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 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 local 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 17% on all arguments and 37% on core arguments over the best published results for gold-standard parse trees on PropBank.

1 Introduction

The release of semantic 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.

In addition to features of individual argument nodes, previous work has explored the dependence among classification decisions at different nodes within a parse tree. 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; Surdeanu et al., 2003; Thompson et al., 2003). 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 Collins' parser.

2 Semantic Role Labeling: Task Definition and Architectures

Consider the pair of sentences,

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

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 particular 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 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. (Gildea and Palmer, 2002) have shown that the use of parse trees greatly improves performance. In the February 2004 version of the PropBank corpus, annotations are done on top of the Penn TreeBank II parse trees. Possible labels of arguments in this corpus are the *core* argument labels ARG[0-5] and the *modifier* argument labels. 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.

One can separate semantic role labeling into *identification* and *classification* phases. In *identification*, our task is to classify nodes of t as either ARG, an argument, 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. We will view both phases as classification tasks on structured input data, namely the tree t, and in the sequel we will outline local and joint approaches for both of these tasks. Let $Id(l)$ denote the function from all labels such that $Id(l)$=NONE if and only if l=NONE, and $Id(l)$=ARG otherwise. If L is a vector of labels for all nodes in t, $Id(L)$ denotes the vector mapped into identification labels denoting argument and non-argument assignments. We can decompose the probability:

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

into probabilities according to an identification model P_{ID} and a classification model P_{CLS}. This decomposition does not encode any independence assumptions, but is a useful way of thinking about the problem. Our system for semantic role labeling is based on 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.

We discuss our probabilistic models for identification and classification next. We start with local models – ones that solve the task as independent labeling of nodes, and later introduce global models.

3 Local Classifiers

In the context of role labeling, we call a classifier local if it assigns a label to an individual parse tree node without knowledge of the labels of other nodes. In the case of a probabilistic classifier, the joint probability of an assignment of labels to all parse tree nodes is the product of probabilities of labels assigned to individual nodes: $P(L|t,v) = \prod_{n_i \in t} P(l_i|t,v)$.

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 can thus be used in a pipeline architecture with identification and classification phases in a more principled way.

The basic features we use are outlined in the top part of Table 1 and pairs of these features can be used to improve performance (Xue and Palmer, 2004). Recent work (Pradhan et al., 2004; Surdeanu et al., 2003), has also shown gains from additional features (see the middle part of Table 1) which include many more lexicalized features as well as constituent-relative features.

3.1 Resolving Constraints In Identification

In the PropBank corpus, argument constituents of the same verb cannot overlap. However, the local classifier does not take this constraint into account

[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)
HEAD POS
FIRST/LAST PHRASE-TYPE/WORD/POS
LEFT/RIGHT SISTER PHRASE-TYPE/POS:
PARENT PHRASE-TYPE/POS
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
Selected Pairs (Xue and Palmer, 2004)
PREDICATE LEMMA & PATH
PREDICATE LEMMA & HEAD WORD
PREDICATE LEMMA & PHRASE TYPE
VOICE & POSITION

Table 1: Baseline Feature

during classification and therefore may label two overlapping nodes as ARGs. Therefore, to produce a consistent set of arguments with local classifiers, we must have a way of resolving overlapping arguments. Previous work (Pradhan et al., 2004), has locally resolved each overlap conflict greedily by selecting the higher-confidence node. We propose the following procedure which globally optimizes our local classifiers' confidence while respecting the non-overlapping node constraint.

We can assign a log-probability to the total assignment of nodes of t to $\{$ARG,NONE$\}$ made by the local classifier simply by adding log-probabilities of each individual assignment; if we suppose that the probabilities are indeed independent this sum corresponds to the log joint probability of the assignment. We compute the most likely assignment of nodes which respects the no-overlap constraint using the following simple procedure:

Suppose we want the most likely valid assignment for subtree t with children trees $t_1, \ldots, t_k$ each storing the most likely valid 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.

