A Robust Combination Strategy for Semantic Role Labeling

Lluís Màrquez, Mihai Surdeanu, Pere Comas, and Jordi Turmo
Technical University of Catalunya
Barcelona, Spain
{lluism,surdeanu,pcomas,turmo}@lsi.upc.edu

Abstract

This paper focuses on semantic role labeling using automatically-generated syntactic information. A simple and robust strategy for system combination is presented, which allows to partially recover from input parsing errors and to significantly boost results of individual systems. This combination scheme is also very flexible since the individual systems are not required to provide any information other than their solution. Extensive experimental evaluation in the CoNLL-2005 shared task framework supports our previous claims. The proposed architecture outperforms the best results reported in that evaluation exercise.

1 Introduction

The task of Semantic Role Labeling (SRL), i.e. the process of detecting basic event structures such as *who* did *what* to *whom*, *when* and *where*, has received considerable interest in the past few years (Gildea and Jurafsky, 2002; Surdeanu et al., 2003; Xue and Palmer, 2004; Pradhan et al., 2005a; Carreras and Màrquez, 2005). It was shown that the identification of such event frames has a significant contribution for many Natural Language Processing (NLP) applications such as Information Extraction (Surdeanu et al., 2003) and Question Answering (Narayanan and Harabagiu, 2004).

Most current SRL approaches can be classified in one of two classes: approaches that take advantage of complete syntactic analysis of text, pioneered by Gildea and Jurafsky (2002), and approaches that use partial syntactic analysis, championed by previous evaluations performed within the Conference on Computational Natural Language Learning (CoNLL) (Carreras and Màrquez, 2004). The wisdom extracted from this volume of work indicates that full syntactic analysis has a significant contribution to the SRL performance, *when using hand-corrected syntactic information*.

On the other hand, when only automatically-generated syntax is available, the quality of the information provided through full syntax decreases because the state-of-the-art of full parsing is less robust and performs worse than the tools used for partial syntactic analysis. Under such real-world conditions, the difference between the two SRL approaches (with full or partial syntax) is not that high. More interestingly, *the two SRL strategies perform better for different semantic roles*. For example, models that use full syntax recognize better agent and theme roles, whereas models based on partial syntax are better at recognizing explicit patient roles, which tend to be farther from the predicate and accumulate more parsing errors (Màrquez et al., 2005).

The above observations motivate the work presented in this paper. We introduce a novel semantic role labeling approach that combines several individual SRL systems. Intuitively, our approach can be separated in two stages: a *candidate generation* phase, where the solutions provided by several individual models are combined into a pool of candidate arguments, and an *inference* phase, where the candidates are filtered using a binary classifier, and possi-

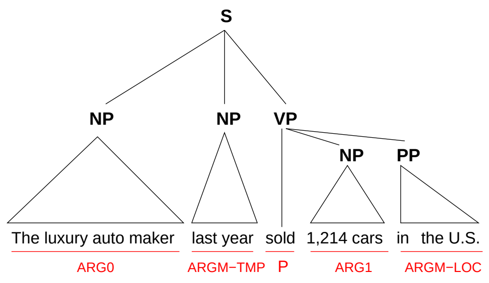

Figure 1: Sample PropBank sentence.

ble conflicts with domain knowledge constraints are resolved to obtain the final solution.

For robustness, the inference model uses only global attributes extracted from the solutions provided by the individual systems, e.g., the sequence of role labels generated by each system for the current predicate. We do not use any attributes specific to the individual models, not even the confidence assigned by the individual classifiers. Besides simplicity, the consequence of this decision is that our approach does not impose any restrictions on the individual SRL strategies, as long as one solution is provided for each predicate. On the other hand, probabilistic inference processes, which have been successfully used for SRL (Koomen et al., 2005), mandate that each individual candidate argument be associated with its raw activation, or confidence, in the given model. However, this information is not directly available in two out of three of our individual models, which classify argument chunks and not entire arguments.

Despite its simplicity, our approach obtains encouraging results: the combined system outperforms any of the individual systems and, using exactly the same data, it is also competitive with the best SRL systems that participated in the latest CoNLL shared task evaluation (Carreras and Màrquez, 2005).

2 Semantic Corpora

In this paper we report results using PropBank, an approximately one-million-word corpus annotated with predicate-argument structures (Kingsbury et al., 2002). To date, PropBank addresses mainly predicates lexicalized by verbs and a small number of predicates lexicalized by verb nominalizations and adjectives.

The arguments of each predicate are numbered sequentially from `ARG0` to `ARG5`. Generally, `ARG0` stands for *agent*, `ARG1` for *theme* or *direct object*, and `ARG2` for *indirect object*, *benefactive* or *instrument*, but mnemonics tend to be verb specific. Additionally, predicates might have “adjunctive arguments”, referred to as `ARGM`s. For example, `ARGM-LOC` indicates a locative and `ARGM-TMP` indicates a temporal. Figure 1 shows a sample sentence where one predicate (“sold”) has 4 arguments.

