useful way of thinking about the problem. Our local models for semantic role labeling use this decomposition. Previous work has also made this distinction because, for example, different features have been found to be more effective for the two tasks, and it has been a good way to make training and search during testing more efficient.

Here we use the same features for local identification and classification models, but use the decomposition for efficiency of training. The identification models are trained to classify each node in a parse tree as ARG or NONE, and the classification models are trained to label each argument node in the training set with its specific label. In this way the training set for the classification models is smaller. Note that we don't do any hard pruning at the identification stage in testing and can find the exact labeling of the complete parse tree, which is the maximizer of Equation 1. Thus we do not have accuracy loss as in the two-pass hard prune strategy described in (Pradhan et al., 2005).

In previous work, various machine learning methods have been used to learn local classifiers for role labeling. Examples are linearly interpolated relative frequency models (Gildea and Jurafsky, 2002), SVMs (Pradhan et al., 2004), decision trees (Surdeanu et al., 2003), and log-linear models (Xue and Palmer, 2004). In this work we use log-linear models for multi-class classification. One advantage of log-linear models over SVMs for us is that they produce probability distributions and thus identification

[1] For a full listing of PropBank argument labels see (Palmer et al., 2003)

Standard Features (Gildea and Jurafsky, 2002)
PHRASE TYPE: Syntactic Category of node
PREDICATE LEMMA: Stemmed Verb
PATH: Path from node to predicate
POSITION: Before or after predicate?
VOICE: Active or passive relative to predicate
HEAD WORD OF PHRASE
SUB-CAT: CFG expansion of predicate's parent
Additional Features (Pradhan et al., 2004)
FIRST/LAST WORD
LEFT/RIGHT SISTER PHRASE-TYPE
LEFT/RIGHT SISTER HEAD WORD/POS
PARENT PHRASE-TYPE
PARENT POS/HEAD-WORD
ORDINAL TREE DISTANCE: Phrase Type with appended length of PATH feature
NODE-LCA PARTIAL PATH Path from constituent to Lowest Common Ancestor with predicate node
PP PARENT HEAD WORD If parent is a PP return parent's head word
PP NP HEAD WORD/POS For a PP, retrieve the head Word / POS of its rightmost NP
Selected Pairs (Xue and Palmer, 2004)
PREDICATE LEMMA & PATH
PREDICATE LEMMA & HEAD WORD
PREDICATE LEMMA & PHRASE TYPE
VOICE & POSITION
PREDICATE LEMMA & PP PARENT HEAD WORD

Table 1: Baseline Features

and classification models can be chained in a principled way, as in Equation 1.

The features we used for local identification and classification models are outlined in Table 1. These features are a subset of features used in previous work. The standard features at the top of the table were defined by (Gildea and Jurafsky, 2002), and the rest are other useful lexical and structural features identified in more recent work (Pradhan et al., 2004; Surdeanu et al., 2003; Xue and Palmer, 2004).

The most direct way to use trained local identification and classification models in testing is to select a labeling L of the parse tree that maximizes the product of the probabilities according to the two models as in Equation 1. Since these models are local, this is equivalent to independently maximizing the product of the probabilities of the two models for the label l_i of each parse tree node n_i as shown below in Equation 2.

$$\begin{aligned} P^{\ell}_{SRL}(L|t,v) &= \prod_{n_i \in t} P_{ID}(Id(l_i)|t,v) \\ &\times \prod_{n_i \in t} P_{CLS}(l_i|t,v,Id(l_i)) \end{aligned} \quad (2)$$

A problem with this approach is that a maximizing labeling of the nodes could possibly violate the constraint that argument nodes should not overlap with each other. Therefore, to produce a consistent set of arguments with local classifiers, we must have a way of enforcing the non-overlapping constraint.

3.1 Enforcing the Non-overlapping Constraint

Here we describe a fast exact dynamic programming algorithm to find the most likely non-overlapping (consistent) labeling of all nodes in the parse tree, according to a product of probabilities from local models, as in Equation 2. For simplicity, we describe the dynamic program for the case where only two classes are possible – ARG and NONE. The generalization to more classes is straightforward. Intuitively, the algorithm is similar to the Viterbi algorithm for context-free grammars, because we can describe the non-overlapping constraint by a "grammar" that disallows ARG nodes to have ARG descendants.