3.2 New Proposed Features

We implemented several additional features to improve the local classifiers' performance. We used a handful of tgrep expressions to detect dependency relationships - such as subject and object - and define a DEPENDENCY feature that takes the value of the relationship between the head of the constituent and the predicate. Also, we noticed that often the head word of a phrase is not the most semantically informative one. By modifying Collins' head propagation rules, as given in (Collins, 1999), we derived the SEMANTIC HEAD feature[2] ; We also noticed that the PATH feature gave different paths to the subject when the lexical verb was deeply embedded in an infinitival clause or when used with auxiliary verbs. To overcome this, we added a PROJECTED-PATH feature which is the path feature taken from the *projected* predicate constituent - the highest VP that has the predicate as its SEMANTIC-HEAD. Similarly, we utilize a PROJECTED-SUBCAT feature, which uses the CFG production of the *projected* predicate's parent. We also used WordNet based features. The results showed a relatively modest increase in performance between $.2 - .3\%$. We concluded that there was probably not huge room left for improvement in devising new basic features for individual constituents, and chose to instead concentrate on features that characterized the whole sequence of arguments to be classified.

4 Joint Classifiers

Joint classifiers assign labels to all nodes in a parse tree simultaneously and a classification is a joint assignment to all nodes.

[2] A similar feature is used in (Pradhan et al., 2004; Surdeanu et al., 2003)

Discriminative 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. Fortunately, most assignments can be ruled out by simple models, and the number of plausible assignments is much smaller.

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 from simpler models making stronger independence assumptions. We use local identification and classification models to generate top N most likely joint assignments of labels.

Generation of top N most likely joint assignments

The problem of generating the top N most likely joint assignments of labels to nodes in a parse tree given a local model $P(l|n, v, t)$ is more challenging for argument identification, since we want to select top N assignments that obey the non-overlapping constraint. We do this by an exact dynamic programming algorithm which is a slight modification of the algorithm for finding the best consistent assignment described in section 3.1. Finding top N joint assignments for argument classification, given a set of argument nodes, is the same as in zero order Markov models.

Parametric Models

We learn log-linear re-ranking models for joint *identification* and *classification*, 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 N selected 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 model is defined as:

$$P(L|t, v) = \frac{\exp \langle \Phi(t, v, L), W \rangle}{\sum_{j=1,\ldots,N} \exp \langle \Phi(t, v, L_j), W \rangle}$$

The score of an assignment L not in the top N is zero. We used values of N between 10 and 20 for training. 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 candidate node and label sequences that capture general properties of predicate-argument structure.

Joint Model Features

We use similar sequence features for joint identification and classification models. We will introduce the features in the context of the example parse tree shown in Figure 1. We include features that capture the correlation between the decisions at different nodes in the parse tree. We model dependencies not only between the label of a node and the labels of 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). The candidate argument sequence usually contains very few of the nodes in the tree – about 2 to 7 nodes. 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. For the example parse tree in Figure 1, if the

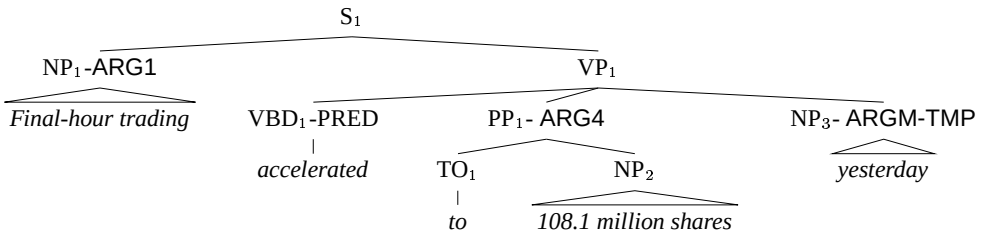

Figure 1: An example tree from the PropBank with Semantic Role Annotations.

assignment of labels to the nodes is the correct assignment shown, the candidate argument sequence will be [NP$_1$-ARG$_1$,VBD$_1$-PRED,PP$_1$-ARG$_4$,NP$_3$-ARGM-TMP] in a model for joint classification, and [NP$_1$-ARG,VBD$_1$-PRED,PP$_1$-ARG,NP$_3$-ARG] in a model for joint identification.