In a departure from “traditional” SRL approaches that train on the hand-corrected syntactic trees associated with PropBank, we do not use any syntactic information from PropBank. Instead, we develop our models using automatically-generated syntax and named-entity (NE) labels, made available by the CoNLL shared task evaluation (Carreras and Màrquez, 2005). From the CoNLL data, our individual models based on full syntactic analysis use the trees generated by the Charniak parser. The partial-syntax model uses the chunk – i.e. basic syntactic phrase – labels and clause boundaries. All individual models make use of the provided NE labels.

Following the CoNLL-2005 setting we evaluated our system also on a fresh test set, derived from the Brown corpus. This second evaluation allows us to re-enforce our robustness claim.

3 Approach Overview

The proposed architecture, summarized in Figure 2, consists of two stages: a *candidate generation* phase and an *inference* stage.

In the candidate generation step, we merge the solutions of three individual SRL models into a unique pool of candidate arguments. The proposed models range from complete reliance on full parsing to using only partial syntactic information. The first two models, Model 1 and 2, are developed as sequential taggers (using the BIO tagging scheme) on a shared framework. The major difference between the two models is that Model 1 uses only partial syntactic information (basic phrases and clause boundaries), whereas Model 2 uses complete syntactic information. To maximize diversity, Model 3 implements a different strategy: it models only arguments that map into exactly one syntactic constituent. Section 4 details all three individual models.

The inference stage starts with *candidate filtering*,

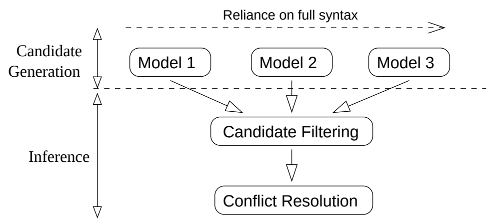

Figure 2: Architecture of the proposed system.

which reduces the number of candidate arguments in the pool using a single binary classifier. Using this classifier's confidence values and a number of domain-specific constraints, e.g. no two arguments can overlap, the *conflict resolution* component enforces the consistency of the final solution using a straightforward greedy strategy. The complete inference model is detailed in Section 5.

4 Individual SRL Models

Models 1 and 2. These models approach SRL as a sequential tagging task. In a pre-process step, the input syntactic structures are traversed in order to select a subset of constituents organized sequentially (i.e. non embedding). Model 1 makes use only of the partial tree defined by base chunks and clause boundaries, while Model 2 explores full parse trees.

Precisely, the sequential tokens are selected as follows. First, the input sentence is splitted into disjoint segments by considering the clause boundaries given by the syntactic structure. Second, for each segment, the set of top-most non-overlapping syntactic constituents completely falling inside the segment are selected as tokens. Note that this strategy provides a set of sequential tokens covering the complete sentence. Also, it is independent of the syntactic annotation explored, given it provides clause boundaries — see (Màrquez et al., 2005) for more details.

Due to this pre-processing stage, the upper-bound recall figures are 95.67% for Model 1 and 90.32% for Model 2 using the datasets defined in Section 6.

The nodes selected are labeled with B-I-O tags (depending if they are at the beginning, inside, or outside of a predicate argument) and they are converted into training examples by considering a rich set of features, mainly borrowed from state-of-the-art systems. These features codify properties from: (a) the argument constituent, (b) the target predicate,

Constituent *type* and *head*: extracted using common head-word rules. If the first element is a PP chunk, then the head of the first NP is extracted.
First and last words and POS tags of the constituent.
POS sequence: if it is less than 5 tags long. *2/3/4-grams* of the POS sequence.
Bag-of-words of nouns, adjectives, and adverbs.
TOP sequence: sequence of types of the top-most syntactic elements in the constituent (if it is less than 5 elements long). In the case of full parsing this corresponds to the right-hand side of the rule expanding the constituent node. *2/3/4-grams* of the TOP sequence.
Governing category as in (Gildea and Jurafsky, 2002).
NamedEnt, indicating if the constituent embeds or strictly matches a named entity along with its type.
TMP, indicating if the constituent embeds or strictly matches a temporal keyword (extracted from `AM-TMP` arguments of the training set).
Previous and following words and POS of the constituent.
The same features characterizing focus constituents are extracted for the *two previous and following tokens*, provided they are inside the clause boundaries of the codified region.

Table 1: Constituent structure features: Models 1/2

Predicate *form*, *lemma*, and *POS tag*.
Chunk type and *type of verb phrase* in which verb is included: single-word or multi-word.
The predicate *voice*. We currently distinguish five voice types: active, passive, copulative, infinitive, and progressive.
Binary flag indicating if the verb is a *start/end* of a clause.
Sub-categorization rule, i.e. the phrase structure rule that expands the predicate immediate parent.

