The Necessity of Syntactic Parsing for Semantic Role Labeling

Vasin Punyakanok **Dan Roth** **Wen-tau Yih**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{punyakan,danr,yih}@uiuc.edu

Abstract

We provide an experimental study of the role of syntactic parsing in semantic role labeling. Our conclusions demonstrate that syntactic parse information is clearly most relevant in the very first stage – the pruning stage. In addition, the quality of the pruning stage cannot be determined solely based on its recall and precision. Instead it depends on the characteristics of the output candidates that make downstream problems easier or harder. Motivated by this observation, we suggest an effective and simple approach of combining different semantic role labeling systems through joint inference, which significantly improves the performance.

1 Introduction

Semantic parsing of sentences is believed to be an important task toward natural language understanding, and has immediate applications in tasks such information extraction and question answering. We study *semantic role labeling (SRL)* in which for each verb in a sentence, the goal is to identify all constituents that fill a semantic role, and to determine their roles, such as Agent, Patient or Instrument, and their adjuncts, such as Locative, Temporal or Manner.

The PropBank project [Kingsbury and Palmer, 2002], which provides a large human-annotated corpus of semantic verb-argument relations, has enabled researchers to apply machine learning techniques to improve SRL systems [Gildea and Palmer, 2002; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Pradhan *et al.*, 2003; Surdeanu *et al.*, 2003; Pradhan *et al.*, 2004; Xue and Palmer, 2004]. However, most systems rely heavily on the full syntactic parse trees. Therefore, the overall performance of the system is largely determined by the quality of the automatic syntactic parsers of which state of the art [Collins, 1999; Charniak, 2001] is still far from perfect.

Alternatively *shallow* syntactic parsers (i.e., chunkers and clausers), although not providing as much information as a full syntactic parser, have been shown to be more robust in their specific task [Li and Roth, 2001]. This raises the very natural and interesting question of quantifying the necessity of the full parse information to semantic parsing and whether it is possible to use only shallow syntactic information to build an outstanding SRL system.

Although PropBank is built by adding semantic annotation to the constituents on syntactic parse trees in Penn Treebank, it is not clear how important syntactic parsing is for building an SRL system. To the best of our knowledge, this problem was first addressed by Gildea and Palmer [2002]. In their attempt of using limited syntactic information, the parser was *very shallow* – clauses were not available and only chunks were used. Moreover, the pruning stage in [Gildea and Palmer, 2002] was too strict since only chunks are considered as argument candidates, meaning that over 60% of the arguments were not treated as candidates. As a result, the overall recall in their approach was very low. As we will demonstrate later, high recall of the pruning stage is in fact essential to a quality SRL system.

Using only the shallow parse information in an SRL system has largely been ignored until the recent CoNLL-04 shared task competition [Carreras and Màrquez, 2004]. In this competition, participants were restricted to only shallow parse information for their SRL systems. As a result, it became clear that the performance of the best shallow parse based system [Hacioglu *et al.*, 2004] is only 10% in F_1 below the best system that uses full parse information [Pradhan *et al.*, 2004]. In addition, there has not been a true quantitative comparison with shallow parsing. First, the CoNLL-04 shared task used only a subset of the data for training. Furthermore, its evaluation treats the continued and referential tags differently, which makes the performance metric stricter and the results worse. Second, an SRL system is usually complicated and consists of several stages. It is still unknown how much and where precisely the syntactic information helps the most.

The goal of this study is twofold. First, we make a fair comparison between SRL systems which use full parse trees and those exclusively using shallow syntactic information. This brings forward a better analysis on the necessity of full parsing in the SRL task. Second, to relieve the dependency of the SRL system on the quality of automatic parsers, we improve semantic role labeling significantly by combining several SRL systems based on different state-of-art full parsers.

To make our conclusions applicable to general SRL systems, we adhere to a widely used two step system architecture. In the first step, the system is trained to identify argument candidates for a given verb predicate. In the second step, the system classifies the argument candidate into their types. In addition, it is also common to use a simple procedure to prune obvious non-candidates before the first step, and to use post-processing inference to fix inconsistent predictions after the second step. We also employ these two additional steps.

