A Global Joint Model for Semantic Role Labeling

Kristina Toutanova[*]
Microsoft Research

Aria Haghighi[**]
University of California Berkeley

Christopher D. Manning[†]
Stanford University

We present a model for semantic role labeling that effectively captures the linguistic intuition that a semantic argument frame is a joint structure, with strong dependencies among the arguments. We show how to incorporate these strong dependencies in a statistical joint model with a rich set of features over multiple argument phrases. The proposed model substantially outperforms a similar state-of-the-art local model that does not include dependencies among different arguments.

We evaluate the gains from incorporating this joint information on the Propbank corpus, when using correct syntactic parse trees as input, and when using automatically derived parse trees. The gains amount to 24.1% error reduction on all arguments and 36.8% on core arguments for gold-standard parse trees on Propbank. For automatic parse trees, the error reductions are 8.3% and 10.3% on all and core arguments, respectively. We also present results on the CoNLL 2005 shared task data set. Additionally, we explore considering multiple syntactic analyses to cope with parser noise and uncertainty.

1. Introduction

Since the release of the FrameNet (Baker, Fillmore, and Lowe 1998) and Propbank (Palmer, Gildea, and Kingsbury 2005) corpora, there has been a large amount of work on statistical models for semantic role labeling. Most of this work relies heavily on **local** classifiers: ones that decide the semantic role of each phrase independently of the roles of other phrases.

However, linguistic theory tells us that a core argument frame is a *joint* structure, with strong dependencies between arguments. For instance, in the sentence

[*] One Microsoft Way, Redmond, WA 98052, USA. E-mail: kristout@microsoft.com.
[**] Department of Electrical Engineering and Computer Sciences, Soda Hall, Berkeley, CA 94720, USA. E-mail: aria42@cs.berkeley.edu.
[†] Department of Computer Science, Gates Building 1A, 353 Serra Mall, Stanford CA 94305, USA. E-mail: manning@cs.stanford.edu.

Submission received: 15 July 2006; Revised submission received: 1 May 2007; Accepted for publication: 19 June 2007.

[*Final-hour trading*]$_{\text{THEME}}$ *accelerated* [*to 108.1 million shares*]$_{\text{TARGET}}$ [*yesterday*]$_{\text{ARGM-TMP}}$, the first argument is the subject noun phrase *final-hour trading* of the active verb *accelerated*. If we did not consider the rest of the sentence, it would look more like an AGENT argument, but when we realize that there is no other good candidate for a THEME argument, because *to 108.1 million shares* must be a TARGET and *yesterday* is most likely ARGM-TMP, we can correctly label it THEME.

Even though previous work has modeled some correlations between the labels of parse tree nodes (see Section 2), many important phenomena have not been modeled. The key properties needed to model this joint structure are: (1) no finite Markov horizon assumption for dependencies among node labels, (2) features looking at the labels of multiple argument nodes and *internal features* of these nodes, and (3) a statistical model capable of incorporating these long-distance dependencies and generalizing well. We show how to build a joint model of argument frames, incorporating novel features into a discriminative log-linear model. This system achieves an error reduction of 24.1% on ALL arguments and 36.8% on CORE arguments over a state-of-the-art independent classifier for gold-standard parse trees on Propbank.

If we consider the linguistic basis for joint modeling of a verb's arguments (including modifiers), there are at least three types of information to be captured. The most basic is to limit occurrences of each kind of argument. For instance, there is usually at most one argument of a verb that is an ARG0 (agent), and although some modifier roles such as ARGM-TMP can fairly easily be repeated, others such as ARGM-MNR also generally occur at most once.[1] The remaining two types of information apply mainly to core arguments (the strongly selected arguments of a verb: ARG0–ARG5 in Propbank), which in most linguistic theories are modeled as belonging together in an argument frame (set of arguments). The information is only marginally useful for adjuncts (the ARGM arguments of Propbank), which are usually treated as independent realizational choices not included in the argument frame of a verb.

Firstly, many verbs take a number of different argument frames. Previous work has shown that these are strongly correlated with the word sense of the verb (Roland and Jurafsky 2002). If verbs were disambiguated for sense, the semantic roles of phrases would be closer to independent given the sense of the verb. However, because in almost all semantic role labeling work (including ours), the word sense is unknown and the model conditions only on the lemma, there is much joint information between arguments when conditioning only on the verb lemma. For example, compare:

(1) Britain's House of Commons passed a law on steroid use.

(2) The man passed the church on his way to the factory.

In the first case the noun phrase after *passed* is an ARG1, whereas in the second case it is a ARGM-LOC, with the choice governed by the sense of the verb *pass*. Secondly, even with same sense of a verb, different patterns of argument realization lead to joint information between arguments. Consider:

(3) The day that the ogre cooked the children is still remembered.

[1] The Propbank semantic role names which we use here are defined in Section 3.

(4) The meal that the ogre cooked the children is still remembered.

Despite both examples having an identical surface syntax, knowing that the ARG1 of *cook* is expressed by the initial noun *meal* in the second example gives evidence that *the children* is the ARG2 (beneficiary), not the ARG1 in this case.

Let us think of a graphical model over a set of m variables, one for each node in the parse tree t, representing the labels of the nodes and the dependencies between them. In order for a model over these variables to capture, for example, the statistical tendency of some semantic roles to occur at most once (e.g., that there is usually at most one constituent labeled AGENT), there must be a dependency link between any two variables. To estimate the probability that a certain node gets the role AGENT, we need to know if any of the other nodes were labeled with this role.

We propose such a model, with a very rich graphical model structure, which is globally conditioned on the observation (the parse tree).[2] Such a model is formally a Conditional Random Field (CRF) (Lafferty, McCallum, and Pereira 2001). However, note that in practice this term has previously been used almost exclusively to describe the restricted case of linear chain Conditional Markov Random Fields (sequence models) (Lafferty, McCallum, and Pereira 2001; Sha and Pereira 2003), or at least models that have strong Markov properties, which allow efficient dynamic programming algorithms (Cohn and Blunsom 2005). Instead, we consider a densely connected CRF structure, with no Markov properties, and use approximate inference by re-ranking the n-best solutions of a simpler model with stronger independence assumptions (for which exact inference is possible).

Such a rich graphical model can represent many dependencies but there are two dangers—one is that the computational complexity of training the model and searching for the most likely labeling given the tree can be prohibitive, and the other is that if too many dependencies are encoded, the model will over-fit the training data and will not generalize well. We propose a model which circumvents these two dangers and achieves significant performance gains over a similar local model that does not add any dependency arcs among the random variables. To tackle the efficiency problem, we adopt dynamic programming and re-ranking algorithms. To avoid overfitting we encode only a small set of linguistically motivated dependencies in features over sets of the random variables. Our re-ranking approach, like the approach to parse re-ranking of Collins (2000), employs a simpler model—a local semantic role labeling algorithm—as a first pass to generate a set of n likely complete assignments of labels to all parse tree nodes. The joint model is restricted to these n assignments and does not have to search the exponentially large space of all possible joint labelings.

2. Related Work

There has been a substantial amount of work on automatic semantic role labeling, starting with the statistical model of Gildea and Jurafsky (2002). Researchers have worked on defining new useful features, and different system architectures and models. Here we review the work most closely related to ours, concentrating on methods for incorporating joint information and for increasing robustness to parser error.

[2] That is, it defines a conditional distribution of labels of all nodes given the parse tree.

2.1 Methods for Incorporating Joint Information

Gildea and Jurafsky (2002) propose a method to model global dependencies by including a probability distribution over multi-sets of semantic role labels given a predicate. In this way the model can consider the assignment of all nodes in the parse tree and evaluate whether the set of realized semantic roles is likely. If a necessary role is missing or if an unusual set of arguments is assigned by the local model, this additional factor can correct some of the mistakes. The distribution over label multi-sets is estimated using interpolation of a relative frequency and a back-off distribution. The back-off distribution assumes each argument label is present or absent independently of the other labels, namely, it assumes a Bernoulli Naive Bayes model.

The most likely assignment of labels according to such a joint model is found approximately using re-scoring of the top $k = 10$ assignments according to a local model, which does not include dependencies among arguments. Using this model improves the performance of the system in F-measure from 59.2 to 62.85. This shows that adding global information improves the performance of a role labeling system considerably. However, the type of global information in this model is limited to label multi-sets. We will show that much larger gains are possible from joint modeling, adding richer sources of joint information using a more flexible statistical model.

The model of Pradhan, Hacioglu, et al. (2004, 2005) is a state-of-the-art model, based on Support Vector Machines, and incorporating a large set of structural and lexical features. At the heart of the model lies a local classifier, which labels each parse tree node with one of the possible argument labels or NONE. Joint information is integrated into the model in two ways:

Dynamic class context: Using the labels of the two nodes to the left as features for classifying the current node. This is similar to the Conditional Markov Models (CMM) often used in information extraction (McCallum, Freitag, and Pereira 2000). Notice that here the previous two nodes classified are not in general the previous two nodes assigned non-NONE labels. If a linear order on all nodes is imposed, then the previous two nodes classified most likely bear the label NONE.

Language model lattice re-scoring: Re-scoring of an *N*-best lattice with a trigram language model over semantic role label sequences. The target predicate is also part of the sequence.

These ways of incorporating joint information resulted in small gains over a baseline system using only the features of Gildea and Jurafsky (2002). The performance gain due to joint information over a system using all features was not reported. The joint information captured by this model is limited by the *n*-gram Markov assumption of the language model over labels. In our work, we improve the modeling of joint dependencies by looking at longer-distance context, by defining richer features over the sequence of labels and input features, and by estimating the model parameters discriminatively.