Below we will talk about maximizing the sum of the logs of local probabilities rather than the product of local probabilities, which is equivalent. The dynamic program works from the leaves of the tree up and finds a best assignment for each tree, using already computed assignments for its children. Suppose we want the most likely consistent assignment for subtree t with children trees $t_1, \ldots, t_k$ each storing the most likely consistent assignment of nodes it dominates as well as the log-probability of the assignment of all nodes it dominates to NONE. The most likely assignment for t is the one that corresponds to the maximum of:

- The sum of the log-probabilities of the most likely assignments of the children subtrees $t_1, \ldots, t_k$ plus the log-probability for assigning the node t to NONE
- The sum of the log-probabilities for assigning all of t_i's nodes to NONE plus the log-probability for assigning the node t to ARG.

Propagating this procedure from the leaves to the root of t, we have our most likely non-overlapping assignment. By slightly modifying this procedure, we obtain the most likely assignment according to a product of local identification and classification models. We use the local models in conjunction with this search procedure to select a most likely labeling in testing. Test set results for our local model P^{ℓ}_{SRL} are given in Table 2.

4 Joint Classifiers

As discussed in previous work, there are strong dependencies among the labels of the semantic argument nodes of a verb. A drawback of local models is that, when they decide the label of a parse tree node, they cannot use information about the labels and features of other nodes in the tree.

Furthermore, these dependencies are highly nonlocal. For instance, to avoid repeating argument labels in a frame, we need to add a dependency from each node label to the labels of all other nodes. A factorized sequence model that assumes a finite Markov horizon, such as a chain Conditional Random Field (Lafferty et al., 2001), would not be able to encode such dependencies.

The need for Re-ranking

For argument identification, the number of possible assignments for a parse tree with n nodes is 2^n. This number can run into the hundreds of billions for a normal-sized tree. For argument labeling, the number of possible assignments is $\approx 20^m$, if m is the number of arguments of a verb (typically between 2 and 5), and 20 is the approximate number of possible labels if considering both core and modifying arguments. Training a model which has such huge number of classes is infeasible if the model does not factorize due to strong independence assumptions. Therefore, in order to be able to incorporate long-range dependencies in our models, we chose to adopt a re-ranking approach (Collins, 2000), which selects from likely assignments generated by a model which makes stronger independence assumptions. We utilize the top N assignments of our local semantic role labeling model P^{ℓ}_{SRL} to generate likely assignments. As can be seen from Table 3, for relatively small values of N, our re-ranking approach does not present a serious bottleneck to performance. We used a value of $N = 20$ for training. In Table 3 we can see that if we could pick, using an oracle, the best assignment out for the top 20 assignments according to the local model, we would achieve an F-Measure of 98.8 on all arguments. Increasing the number of N to 30 results in a very small gain in the upper bound on performance and a large increase in memory requirements. We therefore selected $N = 20$ as a good compromise.

Generation of top N most likely joint assignments

We generate the top N most likely non-overlapping joint assignments of labels to nodes in a parse tree according to a local model P^{ℓ}_{SRL}, by an exact dynamic programming algorithm, which is a generalization of the algorithm for finding the top non-overlapping assignment described in section 3.1.

Parametric Models

We learn log-linear re-ranking models for joint semantic role labeling, which use feature maps from a parse tree and label sequence to a vector space. The form of the models is as follows. Let $\Phi(t, v, L) \in \mathbb{R}^s$ denote a feature map from a tree t, target verb v, and joint assignment L of the nodes of the tree, to the vector space $\mathbb{R}^s$. Let $L_1, L_2, \cdots, L_N$ denote top N possible joint assignments. We learn a log-linear model with a parameter vector W, with one weight for each of the s dimensions of the feature vector. The probability (or score) of an assignment L according to this re-ranking model is defined as:

$$P^r_{SRL}(L|t, v) = \frac{e^{\langle \Phi(t,v,L), W \rangle}}{\sum_{j=1}^{N} e^{\langle \Phi(t,v,L_j).W \rangle}} \quad (3)$$

The score of an assignment L not in the top N is zero. We train the model to maximize the sum of log-likelihoods of the best assignments minus a quadratic regularization term.