In our comparison between the systems using shallow and full syntactic information, we found the most interesting result is that while each step of the system using shallow information exhibits very good performance, the overall performance is significantly inferior to the system that uses full information. This necessity of full parse information is especially noticeable at the pruning stage. In addition, we produce a state-of-the-art SRL system by combining of different SRL systems based on two (potentially noisy) automatic full parsers [Collins, 1999; Charniak, 2001].

The rest of the paper is organized as follows. Section 2 gives a brief description of the semantic role labeling task and the PropBank corpus. Section 3 introduces the general architecture of an SRL system, including the features used in different stages. The detailed experimental comparison between using full parsing and shallow parsing is provided in Section 4, where we try to explain why and where the full parse information contributes to SRL. Inspired by the result, we suggests an approach that combines different SRL systems based on joint inference in Section 5. Finally, Section 6 concludes this paper.

2 Semantic Role Labeling (SRL) Task

The goal of the semantic-role labeling task is to discover the verb-argument structure for a given input sentence. For example, given a sentence " I *left* my pearls to my daughter-in-law in my will", the goal is to identify different arguments of the verb *left* which yields the output:

$[_{\text{A0}}$ I$]$ $[_{\text{V}}$ *left* $]$ $[_{\text{A1}}$ my pearls$]$ $[_{\text{A2}}$ to my daughter-in-law$]$ $[_{\text{AM-LOC}}$ in my will$]$.

Here A0 represents the *leaver*, A1 represents the *thing left*, A2 represents the *benefactor*, AM-LOC is an adjunct indicating the location of the action, and V determines the verb. In addition, each argument can be mapped to a constituent in its corresponding syntactic full parse tree.

Following the definition of the PropBank and CoNLL-2004 shared task, there are six different types of arguments labeled as A0-A5 and AA. These labels have different semantics for each verb as specified in the PropBank Frame files. In addition, there are also 13 types of adjuncts labeled as AM-*adj* where *adj* specifies the adjunct type. In some cases, an argument may span over different parts of a sentence, the label C-*arg* is used to specify the continuity of the arguments, as shown in the example below.

$[_{\text{A1}}$ The pearls$]$, $[_{\text{A0}}$ I$]$ $[_{\text{V}}$ *said*$]$, $[_{\text{C-A1}}$ were left to my daughter-in-law$]$.

Moreover in some cases, an argument might be a relative pronoun that in fact refers to the actual agent outside the clause. In this case, the actual agent is labeled as the appropriate argument type, *arg*, while the relative pronoun is instead labeled as R-*arg*. For example,

$[_{\text{A1}}$ The pearls$]$ $[_{\text{R-A1}}$ which$]$ $[_{\text{A0}}$ I$]$ $[_{\text{V}}$ *left*$]$, $[_{\text{A2}}$ to my daughter-in-law$]$ are fake.

The distribution of these argument labels is fairly unbalanced. In the official release of PropBank I, core arguments (A0–A5 and AA) occupy 71.26%, where the largest parts are A0 (25.39%) and A1 (35.19%). The rest portion is mostly the adjunct arguments (24.90%). The continued (C-*arg*) and referential (R-*arg*) arguments are relatively fewer, occupying 1.22% and 2.63% respectively. For more definitions of PropBank and the semantic role labeling task, readers can refer to [Kingsbury and Palmer, 2002] and [Carreras and Màrquez, 2004].

3 SRL System Architecture

Our SRL system consists of four stages: *pruning*, *argument identification*, *argument classification*, and *inference*. In particular, the goal of pruning and argument identification is to identify argument candidates for a given verb predicate. The system only classifies the argument candidate into their types in the stage of argument classification. Linguistic and structural constraints are incorporated in the inference stage to resolve inconsistent global predictions. This section describes how we build these four stages, including the features used in training the classifiers.