A system which can integrate longer-distance dependencies is that of Punyakanok et al. (2004) and Punyakanok, Roth, and Yih (2005). The idea is to build a semantic role labeling system that is based on local classifiers but also uses a global component that ensures that several linguistically motivated global constraints on argument frames are satisfied. The constraints are categorical and specified by hand. For example, one global constraint is that the argument phrases cannot overlap—that is, if a node is labeled with a non-NONE label, all of its descendants have to be labeled NONE. The proposed framework is integer linear programming (ILP), which makes it possible to find the most likely assignment of labels to all nodes of the parse tree subject to specified constraints. Solving the ILP problem is NP-hard but it is very fast in practice (Punyakanok et al. 2004). The authors report substantial gains in performance due to these global consistency constraints. This method was applied to improve the performance both of a system based on labeling syntactic chunks and one based on labeling parse tree nodes. Our work differs from that work in that our constraints are not categorical (either satisfied or not), but are rather statistical preferences, and that they are learned automatically based on features specified by the knowledge engineer. On the other hand, we solve the search/estimation problem through re-ranking and *n*-best search only approximately, not exactly.

So far we have mainly discussed systems which label nodes in a parse tree. Many systems that only use shallow syntactic information have also been presented (Hacioglu 2004; Punyakanok et al. 2004); using full syntactic parse information was not allowed in the CoNLL 2004 shared task on Semantic Role Labeling and description of such systems can be found in (Carreras and Màrquez 2004). Most systems which use only shallow syntactic information represent the input sentence as a sequence of tokens (words or phrases), which they label with a BIO tagging representation (beginning, inside, and outside argument labels) (Hacioglu 2004). Limited joint information is used by such systems, provided as a fixed size context of tags on previous tokens; for example, a length five window is used in the chunk-based system in (Pradhan, Hacioglu et al. 2005).

A method that models joint information in a different way was proposed by Cohn and Blunsom (2005). It uses a tree-structured CRF, where the statistical dependency structure is exactly defined by the edges in the syntactic parse tree. The only dependencies captured are between the label of a node and the label of each of its children. However, the arguments of a predicate can be arbitrarily far from each other in the syntactic parse tree and therefore a tree-CRF model is limited in its ability to model dependencies among different arguments. For instance, the dependency between *the meal* and *the children* for the sentence in example (4) will not be captured because these phrases are not in the same local tree according to Penn Treebank syntax.

2.2 Increasing Robustness to Parser Error

There have been multiple approaches to reducing the sensitivity of semantic role labeling systems to syntactic parser error. Promising approaches have been to consider multiple syntactic analyses—the top *k* parses from a single or multiple full parsers (Punyakanok, Roth, and Yih 2005), or a shallow parse and a full parse (Màrquez et al. 2005; Pradhan et al. 2005), or several types of full syntactic parses (Pradhan, Ward et al. 2005). Such techniques are important for achieving good performance: The top four systems in the CoNLL 2005 shared task competition all used multiple syntactic analyses (Carreras and Màrquez 2005).

These previous methods develop special components to combine the labeling decisions obtained using different syntactic annotation. The method of Punyakanok, Roth, and Yih (2005) uses ILP to derive a consistent set of arguments, each of which could be derived using a different parse tree. Pradhan, Ward et al. (2005) use stacking to train a classifier which combines decisions based on different annotations, and Màrquez et al. (2005) use special-purpose filtering and inference stages which combine arguments proposed by systems using shallow and full analyses.

Our approach to increasing robustness uses the top *k* parses from a single parser and is a simple general method to factor in the uncertainty of the parser by applying Bayesian inference. It is most closely related to the method described in Finkel, Manning, and Ng (2006) and can be seen as an approximation of that method.

We describe our system in detail by first introducing simpler local semantic role labeling models in Section 4, and later building on them to define joint models in Section 5. Before we start presenting models, we describe the data and evaluation measures used in Section 3. Readers can skip the next section and continue on to Section 4 if they are not interested in the details of the evaluation.

3. Data and Evaluation Measures

3.1 Data

For most of our experiments we used the February 2004 release of Propbank. We also report results on the CoNLL 2005 shared task data (Propbank I) in Section 6.2. For the latter, we used the standard CoNLL evaluation measures, and we refer readers to the description of that task for details of the evaluation (Carreras and Màrquez 2005). In this section we describe the data and evaluation measures we used for the February 2004 data. We use our own set of measures on the February 2004 data for three reasons. Firstly, we wish to present a richer set of measures, which can better illustrate the performance of the system on core arguments as against adjuncts and the performance on identifying versus classifying arguments. Secondly, we technically could not use the CoNLL measure on the February 2004 data, because this earlier data was not available in a format which specifies which arguments should have the additional R-ARGX labels used in the CoNLL evaluation.[3] Finally, these measures are better for comparison with early papers, because most research before 2005 did not distinguish referring arguments. We describe our argument-based measures in detail here in case researchers are interested in replicating our results for the February 2004 data.

For the February 2004 data, we used the standard split into training, development, and test sets—the annotations from sections 02–21 formed the training set, section 24 the development, and section 23 the test set. The set of argument labels considered is the set of core argument labels (ARG0 through ARG5) plus the modifier labels (see Figure 1). The training set contained 85,392 propositions, the test set 4,615, and the development set 2,626.

We evaluate semantic role labeling models on gold-standard parse trees and parse trees produced by Charniak's automatic parser (Charniak 2000). For gold-standard parse trees, we preprocess the trees to discard empty constituents and strip functional tags. Using the trace information provided by empty constituents is very useful for improving performance (Palmer, Gildea, and Kingsbury 2005; Pradhan, Ward et al. 2005), but we have not used this information so that we can compare our results to previous work and since automatic systems that recover it are not widely available.

3.2 Evaluation Measures

Since 2004, there has been a precise, standard evaluation measure for semantic role labeling, formulated by the organizers of the CoNLL shared tasks (Carreras and Màrquez

[3] Currently, a script which can convert original Propbank annotations to CoNLL format is available as part of the CoNLL software distribution.

Modifier Argument Labels	
ARGM-ADV: general-purpose	ARGM-LOC: location
ARGM-EXT: extent	ARGM-DIS: discourse connective
ARGM-NEG: negation marker	ARGM-MOD: modal verb
ARGM-CAU: cause	ARGM-TMP: time
ARGM-PNC: purpose	ARGM-MNR: manner
ARGM-DIR: direction	ARGM-PRD: predication
ARGM-REC: reciprocal	

Figure 1
Labels of modifying arguments occurring in Propbank.

2004, 2005). An evaluation script is also distributed as part of the provided software for the shared task and can be used to evaluate systems on Propbank I data.

For papers published between 2000 and 2005, there are several details of the evaluation measures for semantic role labeling that make it difficult to compare results obtained by different researchers, because researchers use their own implementations of evaluation measures, without making all the exact details clear in their papers. The first issue is the existence of arguments consisting of multiple constituents. In this case it is not clear whether partial credit is to be given for guessing only some of the constituents comprising the argument correctly. The second issue is whether the bracketing of constituents should be required to be recovered correctly, in other words, whether pairs of labelings, such as [*the*]$_{\text{ARG0}}$ [*man*]$_{\text{ARG0}}$ and [*the man*]$_{\text{ARG0}}$ are to be considered the same or not. If they are considered the same, there are multiple labelings of nodes in a parse tree that are equivalent. The third issue is that when using automatic parsers, some of the constituents that are fillers of semantic roles are not recovered by the parser. In this case it is not clear how various research groups have scored their systems (using headword match, ignoring these arguments altogether, or using exact match). If we vary the choice taken for these three issues, we can come up with many (at least eight) different evaluation measures, and these details are important, because different choices can lead to rather large differences in reported performance.

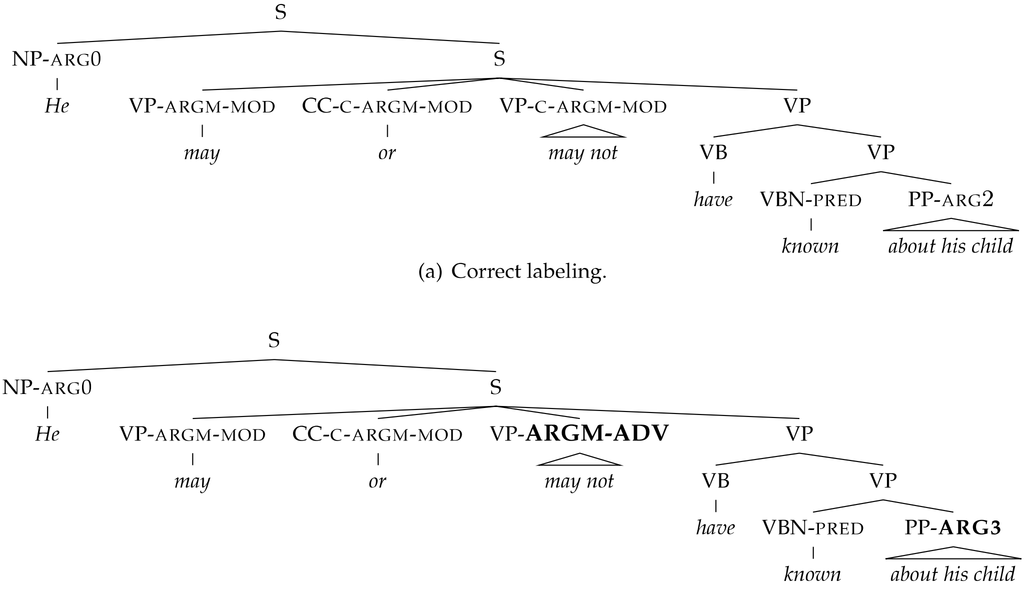
Here we describe in detail our evaluation measures for the results on the February 2004 data reported in this article. The measures are similar to the CoNLL evaluation measure, but report a richer set of statistics; the exact differences are discussed at the end of this section.