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 and concatenates it with the label of the node. 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 [voice:active ARG$_1$,PRED,ARG$_4$,ARGM-TMP] in a model for classification, and [voice:active ARG,PRED,ARG,ARG] in a model for identification. Note that in a model for identification, 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.

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 experimented with repetitions of the same label and bigrams. We report the performance improvement using such sequence features in Section 5, Table 2.

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]$_{ARG0}$ *offered* [Mr. Smith]$_{ARG2}$ [a reimbursement]$_{ARG1}$ " and the alternative realization (*ii*)"[Shaw Publishing]$_{ARG0}$ *offered* [a reimbursement]$_{ARG1}$ [to Mr. Smith]$_{ARG2}$". 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 ARG1, and if not, it will more likely be an ARG2. 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 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. Another variation extracts internal

features and label for a focussed argument node, and only place-holders for other argument nodes, thus allowing conditioning of the label on the position of the node in the argument sequence.

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

To obtain a probabilistic model of complete assignments of labels to nodes, $P_{SRL}(L|t,v)$, we chain the models for identification $P_{ID}(Id(L)|t,v)$ and classification $P_{CLS}(L|t,v,Id(L))$ as described in Section 2. In testing, in order to find the joint assignment L^* which maximizes the probability according to the full SRL model, $P_{ID}(Id(L)|t,v) \times P_{CLS}(L|t,v,Id(L))$, we need to perform computationally expensive search. As done in previous work, we perform a heuristic search by considering only the top K joint identification labelings. We find an approximate most likely assignment as follows:

$$\arg\max_L P_{SRL}(L|t,v) \approx \arg\max_{Id(L) \in topK(P_{ID}(Id(L)|t,v))} [P_{ID}(Id(L)|t,v) \times \arg\max_L P_{CLS}(L|t,v,Id(L))]$$

We experimented with different numbers of top K hypotheses considered, and our results showed highest performance for values of K around 4 to 10. Exploring more hypotheses was slightly harmful, which is probably due to the fact that our joint identification model is trained to distinguish only among a few plausible ones.

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 as training data, 24 for development, and 23 for testing. In addition to reporting the standard results on individual argument precision, recall, and F-measure, we also report Frame Accuracy (Fr. 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 labeling of all arguments in a sentence in order to be effective and partially correct labelings may not be very useful.

We report results for three variations of the semantic role labeling task. For CORE, we identify and label only core arguments. For ARGM, we identify all arguments but label modifiers as only ARGM, with no indication of the specific modifier type. In ARGMPLUS, we identify and classify all arguments with their specific labels. We report results for local and joint models on argument identification, argument classification, and the complete identification and classification pipeline. Our local identification model utilizes the technique for resolving overlapping nodes discussed in Section 3.1.

To illustrate the gains achieved by different joint feature templates, we show in Table 2 the performance on core arguments achieved when adding to a local model the whole sequence and the frame features. The local models used for these experiments include only the standard features and selected pairs (Xue and Palmer, 2004) shown in Table 1, and do not include the additional (Pradhan et al., 2004) features listed in that table. As seen from Table 2, the whole sequence features reduced the error of the local argument classifier by 41% in F-measure and 55.7% in whole frame accuracy. The addition of the frame features achieved an additional 27% error reduction in F-measure. For argument identification, the improvement from joint features was smaller (15.7% error reduction). 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.