3.1 Pruning

When the full parse tree of a sentence is available, only the constituents in the parse tree are considered as argument candidates. In addition, our system exploits the heuristic rules introduced by Xue and Palmer [2004] to filter out simple constituents that are very unlikely to be arguments. The heuristic is a recursive process starting from the verb of which arguments to be identified. It first returns the siblings of the verb as candidates; then it moves to the parent of the verb, and collects the siblings again. The process goes on until it reaches the root. In addition, if a constituent is a PP (propositional phrase), its children are also collected.

3.2 Argument Identification

The argument identification stage utilizes binary classification to identify whether a candidate is an argument or not. When full parsing is available, we train and apply the binary classifiers on the constituents supplied by the pruning stage. When only shallow parsing is available, the system does not have the pruning stage, and also does not have constituents to begin with. Therefore, conceptually the system has to consider all possible subsequences (i.e., consecutive words) in a sentence as potential argument candidates. We avoid this by using a learning scheme by first training two classifiers, one to predict the beginnings of possible arguments, and the other the ends. The predictions are combined to form argument candidates that do not violate the following constraints.

1. Arguments cannot cover the predicate.
2. Arguments cannot overlap with the clauses (they can be embedded in one another).
3. If a predicate is outside a clause, its arguments cannot be embedded in that clause.

The features used in the full parsing and shallow parsing settings are described as follows.

Features when full parsing is available
Most of the features used in our system are standard features which include

- **Predicate and POS tag of predicate** features indicate the lemma of the predicate verb and its POS tag.
- **Voice** feature indicates passive/active voice of the predicate.
- **Phrase type** feature provides the phrase type of the constituent.
- **Head word and POS tag of the head word** feature provides the head word and its POS tag of the constituent. We use rules introduced by Collins [1999] to extract this feature.
- **Position** feature describes if the constituent is before or after the predicate relative to the position in the sentence.
- **Path** records the traversal path in the parse tree from the predicate to the constituent.
- **Subcategorization** feature describes the phrase structure around the predicate's parent. It records the immediate structure in the parse tree that expands to its parent.

We also use the following additional features.

- **Verb class** feature is the class of the active predicate described in PropBank Frames.
- **Lengths** of the target constituent, in the numbers of words and chunks separately.
- **Chunk** tells if the target argument is, embeds, overlaps, or is embedded in a chunk with its type.
- **Chunk pattern** encodes the sequence of chunks from the current words to the predicate.
- **Chunk pattern length** feature counts the number of chunks in the argument.
- **Clause relative position** feature is the position of the target word relative to the predicate in the pseudo-parse tree constructed only from clause constituent. There are four configurations—target constituent and predicate share the same parent, target constituent parent is an ancestor of predicate, predicate parent is an ancestor of target word, or otherwise.
- **Clause coverage** describes how much of the local clause (from the predicate) is covered by the target argument.
- **NEG** feature is active if the target verb chunk has `not` or `n't`.
- **MOD** feature is active when there is a modal verb in the verb chunk. The rules of the **NEG** and **MOD** features are used in a baseline SRL system developed by Erik Tjong Kim Sang [Carreras and Màrquez, 2004].

Features when only shallow parsing is available
Features used are similar to those used by the system with full parsing except those that need full parse trees to generate. For these types of features, we either try to mimic the features with some heuristics rules or discard them. The details of these features are as follows.

- **Phrase type** uses a simple heuristics to identify only VP, PP, and NP.
- **Head word and POS tag of the head word** are the rightmost word for NP, and leftmost word for VP and PP.
- **Shallow-Path** records the traversal path in the pseudo-parse tree constructed only from the clause structure and chunks.
- **Shallow-Subcategorization** feature describes the chunk and clause structure around the predicate's parent in the pseudo-parse tree.
- **Syntactic frame** features are discarded.