For both gold-standard and automatic parses we use one evaluation measure, which we call **argument-based evaluation**. To describe the evaluation measure, we will use as an example the correct and guessed semantic role labelings shown in Figures 2(a) and 2(b). Both are shown as labelings on parse tree nodes with labels of the form ARGX and C-ARGX. The label C-ARGX is used to represent multi-constituent arguments. A constituent labeled C-ARGX is assumed to be a continuation of the closest constituent to the left labeled ARGX. Our semantic role labeling system produces labelings of this form and the gold standard Propbank annotations are converted to this form as well.[4] The evaluation is carried out individually for each predicate and its associated argument frame. If a sentence contains several clauses, the several argument frames are evaluated separately.

[4] This representation is not powerful enough to represent all valid labelings of multi-constituent arguments, because it cannot represent the case where a new argument with label ARGX starts before a previous multi-constituent argument with the same label ARGX has finished. This case, however, is very rare.

Our argument-based measures do not require exact bracketing (if the set of words constituting an argument is correct, there is no need to know how this set is broken into constituents) and do not give partial credit for labeling correctly only some of several constituents in a multi-constituent argument. They are illustrated in Figure 2. For these measures, a semantic role labeling of a sentence is viewed as a labeling on sets of words. These sets can encompass several non-contiguous spans. Figure 2(c) gives the representation of the correct and guessed labelings shown in Figures 2(a) and 2(b), in the first and second rows of the table, respectively. To convert a labeling on parse tree nodes to this form, we create a labeled set for each possibly multi-constituent argument. All remaining sets of words are implicitly labeled with NONE. We can see that, in this way, exact bracketing is not necessary and also no partial credit is given when only some of several constituents in a multi-constituent argument are labeled correctly.

We will refer to word sets as "spans." To compute the measures, we are comparing a guessed set of labeled spans to a correct set of labeled spans. We briefly define the various measures of comparison used herein, using the example guessed and correct



(a) Correct labeling.

(b) Guessed labeling.

Location	Labeling
Correct	{0}-ARG0, {1, 2, 3, 4}-ARGM-MOD, {7, 8, 9}-ARG2
Guessed	{0}-ARG0, {1, 2}-ARGM-MOD, {3, 4}-ARGM-ADV {7, 8, 9}-ARG3

(c) Labelings of spans.

Task	CORE		COARSEARGM		ALL	
	F1	Acc.	F1	Acc.	F1	Acc.
Argument ID	100.0	100.0	57.1	0.0	57.1	0.0
Argument CLS	50.0	0.0	50.0	0.0	50.0	0.0
Argument ID&CLS	50.0	0.0	28.6	0.0	28.6	0.0

(d) Scoring measures.

Figure 2
Argument-based scoring measures for the guessed labeling.

labelings shown in Figure 2(c). All spans not listed explicitly are assumed to have label NONE. The scoring measures are illustrated in Figure 2(d).

The figure shows performance measures—F-Measure (F1) and Whole Frame Accuracy (Acc.)—across nine different conditions. When the sets of labeled spans are compared directly, we obtain the complete task measures, corresponding to the ID&CLS row and ALL column in Figure 2(d). We also define several other measures to understand the performance of the system on different types of labels. We measure the performance on identification (ID), classification (CLS), and the complete task (ID&CLS), when considering only the core arguments (CORE), all arguments but with a single ARGM label for the modifier arguments (COARSEARGM), and all arguments (ALL). This defines nine sub-tasks, which we now describe. For each of them, we compute the Whole Frame Accuracy and F-Measure as follows:

Whole Frame Accuracy (Acc.). This is the percentage of propositions for which there is an exact match between the proposed and correct labelings. For example, the whole frame accuracy for ID&CLS and ALL is 0, because the correct and guessed sets of labeled spans shown in Figure 2(c) do not match exactly. In the figures, "Acc." is always an abbreviation for this whole frame accuracy. Even though this measure has not been used extensively in previous work, we find it useful to track. Most importantly, 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. Moreover, a joint model for semantic role labeling optimizes Whole Frame Accuracy more directly than a local model does.

F-Measure (F_1). Because there may be confusion about what we mean by F-Measure in this multi-class setting, we define it here. F-Measure is defined as the harmonic mean of precision and recall: $f = \frac{2 \times p \times r}{p + r}$; $p = \frac{\textit{true positive}}{\textit{true positive} + \textit{false positive}}$; $r = \frac{\textit{true positive}}{\textit{true positve} + \textit{false negative}}$.

This formula uses the number of true positive, false positive, and false negative spans in a given guessed labeling. *True positive* is the number of spans whose correct label is one of the core or modifier argument labels (not NONE) and whose guessed label is the same as the correct label. *False positive* is the number of spans whose guessed label is non-NONE and whose correct label is different from the guessed label (possibly NONE). *False negative* is the number of spans whose correct label is non-NONE and whose guessed label is not the same as the correct one (possibly NONE). In the figures in this paper we show F-Measure multiplied by 100 so that it is in the same range as Whole Frame Accuracy.

Core Argument Measures (CORE). These measures score the system on core arguments only, without regard to modifier arguments. They can be obtained by first mapping all non-core argument labels in the guessed and correct labelings to NONE.

Coarse Modifier Argument Measures (COARSEARGM). Sometimes it is sufficient to know a given span has a modifier role, without knowledge of the specific role label. In addition, deciding exact modifier argument labels was one of the decisions with highest disagreement among annotators (Palmer, Gildea, and Kingsbury 2005). To estimate performance under this setting, we relabel all ARGM-X arguments to ARGM in the proposed and correct labeling. Such a performance measure was also used by Xue and Palmer (2004). Note that these measures do not exclude the core arguments but instead consider the core plus a coarse version of the modifier arguments. Thus for COARSEARGM

ALL we count $\{0\}$ as a true positive span, $\{1,2\}$, $\{3,4\}$, and $\{7,8,9\}$ as false positive, and $\{1,2,3,4\}$ and $\{7,8,9\}$ as false negative.

Identification Measures (ID). These measure how well we do on the ARG VS. NONE distinction. For the purposes of this evaluation, all spans labeled with a non-NONE label are considered to have the generic label ARG. For example, to compute CORE ID, we compare the following sets of labeled spans:

Correct: $\{0\}$-ARG, $\{7,8,9\}$-ARG

Guessed: $\{0\}$-ARG, $\{7,8,9\}$-ARG

The F-Measure is 1.0 and the Whole Frame Accuracy is 100%.

Classification Measures (CLS). These are performance on argument spans which were also guessed to be argument spans (but possibly the exact label was wrong). In other words, these measures ignore the ARG VS. NONE confusions. They ignore all spans, which were incorrectly labeled NONE, or incorrectly labeled with an argument label, when the correct label was NONE. This is different from "classification accuracy" used in previous work to mean the accuracy of the system in classifying spans when the correct set of argument spans is given. To compute CLS measures, we remove all spans from $S_{guessed}$ and $S_{correct}$ that do not occur in both sets, and compare the resulting sets. For example, to compute the ALL CLS measures, we need to compare the following sets of labeled spans:

Correct: $\{0\}$-ARG0, $\{7,8,9\}$-ARG2

Guessed: $\{0\}$-ARG0, $\{7,8,9\}$-ARG3

The rest of the spans were removed from both sets because they were labeled NONE according to one of the labelings and non-NONE according to the other. The F-Measure is .50 and the Whole Frame Accuracy is 0%.

As we mentioned before, we label and evaluate the semantic frame of every predicate in the sentence separately. It is possible for a sentence to contain several propositions—annotations of predicates occurring in the sentence. For example, in the sentence *The spacecraft faces a six-year journey to explore Jupiter*, there are two propositions, for the verbs *faces* and *explore*. These are:

[*The spacecraft*]$_{\text{ARG0}}$ [*faces*]$_{\text{PRED}}$ [*a six-year journey to explore Jupiter*]$_{\text{ARG1}}$.

[*The spacecraft*]$_{\text{ARG0}}$ *faces a six-year journey to* [*explore*]$_{\text{PRED}}$ [*Jupiter*]$_{\text{ARG1}}$.

Our evaluation measures compare the guessed and correct set of labeled spans for each proposition.

3.3 Relation to the CoNLL Evaluation Measure

The CoNLL evaluation measure (Carreras and Màrquez 2004, 2005) is almost the same as our argument-based measure. The only difference is that the CoNLL measure introduces an additional label type for arguments, of the form R-ARGX, used for referring expressions. The Propbank distribution contains a specification of which multi-constituent arguments are in a coreference chain. The CoNLL evaluation script considers these multi-constituent arguments as several separate arguments having different labels, where one argument has an ARGX label and the others have R-ARGX labels. The decision of which constituents were to be labeled with referring labels was made using a set of rules expressed with regular expressions.[5] A script that converts Propbank annotations to CoNLL format is available as part of the shared task software.