In this framework, we can define arbitrary features of labeled trees that capture general properties of predicate-argument structure.

Joint Model Features

We will introduce the features of the joint re-ranking model in the context of the example parse tree shown in Figure 1. We model dependencies not only between the label of a node and the labels of

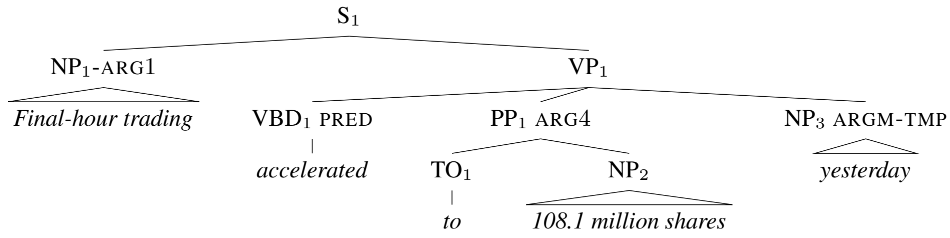

Figure 1: An example tree from the PropBank with Semantic Role Annotations.

other nodes, but also dependencies between the label of a node and input features of other argument nodes. The features are specified by instantiation of templates and the value of a feature is the number of times a particular pattern occurs in the labeled tree.

Templates

For a tree t, predicate v, and joint assignment L of labels to the nodes of the tree, we define the *candidate argument sequence* as the sequence of non-NONE labeled nodes $[n_1, l_1, \ldots, v_{PRED}, n_m, l_m]$ (l_i is the label of node n_i). A reasonable candidate argument sequence usually contains very few of the nodes in the tree – about 2 to 7 nodes, as this is the typical number of arguments for a verb. To make it more convenient to express our feature templates, we include the predicate node v in the sequence. This sequence of labeled nodes is defined with respect to the left-to-right order of constituents in the parse tree. Since non-NONE labeled nodes do not overlap, there is a strict left-to-right order among these nodes. The candidate argument sequence that corresponds to the correct assignment in Figure 1 will be:

[NP_1-ARG1,VBD_1-PRED,PP_1-ARG4,NP_3-ARGM-TMP]

Features from Local Models: All features included in the local models are also included in our joint models. In particular, each template for local features is included as a joint template that concatenates the local template and the node label. For example, for the local feature PATH, we define a joint feature template, that extracts PATH from every node in the candidate argument sequence and concatenates it with the label of the node. Both a feature with the specific argument label is created and a feature with the generic back-off ARG label. This is similar to adding features from identification and classification models. In the case of the example candidate argument sequence above, for the node NP_1 we have the features:

(NP↑S↓)-ARG1, (NP↑S↓)-ARG

When comparing a local and a joint model, we use the same set of local feature templates in the two models.

Whole Label Sequence: As observed in previous work (Gildea and Jurafsky, 2002; Pradhan et al., 2004), including information about the set or sequence of labels assigned to argument nodes should be very helpful for disambiguation. For example, including such information will make the model less likely to pick multiple fillers for the same role or to come up with a labeling that does not contain an obligatory argument. We added a whole label sequence feature template that extracts the labels of all argument nodes, and preserves information about the position of the predicate. The template also includes information about the voice of the predicate. For example, this template will be instantiated as follows for the example candidate argument sequence:

[voice:active ARG1,PRED,ARG4,ARGM-TMP]

We also add a variant of this feature which uses a generic ARG label instead of specific labels. This feature template has the effect of counting the number of arguments to the left and right of the predicate, which provides useful global information about argument structure. As previously observed (Pradhan et al., 2004), including modifying arguments in sequence features is not helpful. This was confirmed in our experiments and we redefined the whole label sequence features to exclude modifying arguments.

One important variation of this feature uses the actual predicate lemma in addition to “voice:active”. Additionally, we define variations of these feature templates that concatenate the label sequence with features of individual nodes. We experimented with variations, and found that including the phrase type and the head of a directly dominating PP – if one exists – was most helpful. We also add a feature that detects repetitions of the same label in a candidate argument sequence, together with the phrase types of the nodes labeled with that label. For example, (NP-ARG0,WHNP-ARG0) is a common pattern of this form.