3.3 Argument Classification

This stage assigns the final argument labels to the argument candidates supplied from the previous stage. A multi-class classifier is trained to classify the types of the arguments supplied by the argument identification stage. In addition, to reduce the excessive candidates mistakenly output by the previous stage, the classifier can also classify the argument as *NULL* (meaning "not an argument") to discard the argument.

The features used here are the same as those used in the argument identification stage. However, when full parsing are available, an additional feature introduced by Xue and Palmer [2004] is used.

- **Syntactic frame** describes the sequential pattern of the noun phrases and the predicate in the sentence.

3.4 Inference

The purpose of this stage is to incorporate some prior linguistic and structural knowledge, such as "arguments do not overlap" or "each verb takes at most one argument of each type." This knowledge is used to resolve any inconsistencies of argument classification in order to generate final legitimate predictions. We use the inference process introduced by Punyakanok *et al.* [2004]. The process is formulated as an integer linear programming (ILP) problem that takes as inputs the confidences over each type of the arguments supplied by the argument classifier. The output is the optimal solution that maximizes the linear sum of the confidence scores (e.g., the conditional probabilities estimated by the argument classifier), subject to the constraints that encode the domain knowledge.

4 The Necessity of Syntactic Parsing

We study the necessity of syntactic parsing experimentally by observing the effects of using full parsing and shallow parsing at each stage of an SRL system. In Section 4.1, we first describe how we prepare the data, as well as the basic system including features and the learning algorithm. The comparison of full parsing and shallow parsing on the three stages (excluding the inference stage) is presented in the reversed order (Sections 4.2, 4.3, 4.4).

4.1 Experimental Setting

We use PropBank sections 02 through 21 as training data, and section 23 as testing. In order to apply the standard CoNLL-

04 evaluation script, our system conforms to both the input and output format defined in the shared task.

The CoNLL-04 evaluation metric is slightly more restricted since an argument prediction is only considered correct when all its *continued* arguments (C-*arg*) are correct and *referential* arguments (R-*arg*) are included – these requirements are often absent in previous SRL systems, given that they only occupy a very small percentage of the data. To provide a fair comparison, we also report the performance when discarding continued and referential arguments. Following the notation used in [Xue and Palmer, 2004], this evaluation metric is referred as "argM+", which considers all the core arguments and adjunct arguments. We note here that all the performance reported excludes V label which usually improves the overall performance if included.

The goal of the experiments in this section is to understand the effective contribution of full parsing versus shallow parsing using only the part-of-speech tags, chunks, and clauses. In addition, we also compare performance when using the correct (gold standard) versus using automatic parse data. The automatic full parse trees are derived using Charniak's parser [Charniak, 2001] (version 0.4). In automatic shallow parsing, the information is generated by a state-of-the-art POS tagger [Even-Zohar and Roth, 2001], chunker [Punyakanok and Roth, 2001], and clauser [Carreras and Màrquez, 2003].

The learning algorithm used is a variation of the Winnow update rule incorporated in SNoW [Roth, 1998; Roth and Yih, 2002], a multi-class classifier that is tailored for large scale learning tasks. SNoW learns a sparse network of linear functions, in which the targets (argument border predictions or argument type predictions, in this case) are represented as linear functions over a common feature space. It improves the basic Winnow multiplicative update rule in several ways. For example, a regularization term is added, which has the effect of trying to separate the data with a large margin separator [Grove and Roth, 2001; Hang *et al.*, 2002] and voted (averaged) weight vector is used [Freund and Schapire, 1999].

Experimental evidences have shown that SNoW activations correlate with the confidence of the prediction and can provide an estimate of probability to be used for both argument identification and inference. We use the softmax function [Bishop, 1995] to convert raw activation to conditional probabilities. Specifically, if there are n classes and the raw activation of class i is act_i, the posterior estimation for class i is

$$\mathrm{score}(i) = p_i = \frac{e^{act_i}}{\sum_{1 \le j \le n} e^{act_j}}.$$

4.2 Argument Classification

To evaluate the performance gap between full parsing and shallow parsing in argument classification, we assume the argument boundaries are known, and only train classifiers to classify the labels of these arguments. In this stage, the only difference between *full parsing* and *shallow parsing* is the construction of three full parsing features: *path*, *subcategorization* and *syntactic frame*. As described in Section 3, *path* and *subcategorization* can be approximated by *shallow-path* and *shallow-subcategorization* using chunks and clauses. However, it is unclear how to mimic the syntactic frame feature since it relies on the internal structure of a full parse tree. Therefore, it does not have a corresponding feature in the shallow parsing case.