For example, in the following sentence, the CoNLL specification annotates the arguments of *began* as follows:

> [*The deregulation*]$_{\text{ARG1}}$ *of railroads* [*that*]$_{\text{R-ARG1}}$ [*began*]$_{\text{PRED}}$ *enabled shippers to bargain for transportation.*

In contrast, we treat all multi-constituent arguments in the same way, and do not distinguish coreferential versus non-coreferential split arguments. According to our argument-based evaluation, the annotation of the arguments of the verb *began* is:

> [*The deregulation*]$_{\text{ARG1}}$ *of railroads* [*that*]$_{\text{C-ARG1}}$ [*began*]$_{\text{PRED}}$ *enabled shippers to bargain for transportation.*

The difference between our argument based measure and the CoNLL evaluation measure is such that we cannot say that the value of one is always higher than the value of the other. Either measure could be higher depending on the kinds of errors made. For example, if the guessed labeling is: [*The deregulation*]$_{\text{ARG0}}$ *of railroads* [*that*]$_{\text{R-ARG1}}$ [*began*]$_{\text{PRED}}$ *enabled shippers to bargain for transportation*, the CoNLL script would count the argument *that* as correct and report precision and recall of .5, whereas our argument-based measure would not count any argument correct and report precision and recall of 0. On the other hand, if the guessed labeling is [*The deregulation*]$_{\text{ARG1}}$ *of railroads* [*that*]$_{\text{C-ARG1}}$ [*began*]$_{\text{PRED}}$ *enabled shippers to bargain for transportation*, the CoNLL measure would report a precision and recall of 0, whereas our argument-based measure would report precision and recall of 1. If the guessed labeling is [*The deregulation*]$_{\text{ARG1}}$ *of railroads* [*that*]$_{\text{R-ARG1}}$ [*began*]$_{\text{PRED}}$ *enabled shippers to bargain for transportation*, both measures would report precision and recall of 1. (For our argument-based measure it does not make sense to propose R-ARGX labels and we assume such labels would be converted to C-ARGX labels if they are after the phrase they refer to.) Nevertheless, overall we expect the two measures to yield very similar results.

4. Local Classifiers

A classifier is **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.

In defining our models, we use the standard separation of the task of semantic role labeling into **identification** and **classification** phases. Formally, let L denote a mapping of the nodes in a tree t to a label set of semantic roles (including NONE) with respect to a predicate v. Let $Id(L)$ be the mapping which collapses L's non-NONE values into ARG.

[5] The regular expressions look for phrases containing pronouns with part-of-speech tags WDT, WRB, WP, or WP$ (Xavier Carreras, personal communication).

Then, like the Gildea and Jurafsky (2002) system, we decompose the probability of a labeling L into probabilities according to an identification model P_{ID} and a classification model P_{CLS}.

$$P_{SRL}(L|t,v) = P_{ID}(Id(L)|t,v)P_{CLS}(L|t,v,Id(L)) \tag{1}$$

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. 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 do not 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).

We use log-linear models for multi-class classification for the local models. Because they produce probability distributions, identification and classification models can be chained in a principled way, as in Equation (1). The baseline features we used for the local identification and classification models are outlined in Figure 3. These features are a subset of the features used in previous work. The standard features at the top of the figure were defined by Gildea and Jurafsky (2002), and the rest are other useful lexical and structural features identified in more recent work (Surdeanu et al. 2003; Pradhan et al. 2004; Xue and Palmer 2004). We also incorporated several novel features which we describe next.

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 (Surdeanu et al. 2003; 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 concatenated with length of Path feature
Node-LCA Partial Path: Path from constituent to lowest common ancestor with predicate
PP Parent Head Word: If parent is a PP, parent's head word
PP NP Head Word/POS: For a PP, 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

Figure 3
Baseline features.

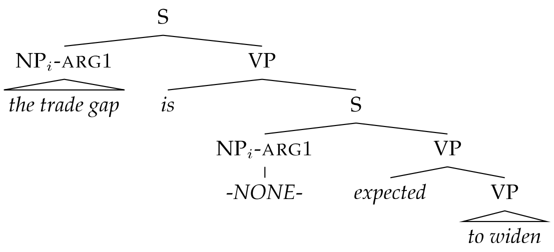


Figure 4
Example of displaced arguments.

4.1 Additional Features for Displaced Constituents

We found that a large source of errors for ARG0 and ARG1 stemmed from cases such as those illustrated in Figure 4, where arguments were dislocated by raising or control verbs. Here, the predicate, *expected*, does not have a subject in the typical position—indicated by the empty NP—because the auxiliary *is* has raised the subject to its current position. In order to capture this class of examples, we use a binary feature, MISSING SUBJECT, indicating whether the predicate is "missing" its subject, and use this feature in conjunction with the PATH feature, so that we learn typical paths to raised subjects conditioned on the absence of the subject in its typical position.[6]

In the particular case of Figure 4, there is another instance of an argument being quite far from its predicate. The predicate *widen* shares the phrase *the trade gap* with *expect* as an ARG1 argument. However, as *expect* is a raising verb, *widen*'s subject is not in its typical position either, and we should expect to find it in the same position as *expected*'s subject. This indicates it may be useful to use the path relative to *expected* to find arguments for *widen*. In general, to identify certain arguments of predicates embedded in auxiliary and infinitival VPs we expect it to be helpful to take the path from the **maximum extended projection** of the predicate—the highest VP in the chain of VPs dominating the predicate. We introduce a new path feature, PROJECTED PATH, which takes the path from the maximal extended projection to an argument node. This feature applies only when the argument is not dominated by the maximal projection (e.g., direct objects). These features also handle other cases of discontinuous and non-local dependencies, such as those arising due to control verbs. The performance gain from these new features was notable, especially in identification. The performance on ALL arguments for the model using only the features in Figure 3, and the model using the additional features as well, are shown in Figure 5. For these results, the constraint that argument phrases do not overlap was enforced using the algorithm presented in Section 4.2.

4.2 Enforcing the Non-Overlapping Constraint

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

[6] We consider a verb to be missing its subject if the highest VP in the chain of VPs dominating the verb does not have an NP or S(BAR) as its immediate left sister.

Task	Features in Figure 3		+ Additional Features	
	F1	Acc.	F1	Acc.
ID	91.7	77.6	92.4	79.0
CLS	95.6	90.6	95.7	90.8
ID&CLS	87.6	70.8	88.4	72.3

Figure 5
Performance of local classifiers on ALL arguments, using the features in Figure 3 only and using the additional local features. Using gold standard parse trees on Section 23.

probabilities according to the two models, as in Equation (1). Because 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).

$$P^{\ell}_{SRL}(L|t,v) = \prod_{n_i \in t} P_{ID}(Id(l_i)|t,v) \prod_{n_i \in t} P_{CLS}(l_i|t,v,Id(l_i)) \tag{2}$$

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.

When labeling parse tree nodes, previous work has either used greedy algorithms to find a non-overlapping assignment, or the general-purpose ILP approach of Punyakanok et al. (2004). For labeling chunks an exact algorithm based on shortest paths was proposed in Punyakanok and Roth (2001). Its complexity is quadratic in the length of the sentence.

Here we describe a faster 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 having ARG descendants.

Subsequently, 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 subtree, using already computed assignments for its children. Suppose we want the most likely consistent assignment for subtree t with child trees $t_1, \ldots, t_k$ each storing the most likely consistent assignment of its nodes, as well as the log-probability of the ALLNONE assignment: the assignment of NONE to all nodes in the tree. 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 child subtrees $t_1, \ldots, t_k$ plus the log-probability of assigning the node t to NONE.
- The sum of the log-probabilities of the ALLNONE assignments of $t_1, \ldots, t_k$ plus the log-probability of assigning the node t to ARG.

Enforcing constraint	CORE		ALL	
	F1	Acc.	F1	Acc.
No	90.3	81.2	88.3	71.8
Yes	90.5	81.4	88.4	72.3

Figure 6
Performance of local model on ALL arguments when enforcing the non-overlapping constraint or not.

The log-probability of the ALLNONE assignment for a tree *t* is the log-probability of assigning the root node of *t* to NONE plus the sum of the log-probabilities of the ALLNONE assignments of the child subtrees of *t*.

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.

The complexity of this algorithm is linear in the number of nodes in the parse tree, which is usually much less than the square of the number of words in the sentence (l^2), the complexity of the Punyakanok and Roth (2001) algorithm. For example, for a binary-branching parse tree, the number of nodes is approximately $2l$. The speedup is due to the fact that when we label parse tree nodes, we make use of the bracketing constraints imposed by the parse tree. The shortest path algorithm proposed by Punyakanok and Roth can also be adapted to achieve this lower computational complexity.

It turns out that enforcing the non-overlapping constraint does not lead to large gains in performance. The results in Figure 5 are from models that use the dynamic program for selecting non-overlapping arguments. To evaluate the gain from enforcing the constraint, Figure 6 shows the performance of the same local model using all features, when the dynamic program is used versus when a most likely possibly overlapping assignment is chosen in testing.

The local model with basic plus additional features is our first pass model used in re-ranking. The non-overlapping constraint is enforced using the dynamic program. This is a state-of-the-art model. Its F-Measure on ALL arguments is 88.4 according to our argument-based scoring measure. This is very similar to the best reported results (as of 2004) using gold-standard parse trees without null constituents and functional tags: 89.4 F-Measure reported for the Pradhan et al. (2004) model.[7]

A more detailed analysis of the results obtained by the local model is given in Figure 7(a), and the two confusion matrices in Figures 7(b) and 7(c), which display the number of errors of each type that the model made. The first confusion matrix concentrates on CORE arguments and merges all modifying argument labels into a single ARGM label. The second concentrates on confusions among modifying arguments.

From the confusion matrix in Figure 7(b), we can see that the largest number of errors are confusions of argument labels with NONE. The number of confusions between pairs of core arguments is low, as is the number of confusions between core and modifier labels. If we ignore the column and row corresponding to NONE in Figure 7(b), the number of off-diagonal entries is very small. This corresponds to the high F-Measures

[7] The results in Pradhan et al. (2004) are based on a measure which is more lenient than our argument-based scoring (Sameer Pradhan, personal communication, July 2005). Our estimated performance using his measure is 89.9.

Task	CORE		COARSEARGM		ALL	
	F1	Acc.	F1	Acc.	F1	Acc.
ID	92.3	83.7	92.4	79.0	92.4	79.0
CLS	98.0	96.8	98.1	95.9	95.7	90.8
ID&CLS	**90.5**	**81.4**	90.6	76.2	**88.4**	**72.3**

(a) Summary performance results.