Frame Features: Another very effective class of features we defined are features that look at the label of a single argument node and internal features of other argument nodes. The idea of these features is to capture knowledge about the label of a constituent given the syntactic realization of all arguments of the verb. This is helpful to capture syntactic alternations, such as the dative alternation. For example, consider the sentence (*i*) "[Shaw Publishing]$_{\text{ARG}_0}$ *offered* [Mr. Smith]$_{\text{ARG2}}$ [a reimbursement]$_{\text{ARG}_1}$" and the alternative realization (*ii*) "[Shaw Publishing]$_{\text{ARG}_0}$ *offered* [a reimbursement]$_{\text{ARG}_1}$ [to Mr. Smith]$_{\text{ARG}_2}$". When classifying the NP in object position, it is useful to know whether the following argument is a PP. If yes, the NP will more likely be an ARG$_1$, and if not, it will more likely be an ARG$_2$. A feature template that captures such information extracts, for each argument node, its phrase type and label in the context of the phrase types for all other arguments. For example, the instantiation of such a template for [a reimbursement] in (*ii*) would be

[voice:active NP,PRED,NP-ARG$_1$,PP]

We also add a template that concatenates the identity of the predicate lemma itself.

We should note that Xue and Palmer (2004) define a similar feature template, called *syntactic frame*, which often captures similar information. The important difference is that their template extracts contextual information from noun phrases surrounding the predicate, rather than from the sequence of argument nodes. Because our model is joint, we are able to use information about other argument nodes when labeling a node.

Final Pipeline

Here we describe the application in testing of a joint model for semantic role labeling, using a local model P^{ℓ}_{SRL}, and a joint re-ranking model P^{r}_{SRL}. P^{ℓ}_{SRL} is used to generate top N non-overlapping joint assignments $L_1, \ldots, L_N$.

One option is to select the best L_i according to P^{r}_{SRL}, as in Equation 3, ignoring the score from the local model. In our experiments, we noticed that for larger values of N, the performance of our re-ranking model P^{r}_{SRL} decreased. This was probably due to the fact that at test time the local classifier produces very poor argument frames near the bottom of the top N for large N. Since the re-ranking model is trained on relatively few good argument frames, it cannot easily rule out very bad frames. It makes sense then to incorporate the local model into our final score. Our final score is given by:

$$P_{SRL}(L|t,v) = (P^{\ell}_{SRL}(L|t,v))^{\alpha}\; P^{r}_{SRL}(L|t,v)$$

where α is a tunable parameter [2] for how much influence the local score has in the final score. Such interpolation with a score from a first-pass model was also used for parse re-ranking in (Collins, 2000). Given this score, at test time we choose among the top N local assignments $L_1, \ldots, L_N$ according to:

$$\underset{L \in \{L_1,\ldots,L_N\}}{\arg\max} \quad \alpha \log P^{\ell}_{SRL}(L|t,v) + \log P^{r}_{SRL}(L|t,v)$$

5 Experiments and Results

For our experiments we used the February 2004 release of PropBank. [3] As is standard, we used the annotations from sections 02–21 for training, 24 for development, and 23 for testing. As is done in some previous work on semantic role labeling, we discard the relatively infrequent discontinuous arguments from both the training and test sets. In addition to reporting the standard results on individual argument F-Measure, we also report Frame Accuracy (Acc.), the fraction of sentences for which we successfully label all nodes. There are reasons to prefer Frame Accuracy as a measure of performance over individual-argument statistics. Foremost, potential applications of role labeling may require correct labeling of all (or at least the core) arguments in a sentence in order to be effective, and partially correct labelings may not be very useful.

[2] We found $\alpha = 0.5$ to work best

[3] Although the first official release of PropBank was recently released, we have not had time to test on it.

Task	CORE		ARGM	
	F1	Acc.	F1	Acc.
Identification	95.1	84.0	95.2	80.5
Classification	96.0	93.3	93.6	85.6
Id+Classification	92.2	80.7	89.9	71.8

Table 2: Performance of local classifiers on identification, classification, and identification+classification on section 23, using gold-standard parse trees.

N	CORE		ARGM	
	F1	Acc.	F1	Acc.
1	92.2	80.7	89.9	71.8
5	97.8	93.9	96.8	89.5
20	99.2	97.4	98.8	95.3
30	99.3	97.9	99.0	96.2

Table 3: Oracle upper bounds for performance on the complete identification+classification task, using varying numbers of top N joint labelings according to local classifiers.