In Tables 3, 4, and 5, we report the performance of local and joint models for argument identification, classification, and integrated semantic role labeling, using all of the features in Table 1. All joint mod-

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

els for these experiments used the whole sequence and frame features. The largest improvement due to joint modeling is in classification for core arguments – 44.2% error reduction. On the complete pipeline the joint models achieved error reductions of 28% for CORE, 18.5% for ARGM, and 14.6% for ARGM-PLUS. The best published result on integrated argument identification and classification for core arguments for gold standard parse trees is 90.6% (Xue and Palmer, 2004), and our joint model achieves a 37% error reduction from it. The best reported result on all arguments is 89.4% (Pradhan et al., 2004) and our joint model reduces its error by 17%.

We also report preliminary results on automatic parses (see Table 6). We tested – but did not train on – automatic parse trees from Collins' parser. For approximately 11.2% 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. Although joint inference showed gains for automatic parses in this setting, we expect it would help more when we also train the models on automatic parses.

6 Discussion

During error analysis, we noticed that many of the difficult argument constituents were annotated with functional tags, which provide additional syntactic and semantic information in the TreeBank II gold standard parses. It is common to strip these tags from the trees in pre-processing since automatic parsers do not generally assign such tags. We hypothesized that such tags would be useful for the features like phrase type, path, subcat, etc. , which utilize the category of phrases, and ran a simple experiment where we left them in for training and testing with local classifiers. The results showed considerable gains for classification and slight gains for identification (see Table 7). We are aware of only one attempt (Blaheta and Charniak, 2000) to automatically tag phrases with the Treebank functional tags. Perhaps this task should receive more attention, in light of its potential application to SRL and other related areas. In future work, we plan to test if we can get comparable gains by using functional tagging with a learned classifier.

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.

References

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

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of ACL (NAACL)-2000*.

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

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.

Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of ACL-2002*.

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*.

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*.

Model	Classification		Identification	
	Fr. Acc.	F-Measure	Fr. Acc.	F-Measure
Local	90.3%	94.4 %	81.7%	94.3 %
+ Whole Sequence Features	95.7%	96.7 %	86.0%	95.0%
+ Frame Features	96.8%	**97.6** %	86.5%	**95.2**%

Table 2: Test set identification and classification results on CORE arguments when varying joint feature templates.

Model	CORE				ARGM			
	Fr. Acc.	F-Measure	Prec.	Recall	Fr. Acc.	F-Measure	Prec	Recall
Local	84.1%	95.1%	95.6%	94.6%	81.2%	95.1%	95.7%	94.5%
Joint	88.6%	96.1%	96.0%	96.2%	84.6%	95.6%	96.0%	95.2%

Table 3: Argument identification results on section 23 (gold-standard parses).

Model	CORE		ARGM		ARGMPLUS	
	Fr. Acc.	Arg. Acc.	Fr. Acc.	Arg. Acc.	Fr. Acc.	Arg. Acc.
Local	92.6%	95.7%	90.7%	96.0	85.2 %	93.4%
Joint	96.7%	**97.6**%	95.1%	**97.5**%	89.0 %	**94.9**%

Table 4: Argument classification results on section 23 (gold-standard parses).

Model	CORE		ARGM		ARGMPLUS	
	Fr. Acc.	F-Measure	Fr. Acc.	F-Measure	Fr. Acc.	F-Measure
Local	79.8%	91.8%	75.6%	91.9%	72.0%	89.7%
Joint	86.8%	**94.1**%	81.6%	**93.4**%	77.3%	**91.2**%

Table 5: Argument identification and classification results on section 23 (gold-standard parses).

Model	CORE	
	Fr. Acc.	F-Measure
Local	63.3 %	82.8 %
Joint	69.9 %	84.3 %

Table 6: Preliminary argument identification and classification results for CORE arguments on section 23 (Collins' automatic parses).

Condition	Identification		Classification	
	F-Measure	Fr. Acc.	Arg. Acc.	Fr. Acc.
With Functional Tags	94.9%	79.4%	**94.2**%	86.7%
No Functional Tags	94.7%	78.9%	90.7%	79.4%

Table 7: Results from functional tag experiment on the test set with standard features + selected pairs from Table 1 on ARGMPLUS