	Guessed								
Correct	ARG0	ARG1	ARG2	ARG3	ARG4	ARG5	ARGM	NONE	F-Measure
ARG0	2912	22	1	0	0	0	4	248	91.7
ARG1	69	3964	15	1	1	0	12	302	91.8
ARG2	7	25	740	3	2	0	9	151	82.4
ARG3	1	5	3	83	1	0	5	36	70.3
ARG4	0	1	3	0	63	0	0	7	88.1
ARG5	0	0	0	0	0	5	0	0	100.0
ARGM	0	7	10	0	0	0	2907	322	91.0
NONE	173	248	87	15	2	0	204	–	–

(b) COARSEARGM confusion matrix.

	Guessed														
Correct	ADV	CAU	DIR	DIS	EXT	LOC	MNR	MOD	NEG	PNC	PRD	REC	TMP	CORE	NONE	F_1
ADV	295	3	0	13	3	10	35	0	0	5	0	0	20	1	51	71.3
CAU	0	48	0	1	0	2	3	0	0	2	0	0	3	0	6	81.4
DIR	0	0	40	0	0	0	6	0	0	0	0	0	1	2	25	61.1
DIS	13	0	0	214	0	3	2	0	0	0	0	0	8	0	31	79.9
EXT	2	0	0	1	17	0	5	0	0	0	0	0	0	2	5	63.0
LOC	4	0	0	2	0	251	3	0	0	2	1	0	8	1	45	77.5
MNR	17	0	5	0	2	12	196	0	0	0	0	0	4	5	66	65.8
MOD	0	0	0	0	0	0	0	453	0	0	0	0	0	0	2	99.4
NEG	0	0	0	0	0	0	0	0	200	0	0	0	0	0	2	99.0
PNC	4	2	0	0	0	1	0	0	0	59	0	0	5	3	26	64.8
PRD	1	0	1	0	0	0	0	0	0	1	1	0	0	1	0	28.6
REC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.0
TMP	23	0	0	4	0	11	3	0	1	1	0	0	874	2	61	88.7
CORE	4	0	2	2	0	0	6	0	0	7	0	0	9	7927	744	92.3
NONE	28	0	9	28	0	41	30	3	1	5	0	0	59	525	–	–

(c) Modifier arguments confusion matrix.

Figure 7
Performance measures for local model using all local features and enforcing the non-overlapping constraint. Results are on Section 23 using gold standard parse trees.

on COARSEARGM CLS and CORE CLS, 98.1 and 98.0 respectively, shown in Figure 7(a). The number of confusions of argument labels with NONE, shown in the NONE column, is larger than the number of confusions of NONE with argument labels, shown in the NONE row. This shows that the model generally has higher precision than recall. We experimented with the precision–recall tradeoff but this did not result in an increase in F-Measure.

From the confusion matrix in Figure 7(c) we can see that the number of confusions between modifier argument labels is higher than the number of confusions between core argument labels. This corresponds to the ALL CLS F-Measure of 95.7 versus the CORE CLS F-Measure of 98.0. The per-label F-Measures in the last column show that the performance on some very frequent modifier labels is in the low sixties or seventies. The confusions between modifier labels and NONE are quite numerous.

Thus, to improve the performance on CORE arguments, we need to improve recall without lowering precision. In particular, when the model is uncertain which of several likely CORE labels to assign, we need to find additional sources of evidence to improve its confidence. To improve the performance on modifier arguments, we also need to lower the confusions among different modifier arguments. We will see that our joint model improves the overall performance mainly by improving the performance on

CORE arguments, through increasing recall and precision by looking at wider sentence context.

4.3 On Split Constituents

As discussed in Section 3, multiple constituents can be part of the same semantic argument as specified by Propbank. An automatic system that has to recover such information needs to have a way of indicating when multiple constituents labeled with the same semantic role are a part of the same argument. Some researchers (Pradhan et al. 2004; Punyakanok et al. 2004) have chosen to make labels of the form C-ARGX distinct argument labels that become additional classes in a multi-class constituent classifier. These C-ARGX are used to indicate continuing arguments as illustrated in the two trees in Figure 2. We chose to not introduce additional labels of this form, because they might unnecessarily fragment the training data. Our automatic classifiers label constituents with one of the core or modifier semantic role labels, and a simple post-processing rule is applied to the output of the system to determine which constituents that are labeled the same are to be merged as the same argument. The post-processing rule is the following: For every constituent that bears a core argument label ARGX, if there is a preceding constituent with the same label, re-label the current constituent C-ARGX. Therefore, according to our algorithm, all constituents having the same core argument label are part of the same argument, and all constituents having the same modifier labels are separate arguments by themselves. This rule is fairly accurate for core arguments but is not always correct; it fails more often on modifier arguments. An evaluation of this rule using the CoNLL data set and evaluation measure shows that our upper bound in performance because of this rule is approximately 99.0 F-Measure on ALL arguments.

5. Joint Classifiers

We proceed to describe our models incorporating dependencies between labels of nodes in the parse tree. As we discussed briefly before, the dependencies we would like to model are highly non-local. A factorized sequence model that assumes a finite Markov horizon, such as a chain CRF (Lafferty, McCallum, and Pereira 2001), would not be able to encode such dependencies. We define a CRF with a much richer dependency structure.

5.1 Form of the Joint Classifiers

Motivation 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 a 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 Figure 8(a), for relatively small values of n, our re-ranking approach does not present a serious bottleneck to performance. We used a value of $n = 10$ for training. In Figure 8(a) we can see that if we could pick, using an oracle, the best assignment out of the top 10 assignments according to the local model, we would achieve an F-Measure of 97.3 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 $n = 10$ 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}, using an exact dynamic programming algorithm, which is a direct generalization of the algorithm for finding the top non-overlapping assignment described in Section 4.2.

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 the 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

$$P^r_{SRL}(L|t, v) = \frac{e^{<\Phi(t,v,L),W>}}{\sum_{j=1}^{N} e^{<\Phi(t,v,L_j),W>}} \tag{3}$$

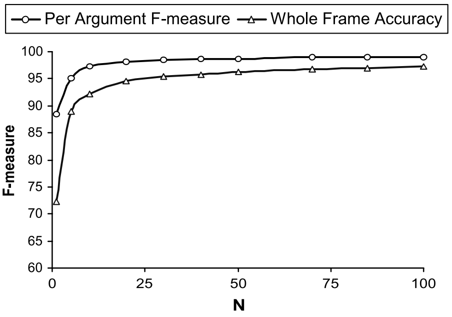
The score of an assignment L not in the top n is zero. We train the model to maximize the sum of log-likelihoods of the best assignments minus a quadratic regularization term. In this framework, we can define arbitrary features of labeled trees that capture general properties of predicate–argument structure.

5.2 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 9. We model dependencies not only between the label of a

n	CORE		ALL	
	F1	Acc.	F1	Acc.
1	90.5	81.4	88.4	72.3
5	96.9	93.5	95.0	88.9
10	98.0	95.6	97.3	92.1
30	98.9	97.5	98.4	95.4
100	99.4	98.8	99.0	97.2

(a) Tabular form.



(b) Graphical form.

Figure 8
Oracle upper bounds for top n non-overlapping assignments from local model on CORE and ALL arguments, using gold-standard parse trees.

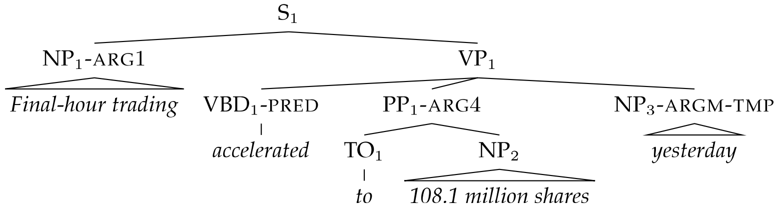


Figure 9
An example tree from Propbank with semantic role annotations, for the sentence *Final-hour trading accelerated to 108.1 million shares yesterday.*

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.

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 $[n_1, l_1, \ldots, v_{PRED}, \ldots, n_m, l_m]$ (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—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. Because 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 9 is then:

$$[\text{NP}_1\text{-ARG1}, \text{VBD}_1\text{-PRED}, \text{PP}_1\text{-ARG4}, \text{NP}_3\text{-ARGM-TMP}]$$

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 and a feature with the generic back-off ARG label are created. This is similar to adding features from identification and classification models. In the case of the example candidate argument sequence provided, for the node NP$_1$ we have the features:

$$\{(\text{NP}\uparrow\text{S}\downarrow\text{VP}\downarrow\text{VBD})\text{-ARG1}, (\text{NP}\uparrow\text{S}\downarrow\text{VP}\downarrow\text{VBD})\text{-ARG}\}$$

When comparing a local and a joint model, we use the same set of local feature templates in the two models. If these were the only features that a joint model used, we would expect its performance to be roughly the same as the performance of a local model. This is because the two models will in fact be in the same parametric family but will only differ slightly in the way the parameters are estimated. In particular, the likelihood of an assignment according to the joint model with local features will differ from the likelihood of the same assignment according to the local model only in the denominator (the partition function). The joint model sums over a few likely assignments in the denominator, whereas the local model sums over all assignments; also, the joint model does not treat the decomposition into identification and classification models in exactly the same way as the local model.

Whole Label Sequence Features. 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 nodes to fill 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. Two templates for whole label sequences were added: one having the predicate voice only, and another also including the predicate lemma. These templates are instantiated as follows for the example candidate argument sequence:

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

We also add variants of these templates that use a generic ARG label instead of specific labels for the arguments. These feature templates have the effect of counting the number of arguments to the left and right of the predicate, which provides useful global information about argument structure. A local model is not able to represent the count of arguments since the label of each node is decided independently. This feature can very directly and succinctly encode preferences for required arguments and expected number of arguments.