Model	CORE		ARGM	
	F1	Acc.	F1	Acc.
Local	92.2	80.7	89.9	71.8
Joint	**94.7**	**88.2**	**92.1**	**79.4**

Table 4: Performance of local and joint models on identification+classification on section 23, using gold-standard parse trees.

We report results for two variations of the semantic role labeling task. For CORE, we identify and label only core arguments. For ARGM, we identify and label core as well as modifier arguments. We report results for local and joint models on argument identification, argument classification, and the complete identification and classification pipeline. Our local models use the features listed in Table 1 and the technique for enforcing the non-overlapping constraint discussed in Section 3.1.

The labeling of the tree in Figure 1 is a specific example of the kind of errors fixed by the joint models. The local classifier labeled the first argument in the tree as ARG0 instead of ARG1, probably because an ARG0 label is more likely for the subject position.

All joint models for these experiments used the whole sequence and frame features. As can be seen from Table 4, our joint models achieve error reductions of 32% and 22% over our local models in F-Measure on CORE and ARGM respectively. With respect to the Frame Accuracy metric, the joint error reduction is 38% and 26% for CORE and ARGM respectively.

We also report results on automatic parses (see Table 5). We trained and tested on automatic parse trees from Charniak's parser (Charniak, 2000). For approximately 5.6% of the argument constituents in the test set, we could not find exact matches in the automatic parses. Instead of discarding these arguments, we took the largest constituent in the automatic parse having the same head-word as the gold-standard argument constituent. Also, 19 of the propositions in the test set were discarded because Charniak's parser altered the tokenization of the input sentence and tokens could not be aligned. As our results show, the error reduction of our joint model with respect to the local model is more modest in this setting. One reason for this is the lower upper bound, due largely to the the much poorer performance of the identification model on automatic parses. For ARGM, the local identification model achieves 85.9 F-Measure and 59.4 Frame Accuracy; the local classification model achieves 92.3 F-Measure and 83.1 Frame Accuracy. It seems that the largest boost would come from features that can identify arguments in the presence of parser errors, rather than the features of our joint model, which ensure global coherence of the argument frame. We still achieve 10.7% and 18.5% error reduction for CORE arguments in F-Measure and Frame Accuracy respectively.

Model	CORE		ARGM	
	F1	Acc.	F1	Acc.
Local	84.1	66.5	81.4	55.6
Joint	85.8	72.7	82.9	60.8

Table 5: Performance of local and joint models on identification+classification on section 23, using Charniak automatically generated parse trees.

6 Related Work

Several semantic role labeling systems have successfully utilized joint information. (Gildea and Jurafsky, 2002) used the empirical probability of the set of proposed arguments as a prior distribution. (Pradhan et al., 2004) train a language model over label sequences. (Punyakanok et al., 2004) use a linear programming framework to ensure that the only argument frames which get probability mass are ones that respect global constraints on argument labels.

The key differences of our approach compared to previous work are that our model has all of the following properties: (*i*) we do not assume a finite Markov horizon for dependencies among node labels, (*ii*) we include features looking at the labels of multiple argument nodes and *internal features* of these nodes, and (*iii*) we train a discriminative model capable of incorporating these long-distance dependencies.

7 Conclusions

Reflecting linguistic intuition and in line with current work, we have shown that there are substantial gains to be had by jointly modeling the argument frames of verbs. This is especially true when we model the dependencies with discriminative models capable of incorporating long-distance features.

8 Acknowledgements

The authors would like to thank the reviewers for their helpful comments and Dan Jurafsky for his insightful suggestions and useful discussions. This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program.

References

Collin Baker, Charles Fillmore, and John Lowe. 1998. The Berkeley Framenet project. In *Proceedings of COLING-ACL-1998*.

Xavier Carreras and Luís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML-2000*.

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

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-2001*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2003. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL-2004*.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning Journal*.

Vasin Punyakanok, Dan Roth, Wen tau Yih, Dav Zimak, and Yuancheng Tu. 2004. Semantic role labeling via generalized inference over classifiers. In *Proceedings of CoNLL-2004*.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL-2003*.

Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling. In *Proceedings of ECML-2003*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*.