Table 1 reports the experimental results of argument classification when argument boundaries are known. Although full parsing features seem to help when using the gold standard data, the difference in F_1 is only 0.32% and 0.13% for the CoNLL-2004 and ArgM+ evaluation respectively. When the automatic (full and shallow) parsers are used, the gap is smaller.

	Full Parsing	Shallow Parsing
Gold	91.32	91.00
Auto	90.93	90.69
Gold (ArgM+)	90.67	91.54
Auto (ArgM+)	90.87	90.93

Table 1: The accuracy of argument classification when argument boundaries are known

Lesson When the argument boundaries are known, the performance of the full paring systems is about the same as the shallow parsing system.

4.3 Argument Identification

Argument identification is an important stage that effectively reduces the number of argument candidates after pruning. Given an argument candidate, an argument identifier is a binary classifier that decides whether or not the candidate should be considered as an argument. To evaluate the influence of full parsing in this stage, the candidate list used here is the pruning results on the gold standard parse trees.

Similar to the argument classification stage, the only difference between full-parse and shallow-parse is the use of *path* and *subcategorization* features. Again, we replace them with *shallow-path* and *shallow-subcategorization* when the binary classifier is trained using the shallow parsing information.

Table 2 reports the performance of the argument identifier on the test set using the direct predictions of the trained binary classifier. The recall and precision of the full parsing system are around 2 to 3 percents higher than the shallow parsing system on the gold standard data. As a result, the F_1 score is 2.5% higher. The performance on automatic parse data is unsurprisingly lower, but the difference between full parsing and shallow parsing is relatively the same. In terms of filtering efficiency, around 25% of the examples are predicted as positive. In other words, both argument identifiers filter out around 75% of the argument candidates after pruning.

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	96.53	93.57	95.03	93.66	91.72	92.68
Auto	94.68	90.60	92.59	92.31	88.36	90.29

Table 2: The performance of argument identification after pruning (based on the gold standard full parse trees)

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	92.13	95.62	93.84	88.54	94.81	91.57
Auto	89.48	94.14	91.75	86.14	93.21	89.54

Table 3: The performance of argument identification after pruning (based on the gold standard full parse trees) and with threshold=0.1

Since the recall in argument identification sets the upper bound of the recall in argument classification, in practice, the threshold that predicts examples as positive is usually lowered to allow more positive predictions. That is, a candidate is predicted as positive when its probability estimation is larger than the threshold. Table 3 shows the performance of the argument identifiers when the threshold is 0.1.

Since argument identification is just an intermediate step of a complete system, a more realistic evaluation method is to see how each final system performs. Table 4 and Table 5 report the final results in recall, precision, and F_1 in CoNLL and ArgM+ metrics. The F_1 difference is about 4.5% when using the gold standard data. However, when automatic parsers are used, shallow-parse is in fact slightly better. This may be due to the fact that shallow parsers are more accurate in chunk or clause predictions compared to a regular full parser [Li and Roth, 2001].

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	88.81	89.35	89.08	84.19	85.03	84.61
Auto	84.21	85.04	84.63	86.17	84.02	85.08

Table 4: The CoNLL-04 evaluation of the overall system performance when pruning (using the gold standard full parse trees) is available

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	89.02	89.57	89.29	84.46	85.31	84.88
Auto	84.38	85.38	84.87	86.37	84.36	85.35

Table 5: ArgM+ performance of the overall system when pruning (using the gold standard full parse trees) is available

Lesson Full parsing helps in argument identification. However, when the automatic shallow parser is more accurate than the full parser, using the full parsing information may not have advantages over shallow parsing.