As previously observed (Pradhan et al. 2004), including modifying arguments in sequence features is not helpful. This corresponds to the standard linguistic understanding that there are no prevalent constraints on the position or presence of adjuncts in an argument frame, and was confirmed in our experiments. We redefined the whole label sequence features to exclude modifying arguments.

The whole label sequence features are the first type of features we add to relax the independence assumptions of the local model. Because these features look at the sequence of labels of all arguments, they capture joint information. There is no limit on the length of the label sequence and thus there is no *n*-gram Markov order independence assumption (in practice the candidate argument sequences in the top *n* complete assignments are rarely more than 7 nodes long). Additionally, the nodes in the candidate argument sequences are in general not in the same local tree in the syntactic analysis and a tree-CRF model (Cohn and Blunsom 2005) would not be able to encode these dependencies.

Joint Syntactic–Semantic Features. This class of features is similar to the whole label sequence features, but in addition to labels of argument nodes, it includes syntactic features of the nodes. These features can capture the joint mapping from the syntactic realization of the predicate's arguments to its semantic frame. The idea of these features is to capture knowledge about the label of a constituent given the syntactic realization and labels of all other arguments of the verb. This is helpful in capturing syntactic alternations, such as the dative alternation. For example, consider the sentence (*i*) [*Shaw Publishing*]$_{\text{ARG0}}$ [*offered*]$_{\text{PRED}}$ [*Mr. Smith*]$_{\text{ARG2}}$ [*a reimbursement*]$_{\text{ARG1}}$ and the alternative realization (*ii*) [*Shaw Publishing*]$_{\text{ARG0}}$ [*offered*]$_{\text{PRED}}$ [*a reimbursement*]$_{\text{ARG1}}$ [*to Mr. Smith*]$_{\text{ARG2}}$.

When classifying the NP in object position, it is useful to know whether the following argument is a PP. If it is, 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 candidate argument node, its phrase type and label. For example, the instantiations of such templates in (*ii*), including only the predicate voice or also the predicate lemma, would be:

[voice:active, NP-ARG0, PRED, NP-ARG1, PP-ARG2]

[voice:active,lemma:offer, NP-ARG0, PRED, NP-ARG1, PP-ARG2]

We experimented with extracting several kinds of features from each argument node and found that the phrase type and the head of a directly dominating PP—if one exists—were most helpful.

Local models normally consider only features of the phrase being classified in addition to features of the predicate. They cannot take into account the features of other argument nodes, because they are only given the input (parse tree), and the identity of the argument nodes is unknown. It is conceivable that a local model could condition on the features of all nodes in the tree but the number of parameters (features) would be extremely large. The joint syntactic–semantic features proposed here encode important dependencies using a very small number of parameters, as we will show in Section 5.4.

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 we use a joint model, we are able to use information about other argument nodes when labeling a node.

Repetition Features. We also add features that detect 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. Variants of this feature template also indicate whether all repeated arguments are sisters in the parse tree, or whether all repeated arguments are adjacent in terms of word spans. These features can provide robustness to parser errors, making it more likely to assign the same label to adjacent phrases that may have been incorrectly split by the parser. In Section 5.4 we report results from the joint model and an ablation study to determine the contribution of each of the types of joint features.

5.3 Applying Joint Models in Testing

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}. The local model P^{ℓ}_{SRL} is used to generate 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. Because 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 determining the amount of influence the local score has on the final score (we found $\alpha = 1.0$ to work best). 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:

$$\text{argmax}_{L \in L_1,\ldots,L_n}\ \alpha \log P^{\ell}_{SRL}(L|t,v) + \log P^{r}_{SRL}(L|t,v) \qquad (4)$$

5.4 Joint Model Results

We compare the performance of joint re-ranking models and local models. We used $n = 10$ joint assignments for training re-ranking models, and $n = 15$ for testing. The weight α of the local model was set to 1. Using different numbers of joint assignments in training and testing is in general not ideal, but due to memory requirements, we could not experiment with larger values of n for training.

Figure 10 shows the summary performance of the local model (LOCAL), repeated from earlier figures, a joint model using only local features (JOINTLOCAL), a joint model using local + whole label sequence features (LABELSEQ), and a joint model using all described types of features (ALLJOINT). The evaluation is on gold-standard parse trees. In addition to performance measures, the figure shows the number of binary features included in the model. The number of features is a measure of the complexity of the hypothesis space of the parametric model.

We can see that a joint model using only local features outperforms a local model by .5 points of F-Measure. The joint model using local features estimates the feature weights only using the top n consistent assignments, thus making the labels of different nodes non-independent according to the estimation procedure, which may be a cause of the improved performance. Another factor could be that the model JOINTLOCAL is a combination of two models as specified in Equation (4), which may lead to gains (as is usual for classifier combination).

The label sequence features added in Model LABELSEQ result in another 1.5 points jump in F-Measure on all arguments. An additional .8 gain results from the inclusion of syntactic–semantic and repetition features. The error reduction of model ALLJOINT

Model	# Features	CORE		COARSEARGM		ALL	
		F1	Acc.	F1	Acc.	F1	Acc.
LOCAL	5,201K	90.5	81.4	90.6	76.2	88.4	72.3
JOINTLOCAL	2,193K	90.9	82.6	91.1	78.3	88.9	74.3
LABELSEQ	2,357K	92.9	86.1	92.6	81.4	90.4	77.0
ALLJOINT	2,811K	94.0	87.6	93.4	82.7	**91.2**	**78.3**

Figure 10
Performance of local and joint models on ID&CLS on Section 23, using gold-standard parse trees. The number of features of each model is shown in thousands.

over the local model is 36.8% in CORE arguments F-Measure, 33.3% in CORE arguments whole frame accuracy, 24.1% in ALL arguments F-Measure, and 21.7% in ALL arguments whole frame accuracy. All differences in ALL arguments F-Measure are statistically significant according to a paired Wilcoxon signed rank test. JOINTLOCAL is significantly better than LOCAL ($p < .001$), LABELSEQ is significantly better than JOINTLOCAL ($p < .001$), and ALLJOINT is significantly better than LABELSEQ ($p < .001$). We performed the Wilcoxon signed rank test on per-proposition ALL arguments F-Measure for all models.

We also note that the joint models have fewer features than the local model. This is due to the fact that the local model has seen many more negative examples and therefore more unique features. The joint features are not very numerous compared to the local features in the joint models. The ALLJOINT model has around 30% more features than the JOINTLOCAL model.

These experiments showed that the label sequence features were very useful, especially on CORE arguments, increasing the F-Measure on these arguments by two points when added to the JOINTLOCAL model. This shows that even though the local model is optimized to use a large set of features and achieve state-of-the-art performance, it is still advantageous to model the joint information in the sequence of labels in a predicate's argument frame. Additionally, the joint syntactic–semantic features improved performance further, showing that when predicting the label of an argument, it is useful to condition on the features of other arguments, in addition to their labels.

A more detailed analysis of the results obtained by the joint model ALLJOINT is given in Figure 11(a) (Summary results), and the two confusion matrices in Figures 11(b) and 11(c), which display the number of errors of each type that the model made. The first confusion matrix concentrates on CORE arguments and merges all modifying argument labels into a single ARGM label. The second confusion matrix concentrates on confusions among modifying arguments. This figure can be compared to Figure 7, which summarizes the results for the local model in the same form. The biggest differences are in the performance on CORE arguments, which can be seen by comparing the confusion matrices in Figures 7(b) and 11(b). The F-Measure on each of the core argument labels has increased by at least three points: the F-Measure on ARG2 by 5.7 points, and the F-Measure on ARG3 by eight points. The confusions of core argument labels with NONE have gone down significantly, and also there is a large decrease in the confusions of NONE with ARG1. There is generally a slight increase in F-Measure on modifier labels as well, but the performance on some of the modifier labels has gone down. This makes sense because our joint features are targeted at capturing the dependencies among core arguments. There may be useful regularities for modifier arguments as well, but capturing them may require different joint feature templates.

Figure 12 lists the frequency with which each of the top *k* assignments from the LOCAL model was ranked first by the re-ranking model ALLJOINT. For example, for 84.1% of the propositions, the re-ranking model chose the same assignment that the LOCAL model would have chosen. The second best assignment according to the LOCAL model was promoted to first 8.6% of the time. The figure shows statistics for the top ten assignments only. The rest of the assignments, ranked 11 through 15, were chosen as best by the re-ranking model for a total of 0.3% of the propositions.

The labeling of the tree in Figure 9 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.

6. Semantic Role Labeling of Automatic Parses

We now evaluate our models when trained and tested using automatic parses produced by Charniak's parser. The Propbank training set Sections 2–21 is also the training set of the parser. The performance of the parser is therefore better on the training set. When the constituents of an argument do not have corresponding constituents in an automatically produced parse tree, it will be very hard for a model to get the semantic role labeling correct. However, this is not impossible and systems which are more robust to parser error have been proposed (Pradhan et al. 2005; Màrquez et al. 2005). Our system can also theoretically guess the correct set of words by labeling a set of constituents that cover

Task	CORE		COARSEARGM		ALL	
	F1	Acc.	F1	Acc.	F1	Acc.
ID	95.6	89.2	95.0	85.0	95.0	85.0
CLS	98.3	97.6	98.3	96.6	96.0	91.4
ID&CLS	94.0	87.6	93.4	82.7	91.2	78.3

(a) Summary performance results.

	Guessed								
Correct	ARG0	ARG1	ARG2	ARG3	ARG4	ARG5	ARGM	NONE	F-Measure
ARG0	3025	23	3	0	0	0	5	131	94.8
ARG1	39	4147	17	1	0	0	7	153	95.0
ARG2	6	34	821	2	2	0	9	63	88.1
ARG3	0	7	2	99	0	0	5	21	78.3
ARG4	0	0	2	0	68	0	0	4	93.8
ARG5	0	0	0	0	0	5	0	0	100.0
ARGM	0	14	11	1	0	0	2975	245	92.0
NONE	124	137	70	16	1	0	217	–	–

(b) COARSEARGM confusion matrix.

	Guessed														
Correct	ADV	CAU	DIR	DIS	EXT	LOC	MNR	MOD	NEG	PNC	PRD	REC	TMP	CORE	NONE	F_1
ADV	307	2	0	13	3	11	39	0	0	4	0	0	20	3	34	72.9
CAU	2	47	0	1	0	2	3	0	0	2	0	0	3	0	5	80.3
DIR	0	0	44	0	0	0	6	0	0	0	0	0	0	4	20	64.2
DIS	12	0	0	217	0	2	2	0	0	0	0	0	7	0	31	81.0
EXT	2	0	0	1	16	0	3	0	0	0	0	0	0	4	6	61.5
LOC	4	0	1	3	0	250	4	0	0	1	1	0	10	2	41	76.8
MNR	17	0	5	0	1	13	224	0	0	1	0	0	6	4	36	69.7
MOD	0	0	0	0	0	0	0	453	0	0	0	0	0	0	2	99.4
NEG	0	0	0	0	0	0	0	0	199	0	0	0	0	0	3	98.8
PNC	3	2	1	0	0	0	0	0	0	74	0	0	3	4	13	76.3
PRD	1	0	1	0	0	0	0	0	0	0	2	0	0	0	1	50.0
REC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.0
TMP	23	0	0	4	0	11	7	0	1	1	0	0	877	5	51	89.1
CORE	4	0	2	1	0	0	5	0	0	6	0	0	8	8303	372	95.6
NONE	31	1	9	25	0	45	43	3	1	5	0	0	54	348	–	–

(c) Modifier arguments confusion matrix.

Figure 11
Performance measures for joint model using all features (AllJoint). Results are on Section 23 using gold-standard parse trees.

Rank	1	2	3	4	5	6	7	8	9	10
Chosen	84.1	8.6	2.7	1.6	0.8	0.4	0.5	0.3	0.2	0.4

Figure 12
Percentage of test set propositions for which each of the top ten assignments from the Local model was selected as best by the joint model AllJoint.

Set	Constituents	Propositions
Training	2.9%	7.4%
Development	7.1%	17.3%
Test	6.2%	15.7%

Figure 13
Percentage of argument constituents that are not present in the automatic parses of Charniak's parser. **Constituents** shows the percentage of missing constituents and **Propositions** shows the percentage of propositions that have missing constituents.

the argument words, but we found that this rarely happens in practice. Figure 13 shows the percentage of argument constituents that are missing in the automatic parse trees produced by Charniak's parser. We can see that the percentage of missing constituents is quite high.

We report local and joint model results in Figures 14(a) and 14(b), respectively. As for gold-standard parses, we test on all arguments regardless of whether they correspond to constituents that have been recovered by the parser and use the same measures detailed in Section 3.2. We also compare the confusion matrices for the local and joint models, ignoring the confusions among modifier argument labels (COARSEARGM setting) in Figure 15. The error reduction of the joint over the local model is 10.3% in CORE arguments F-Measure and 8.3% in ALL arguments F-Measure.

6.1 Using Multiple Automatic Parse Guesses

Semantic role labeling is very sensitive to the correctness of the given parse tree, as the results show. If an argument does not correspond to any constituent in a parse tree, or a constituent exists but is not attached or labeled correctly, our model will have a very hard time guessing the correct labeling.

Thus, if the syntactic parser makes errors, these errors influence directly the semantic role labeling system. The theoretically correct way to propagate the uncertainty of the syntactic parser is to consider (sum over) multiple possible parse trees, weighted by their likelihood. In Finkel, Manning, and Ng (2006), this is approximated by sampling

Task	CORE		COARSEARGM		ALL	
	F1	Acc.	F1	Acc.	F1	Acc.
ID	82.2	67.3	81.7	60.2	81.7	60.2
CLS	98.0	97.1	98.0	96.1	95.7	91.7
ID&CLS	80.6	65.6	80.1	58.1	78.2	55.3

(a) Summary performance of local model.

Task	CORE		COARSEARGM		ALL	
	F1	Acc.	F1	Acc.	F1	Acc.
ID	84.2	70.5	83.4	64.0	83.4	64.0
CLS	98.0	97.3	98.1	96.4	95.9	92.0
ID&CLS	82.6	69.1	81.8	62.0	80.0	59.1

(b) Summary performance of joint model.

Figure 14
Comparison of local and joint model results on Section 23 using Chaniak's automatic parser.

	Guessed								
Correct	ARG0	ARG1	ARG2	ARG3	ARG4	ARG5	ARGM	NONE	F-Measure
ARG0	2710	23	3	0	0	0	5	446	86.1
ARG1	62	3423	14	0	0	0	10	855	79.7
ARG2	6	23	610	0	0	0	7	291	69.0
ARG3	1	4	2	62	1	0	4	60	55.4
ARG4	0	1	0	0	50	0	1	22	73.5
ARG5	0	0	0	0	0	2	0	3	57.1
ARGM	2	9	8	3	0	0	2446	778	78.7
NONE	330	740	194	25	11	0	497	–	–

(a) COARSEARGM confusion matrix of local model.

	Guessed								
Correct	ARG0	ARG1	ARG2	ARG3	ARG4	ARG5	ARGM	NONE	F-Measure
ARG0	2808	32	4	0	0	0	4	339	88.2
ARG1	34	3579	18	0	0	0	9	724	81.9
ARG2	5	34	653	1	1	0	5	238	70.6
ARG3	1	5	4	72	1	0	2	49	59.3
ARG4	0	0	1	0	54	0	0	19	73.0
ARG5	0	0	0	0	0	4	0	1	88.9
ARGM	2	15	10	2	1	0	2519	697	79.5
NONE	329	712	223	34	17	0	552	–	–

(b) COARSEARGM confusion matrix of joint model.

Figure 15
COARSEARGM argument confusion matrices for local and joint model using Charniak's automatic parses.

parse trees. We implement this idea by an argmax approximation, using the top k parse trees from the parser of Charniak (2000).

We use these alternative parses as follows: Suppose $t_1, \ldots, t_k$ are trees for sentence s with probabilities $P(t_i|s)$ given by the parser. Then for a fixed predicate v, let L_i denote the best joint labeling of tree t_i, with score $score_{SRL}(L_i|t_i)$ according to our final joint model. Then we choose the labeling L which maximizes

$$\text{argmax}_{i \in \{1,\ldots,k\}} \beta \log P(t_i|S) + score_{SRL}(L_i|t_i)$$

This method of using multiple parse trees is very simple to implement and factors in the uncertainty of the parser to some extent. However, according to this method (due to the argmax operation) we are choosing a single parse and a complete semantic frame derived from that parse. Other methods are able to derive different arguments of the semantic frame from different syntactic annotations which may make them more robust (Màrquez et al. 2005; Pradhan, Ward et al. 2005; Punyakanok, Roth, and Yih 2005).

Figure 16 shows summary results for the test set when using the top ten parses and the joint model. The weighting parameter for the parser probabilities was $\beta = 1$. We did not experiment extensively with different values of β. Preliminary experiments showed that considering 15 parses was a bit better, and considering the top 20 was a bit worse.

6.2 Evaluation on the CoNLL 2005 Shared Task

The CoNLL 2005 data is derived from Propbank version I, which is the first official release in 2005, whereas the results we have been reporting in the previous sections used the pre-final February 2004 data. Using the CoNLL 2005 evaluation standard ensures that results obtained by different groups are evaluated in exactly the same way. In

Task	CORE		COARSEARGM		ALL	
	F1	Acc.	F1	Acc.	F1	Acc.
ID	85.2	72.1	84.0	65.7	84.0	65.7
CLS	98.2	97.4	98.1	96.5	96.2	92.6
ID&CLS	83.6	70.8	82.5	63.9	80.8	61.2

Figure 16
Performance of the joint model using the top ten parses from Charniak's parser. Results are on Section 23.

Propbank I, there have been several changes in the annotation conventions, as well as error fixes and addition of new propositions. There was also a change in the way PP arguments are annotated: In the February 2004 data some PP arguments are annotated at the head NP child, but in Propbank I all PP arguments are annotated at the PP nodes. In order to achieve maximal performance with respect to these annotations, it would probably be best to change the feature definitions to account for the changes. However, we did no adaptation of the features.

The training set consists of the annotations in Sections 2 to 21, the development set is section 24 (Devset), and one of the test sets is section 23 (Test WSJ). The other test set is from the Brown corpus (Test Brown). The CoNLL annotations distinguish referring arguments, of the form R-ARGX, as discussed in Section 3.

Our approach to dealing with referring arguments and deciding when multiple identically labeled constituents are part of the same argument was to label constituents with only the set of argument labels and NONE and then map some of these labels into referring or continuation labels. We converted an ARGX into a R-ARGX if and only if the label of the constituent began with "WH". The rule for deciding when to add continuation labels was the same as for our systems for the February 2004 data described in Section 4.3: A constituent label becomes continuing if and only if it is a core argument label and there is another constituent with the same core argument label to the left. Therefore, for the CoNLL 2005 shared task we employ the same semantic role labeling system, just using a different post-processing rule to map to CoNLL-style labelings of sets of words.

We tested the upper bound in performance due to our conversion scheme in the following way: Take the gold-standard CoNLL annotations for the development set (including referring and continuing labels), convert these to basic argument labels of the form ARGX, then convert the resulting labeling to CoNLL-style labeling using our rules to recover the referring and continuing annotations. The F-Measure obtained was 99.0.