4.4 Pruning

As shown in the previous two subsections, the performance difference of full parsing and shallow parsing is not large when the pruning information is given. We conclude that the main contribution of the full parse is in the pruning stage. Since the shallow parsing system does not have enough information for the pruning heuristics, we train two word based classifiers to replace the pruning stage. One classifier is trained to predict whether a given word is the start (S) of an argument; the other classifier is to predict the end (E) of an argument. If the product of probabilities of a pair of S and E predictions is larger than a predefined threshold, then this pair is considered as an argument candidate. The pruning comparison of using the classifiers and heuristics is shown in Table 6.

	Full Parsing			Classifier th=0.04		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	25.94	97.27	40.96	29.58	97.18	45.35
Auto	22.79	86.08	36.04	24.68	94.80	39.17

Table 6: The performance of pruning

Amazingly, the classifier pruning strategy seems better than the heuristics. With about the same recall, the classifiers achieve higher precision. However, to really compare systems using full parsing and shallow parsing, we still need to see the overall performance. We build two semantic role systems based on full parsing and shallow parsing. The full parsing system follows the pruning, argument identification, argument classification, and inference stages, as described earlier. For the shallow parsing system, pruning is replaced by the word-based pruning classifiers, and the rest stages are designed only to use shallow parsing information as described in previous sections. Table 7 and Table 8 show the overall performance in the two evaluation methods.

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	88.81	89.35	89.08	75.34	75.28	75.31
Auto	77.09	75.51	76.29	75.48	67.13	71.06

Table 7: The CoNLL-04 evaluation of the overall system performance

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	89.02	89.57	89.29	75.35	75.20	75.27
Auto	77.09	75.57	76.32	75.54	67.14	71.09

Table 8: ArgM+ performance of the overall system

As indicated in the tables, the gap in F_1 between the full parsing and shallow parsing systems enlarges to more than 13% on the gold standard data. At first glance, this result seems to contradict our conclusion in Section 4.3. After all, if the pruning stage of the shallow parsing SRL system performs equally well or even better, the overall performance gap in F_1 should be small.

After we carefully examine the output of the word-based classifier pruning, we realize that it in fact filters out "easy" candidates, and leaves examples that are difficult to the later stages. To be specific, these argument candidates often overlap and differ only with one or two words. On the other hand, the pruning heuristics based on full parsing never outputs overlapping candidates. The following argument identification stage can be thought of as good in discriminating different types of candidates.

Lesson The most crucial contribution of full parsing is in pruning. The internal tree structure helps significantly in discriminating argument candidates, which makes the work easy to the following stages.

5 Combine Different SRL Systems

The empirical study in Section 4 indicates the performance of an SRL system primarily depends on the very first stage – pruning, which is derived directly from the full parse trees. This also means that in practice the quality of the syntactic parser is decisive to the quality of the SRL system. To improve semantic role labeling, one possible way is to combine different SRL systems through a joint inference stage, given that the systems are derived using different full parse trees.

To test this idea, we first build two SRL systems that use Collins' parser[1] and Charniak's parser respectively. In fact, these two parsers have noticeably different output. Applying punning heuristics on the output of Collins' parser produces a list of candidates with 81.05% recall. Although this number is significantly lower that 86.08% recall produced by Charniak's parser, the union of the two candidate lists still significantly improves recall to 91.37%. We construct the two systems by implementing the first three stages, namely pruning, argument identification, and argument classification. When a testing sentence is given, a joint inference stage is used to resolve the inconsistency of the output of argument classification in these two systems.