Figure 17 shows the performance of the local and joint model on one of the CoNLL test sets—Test WSJ (Section 23)—when using gold-standard parse trees. Performance on gold-standard parse trees was not measured in the CoNLL 2005 shared task, but we report it here to provide a basis for comparison with the results of other researchers.

Model	Test WSJ	
	F1	Acc.
Local	87.61	71.41
Joint	89.89	75.64

Figure 17
Results on the CoNLL WSJ Test set, when using gold-standard parse trees.

Model	Devset		Test WSJ		Test Brown	
	F1	Acc.	F1	Acc.	F1	Acc.
Local	75.13	51.60	77.03	53.24	65.37	34.83
Joint	77.20	55.79	78.68	56.43	67.67	38.68

Figure 18
Results on the CoNLL data set, when using Charniak automatic parse trees as provided in the CoNLL 2005 shared task data.

Model	Devset		Test WSJ		Test Brown	
	F1	Acc.	F1	Acc.	F1	Acc.
Local	75.80	53.05	78.00	55.31	65.55	35.70
Joint	77.93	57.20	79.71	58.65	67.79	39.43
Joint top 5	78.64	58.65	80.32	60.13	68.81	40.80

Figure 19
Results on the CoNLL data set, using automatic parse trees from the May 2005 version of the Charniak parser with correct treatment of forward quotes.

Next we present results using Charniak's automatic parses on the development and two test sets. We present results for the local and joint models using the max-scoring Charniak parse tree. Additionally, we report results for the joint model using the top five Charniak parse trees according to the algorithm described in Section 6.1. The performance measures reported here are higher than the results of our submission in the CoNLL 2005 shared task (Haghighi, Toutanova, and Manning 2005), because of two changes. One was changing the rule that produces continuing arguments to only add continuation labels to core argument labels; in the previous version the rule added continuation labels to all repeated labels. Another was fixing a bug in the way the sentences were passed in as input to Charniak's parser, leading to incorrect analyses of forward quotes.[8]

We first present results of our local and joint model using the parses provided as part of the CoNLL 2005 data (and having wrong forward quotes) in Figure 18. We then report results from the same local and joint model, and the joint model using the top five Charniak parses, where the parses have correct representation of the forward quotes in Figure 19. For these results we used the version of the Charniak parser from 4 May 2005. The results were very similar to the results we obtained with the version from 18 March 2005. We did not experiment with the new re-ranking model of Charniak and Johnson (2005), even though it improves upon Charniak (2000) significantly.

For comparison, the system we submitted to CoNLL 2005 had an F-Measure of 78.45 on the WSJ Test set. The winning system (Punyakanok, Roth, and Yih 2005) had an F-Measure of 79.44 and our current system has an F-Measure of 80.32. For the Brown Test set, our submitted version had an F-Measure of 67.71, the winning system had 67.75, and our current system has 68.81.

Figure 20 shows the per-label performance of our joint model using the top five Charniak parse trees on the Test WSJ test set. The columns show the Precision, Recall, F-Measure, and the total number of arguments for each label.

[8] The Charniak parses provided as part of the CoNLL shared task data uniformly ignore the distinction between forward and backward quotes and all quotes are backward. We re-ran the parser and obtained analyses with correct treatment of quotes.

7. Conclusions

In accord with standard linguistic assumptions, 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 non-local features. We incorporated joint information by using two types of features: features of the complete sequence of argument labels and features modeling dependencies between the labels of arguments and syntactic features of other arguments. We showed that both types of features yielded significant performance gains over a state-of-the-art local model.

For further improving performance in the presence of perfect syntactic parses, we see at least three promising avenues for improvement. First, one could improve the identification of argument nodes, by better handling of long-distance dependencies; for example, by incorporating models which recover the trace and null element information in Penn Treebank parse trees, as in Levy and Manning (2004). Second, it may be possible to improve the accuracy on modifier labels, by enhancing the knowledge about the semantic characteristics of specific words and phrases, such as by improving lexical statistics; for instance, our performance on ARGM-TMP roles is rather worse than that of some other groups. Finally, it is worth exploring alternative handling of multi-constituent arguments; our current model uses a simple rule in a post-processing step

	Precision	Recall	$F_{\beta=1}$	# Args
Overall	81.90%	78.81%	80.32	14,077
A0	88.37%	88.91%	88.64	3,563
A1	81.50%	81.27%	81.38	4,927
A2	73.44%	68.74%	71.01	1,110
A3	75.00%	55.49%	63.79	173
A4	74.74%	69.61%	72.08	102
A5	100.00%	80.00%	88.89	5
AM-ADV	64.86%	54.35%	59.14	506
AM-CAU	67.92%	49.32%	57.14	73
AM-DIR	55.74%	40.00%	46.58	85
AM-DIS	81.69%	75.31%	78.37	320
AM-EXT	65.00%	40.62%	50.00	32
AM-LOC	66.45%	56.75%	61.22	363
AM-MNR	62.50%	56.69%	59.45	344
AM-MOD	98.00%	98.00%	98.00	351
AM-NEG	97.83%	97.83%	97.83	230
AM-PNC	54.22%	39.13%	45.45	115
AM-PRD	100.00%	20.00%	33.33	5
AM-REC	0.00%	0.00%	0.00	2
AM-TMP	80.12%	72.31%	76.02	1,087
R-A0	93.15%	91.07%	92.10	224
R-A1	80.50%	82.05%	81.27	156
R-A2	61.54%	50.00%	55.17	16
R-A3	0.00%	0.00%	0.00	1
R-A4	0.00%	0.00%	0.00	1
R-AM-ADV	0.00%	0.00%	0.00	2
R-AM-CAU	100.00%	50.00%	66.67	4
R-AM-EXT	0.00%	0.00%	0.00	1
R-AM-LOC	76.92%	47.62%	58.82	21
R-AM-MNR	50.00%	33.33%	40.00	6
R-AM-TMP	74.00%	71.15%	72.55	52
V	97.32%	97.32%	97.32	5,267

Figure 20
Per-label performance of joint model using the top five Charniak automatic parse trees on the Test WSJ test set.

to decide which constituents given the same label are part of the same argument. This could be done more intelligently by the machine learning model.

Because perfect syntactic parsers do not yet exist and the major bottleneck to the performance of current semantic role labeling systems is syntactic parser performance, the more important question is how to improve performance in the presence of parser errors. We explored a simple approach of choosing from among the top *k* parses from Charniak's parser, which resulted in an improvement. Other methods have also been proposed, as we discussed in Section 2 (Màrquez et al. 2005; Pradhan, Ward et al. 2005; Punyakanok, Roth, and Yih 2005; Yi and Palmer 2005; Finkel, Manning, and Ng 2006). This is a very promising line of research.

Acknowledgments

This research was carried out while all the authors were at Stanford University. We thank the journal reviewers and the reviewers and audience at ACL 2005 and CoNLL 2005 for their helpful comments. We also thank Dan Jurafsky for his insightful comments and useful discussions. This work was supported in part by the Disruptive Technology Organization (DTO)'s Advanced Question Answering for Intelligence (AQUAINT) Program.

References

Baker, Collin, Charles Fillmore, and John Lowe. 1998. The Berkeley Framenet project. In *Proceedings of COLING-ACL*, pages 86–90, San Francisco, CA.

Carreras, Xavier and Luís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 89–97, Boston, MA.

Carreras, Xavier and Luís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 152–164, Ann Arbor, MI.

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

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, pages 173–180, Ann Arbor, MI.

Cohn, Trevor and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of CoNLL*, pages 169–172, Ann Arbor, MI.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, pages 175–182, Stanford, CA.

Finkel, Jenny, Christopher Manning, and Andrew Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of EMNLP*, pages 618–626, Sydney, Australia.

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

Hacioglu, Kadri. 2004. A lightweight semantic chunking model based on tagging. In *Proceedings of HLT-NAACL: Short Papers*, pages 145–148, Boston, MA.

Haghighi, Aria, Kristina Toutanova, and Christopher D. Manning. 2005. A joint model for semantic role labeling. In *Proceedings of CoNLL*, pages 173–176, Ann Arbor, MI.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, Williamstown, MA.

Levy, Roger and Chris Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of ACL*, pages 327–334, Barcelona, Spain.

Màrquez, Luís, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A robust combination strategy for semantic role labeling. In *Proceedings of EMNLP*, pages 644–651, Vancouver, Canada.

McCallum, Andrew, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML*, pages 591–598, Stanford, CA.

Palmer, Martha, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Pradhan, Sameer, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning Journal*, 60(1):11–39.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT-NAACL*, pages 233–240, Boston, MA.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of ACL*, pages 581–588, Ann Arbor, MI.

Punyakanok, Vasin and Dan Roth. 2001. The use of classifiers in sequential inference. In *Proceedings of NIPS*, pages 995–1001, Vancouver, Canada.

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI*, pages 1117–1123, Acapulco, Mexico.

Punyakanok, Vasin, Dan Roth, Wen-tau Yih, Dav Zimak, and Yuancheng Tu. 2004. Semantic role labeling via generalized inference over classifiers. In *Proceedings of CoNLL*, pages 130–133, Boston, MA.

Roland, Douglas and Daniel Jurafsky. 2002. Verb sense and verb subcategorization probabilities. In Paola Merlo and Suzanne Stevenson, editors, *The Lexical Basis of Sentence Processing: Formal, Computational, and Experimental Issues*. John Benjamins, Amsterdam, pages 325–345.

Sha, Fei and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 134–141, Edmonton, Canada.

Surdeanu, Mihai, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL*, pages 8–15, Sapporo, Japan.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*, pages 88–94, Barcelona, Spain.

Yi, Szu-ting and Martha Palmer. 2005. The integration of syntactic parsing and semantic role labeling. In *Proceedings of CoNLL*, pages 237–240, Ann Arbor, MI.