We first briefly describe the inference procedure introduced by Punyakanok *et al.* [2004]. Formally speaking, the argument classifier attempts to assign labels to a set of arguments, $S^{1:M}$, indexed from 1 to M. Each argument S^i can take any label from a set of argument labels, $\mathcal{P}$, and the indexed set of arguments can take a set of labels, $c^{1:M} \in \mathcal{P}^M$. If we assume that the argument classifier returns an estimated conditional probability distribution, $Prob(S^i = c^i)$, then, given a sentence, the inference procedure seeks an global assignment that maximizes the following objective function,

$$\hat{c}^{1:M} = \underset{c^{1:M} \in \mathcal{P}^M}{\operatorname{argmax}} \sum_{i=1}^{M} Prob(S^i = c^i), \tag{1}$$

subject to linguistic and structural constraints. In other words, this objective function reflects the expected number of correct argument predictions, subject to the constraints.

When there are two or more argument classifiers from different SRL systems, a joint inference procedure can take the output estimated probabilities for these candidates as input, although some candidates may refer to the same phrases in the sentence. For example, Figure 1 shows the two candidate sets for a fragment of a sentence, "*..., traders say, unable to* ***cool*** *the selling panic in both stocks and futures.*" In this example, system A has two argument candidates, a_1 = "traders" and a_4 = "the selling panic in both stocks and futures"; system B has three argument candidates, b_1 = "traders", b_2 = "the selling panic", and b_3 = "in both stocks and futures".

[1]We use the Collins' parser implemented by Bikel [2004].

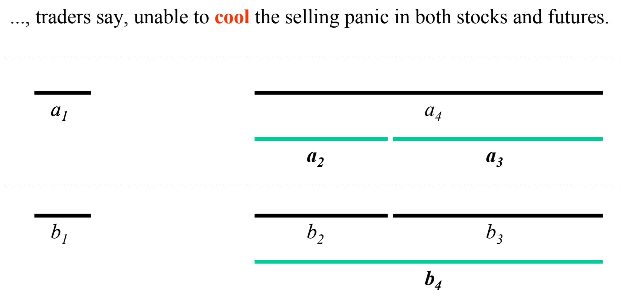
..., traders say, unable to cool the selling panic in both stocks and futures.
a_1
a_4
a_2
a_3
b_1
b_2
b_3
b_4

Figure 1: The output of two SRL systems: system A has two candidates, a_1 = "traders" and a_4 = "the selling panic in both stocks and futures"; system B has three argument candidates, b_1 = "traders", b_2 = "the selling panic", and b_3 = "in both stocks and futures". In addition, we create two phantom candidates a_2 and a_3 for system A that correspond to b_2 and b_3 respectively, and b_4 for system B that corresponds to a_4.

If we throw all these variables together into the inference procedure, then the final prediction will be more likely dominated by the system that has more candidates, which is system B in this example. The reason is because our objective function is the sum of the probabilities of all the candidate assignments.

This bias can be corrected by the following observation. Although system A only has two candidates, a_1 and a_4, it can be treated as it also has two additional *phantom* candidates, a_2 and a_3, where a_2 and b_2 refer to the same phrase, and so do a_3 and b_3. Similarly, system B has a phantom candidate b_4 that corresponds to a_4. Because system A does not really generate a_2 and a_3, we can assume that these two phantom candidates are predicted as *NULL* (i.e., not an argument). We assign the same prior distribution to each phantom candidate. In particular, the probability of the *null* class is set to be 0.55 based on empirical tests, and the probabilities of the rest classes are set based on their occurrence frequencies in the training data.

Tables 9 and 10 report the performance of individual systems, as well as the joint system. The joint system based on this straightforward strategy significantly improves the performance compared to the two original SRL systems in both recall and precision, and thus achieves a much higher F_1.

6 Conclusions

In this paper, we make a fair comparison between the SRL systems using full parse trees and using only shallow syntactic information. What we found confirms the necessity of full parsing for the SRL problem. In particular, this information is the most crucial in the pruning stage of the system, and relatively less important to the following stages. Inspired by this observation, we proposed an effective and simple approach that combines different SRL systems through a joint inference stage. The combined system significantly improves the performance compared to the original systems.

	Prec	Rec	F_1
Collins' Parse	75.92	71.45	73.62
Charniak's Parse	77.09	75.51	76.29
Combined Result	80.53	76.94	78.69

Table 9: The performance in CoNLL-04's evaluation of individual and combined SRL systems

	Prec	Rec	F_1
Collins' Parse	75.87	71.36	73.54
Charniak's Parse	77.09	75.57	76.32
Combined Result	80.56	76.99	78.73

Table 10: The performance in argM+'s evaluation of individual and combined SRL systems

Acknowledgments

We thank Dav Zimak, Kevin Small, and the anonymous referees for their useful comments. We are also grateful to Dash Optimization for the free academic use of Xpress-MP. This research is supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program, a DOI grant under the Reflex program, and an ONR MURI Award.

References

[Bikel, 2004] Dan Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4), December 2004.

[Bishop, 1995] C. Bishop. *Neural Networks for Pattern Recognition*, chapter 6.4: Modelling conditional distributions, page 215. Oxford University Press, 1995.

[Carreras and Màrquez, 2003] X. Carreras and L. Màrquez. Phrase recognition by filtering and ranking with perceptrons. In *Proc. of RANLP-2003*, 2003.

[Carreras and Màrquez, 2004] X. Carreras and L. Màrquez. Introduction to the conll-2004 shared tasks: Semantic role labeling. In *Proc. of CoNLL-2004*, 2004.

[Charniak, 2001] E. Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics*, Toulouse, France, 2001.

[Chen and Rambow, 2003] J. Chen and O. Rambow. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proc. of EMNLP-2003*, Sapporo, Japan, 2003.

[Collins, 1999] M. Collins. *Head-driven Statistical Models for Natural Language Parsing.* PhD thesis, Computer Science Department, University of Pennsylvenia, Philadelphia, 1999.

[Even-Zohar and Roth, 2001] Y. Even-Zohar and D. Roth. A sequential model for multi class classification. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 10–19, 2001.

[Freund and Schapire, 1999] Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[Gildea and Hockenmaier, 2003] D. Gildea and J. Hockenmaier. Identifying semantic roles using combinatory categorial grammar. In *Proc. of the EMNLP-2003*, Sapporo, Japan, 2003.

[Gildea and Palmer, 2002] D. Gildea and M. Palmer. The necessity of parsing for predicate argument recognition. In *Proc. of ACL 2002*, Philadelphia, PA, 2002.

[Grove and Roth, 2001] A. Grove and D. Roth. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141, 2001.

[Hacioglu *et al.*, 2004] K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. Semantic role labeling by tagging syntactic chunks. In *Proc. of CoNLL-04*, 2004.

[Hang *et al.*, 2002] T. Hang, F. Damerau, and D. Johnson. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637, 2002.

[Kingsbury and Palmer, 2002] P. Kingsbury and M. Palmer. From Treebank to PropBank. In *Proc. of LREC-2002*, Spain, 2002.

[Li and Roth, 2001] X. Li and D. Roth. Exploring evidence for shallow parsing. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 107–110, 2001.

[Pradhan *et al.*, 2003] S. Pradhan, K. Hacioglu, W. Ward, J. Martin, and D. Jurafsky. Semantic role parsing adding semantic structure to unstructured text. In *Proc. of ICDM-2003*, 2003.

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

[Punyakanok and Roth, 2001] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS-13*, pages 995–1001, 2001.

[Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proc. of COLING-2004*, 2004.

[Roth and Yih, 2002] D. Roth and W. Yih. Probabilistic reasoning for entity & relation recognition. In *Proc. of COLING-2002*, pages 835–841, 2002.

[Roth, 1998] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI*, pages 806–813, 1998.

[Surdeanu *et al.*, 2003] M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proc. of ACL 2003*, 2003.

[Xue and Palmer, 2004] N. Xue and M. Palmer. Calibrating features for semantic role labeling. In *Proc. of the EMNLP-2004*, pages 88–94, Barcelona, Spain, 2004.