Table 2: Predicate structure features: Models 1/2

and (c) the distance between the argument and predicate. The three feature sets are listed in Tables 1, 2, and 3, respectively.[1]

Regarding the learning algorithm, we used generalized AdaBoost with real-valued weak classifiers, which constructs an ensemble of decision trees of fixed depth (Schapire and Singer, 1999). We considered a one-vs-all decomposition into binary problems to address multi-class classification. AdaBoost binary classifiers are then used for *labeling* test sequences, from left to right, using a recurrent sliding window approach with information about the tag assigned to the preceding token. This tagging module enforces some basic constraints, e.g., BIO correct structure, arguments cannot overlap with clause nor chunk boundaries, discard `ARG0-5` arguments not present in PropBank frames for a certain verb, etc.

[1]Features extracted from partial parsing and Named Entities are common to Model 1 and 2, while features coming from full parse trees only apply to Model 2.

Relative position, *distance* in words and chunks, and *level of embedding* (in #clause-levels) with respect to the constituent.
Constituent path as described in (Gildea and Jurafsky, 2002) and all *3/4/5-grams* of path constituents beginning at the verb predicate or ending at the constituent.
Partial parsing path as described in (Carreras et al., 2004) and all *3/4/5-grams* of path elements beginning at the verb predicate or ending at the constituent.
Syntactic frame as described by Xue and Palmer (2004)

Table 3: Predicate–constituent features: Models 1/2

The *syntactic label* of the candidate constituent.
The constituent *head word*, *suffixes* of length 2, 3, and 4, *lemma*, and *POS tag*.
The constituent *content word*, *suffixes* of length 2, 3, and 4, *lemma*, *POS tag*, and *NE label*. Content words, which add informative lexicalized information different from the head word, were detected using the heuristics of (Surdeanu et al., 2003).
The *first and last constituent words* and their *POS tags*.
NE labels included in the candidate phrase.
Binary features to indicate the presence of *temporal cue words*, i.e. words that appear often in `AM-TMP` phrases in training.
For each TreeBank syntactic label we added a feature to indicate the *number of such labels* included in the candidate phrase.
The *sequence of syntactic labels* of the constituent immediate children.
The phrase *label*, *head word and POS tag* of the constituent parent, left sibling, and right sibling.

Table 4: Constituent structure features: Model 3

Model 3. The third individual SRL model makes the strong assumption that each predicate argument maps to one syntactic constituent. For example, in Figure 1 `ARG0` maps to a noun phrase, `ARGM-LOC` maps to a prepositional phrase etcetera. This assumption holds well on hand-corrected parse trees and simplifies significantly the SRL process because only one syntactic constituent has to be correctly classified in order to recognize one semantic argument. On the other hand, this approach is limited when using automatically-generated syntactic trees. For example, only slightly over 91% of the arguments can be mapped to one of the syntactic constituents produced by the Charniak parser.

Using a bottom-up approach, Model 3 maps each argument to the first syntactic constituent that has the exact same boundaries and then climbs as high as possible in the tree across unary production chains. We currently ignore all arguments that do not map to a single syntactic constituent.

The predicate *word* and *lemma*.
The predicate *voice*. Same defi nition as Models 1 and 2.
A binary feature to indicate if the predicate is *frequent* (i.e., it appears more than twice in the training data) or not.
Sub-categorization rule. Same def. as Models 1 and 2.

Table 5: Predicate structure features: Model 3

The *path* in the syntactic tree between the argument phrase and the predicate as a chain of syntactic labels along with the traversal direction (up or down).
The *length* of the above syntactic path.
The *number of clauses* (`S*` phrases) in the path.
The *number of verb phrases* (`VP`) in the path.
The *subsumption count*, i.e. the difference between the depths in the syntactic tree of the argument and predicate constituents. This value is 0 if the two phrases share the same parent.
The *governing category*, which indicates if `NP` arguments are dominated by a sentence (typical for subjects) or a verb phrase (typical for objects).
We *generalize* syntactic paths with more than 3 elements using two templates: (a) $\texttt{Arg} \uparrow \texttt{Ancestor} \downarrow \texttt{N}_i \downarrow \texttt{Pred}$, where `Arg` is the argument label, `Pred` is the predicate label, `Ancestor` is the label of the common ancestor, and $\texttt{N}_i$ is instantiated with all the labels between `Pred` and `Ancestor` in the full path; and (b) $\texttt{Arg} \uparrow \texttt{N}_i \uparrow \texttt{Ancestor} \downarrow \texttt{Pred}$, where $\texttt{N}_i$ is instantiated with all the labels between `Arg` and `Ancestor` in the full path.
The *surface distance* between the predicate and the argument phrases encoded as: the number of tokens, verb terminals (`VB*`), commas, and coordinations (`CC`) between the argument and predicate phrases, and a binary feature to indicate if the two constituents are adjacent.
A binary feature to indicate if the argument *starts with a predicate particle*, i.e. a token seen with the `RP*` POS tag and directly attached to the predicate in training.

Table 6: Predicate–constituent features: Model 3

Once the mapping process completes, Model 3 extracts a rich set of lexical, syntactic, and semantic features. Tables 4, 5, and 6 present these features organized in the same three categories as the previous Models 1 and 2 — see (Surdeanu and Turmo, 2005) for more details.

Similarly with Models 1 and 2, Model 3 trains one-vs-all classifiers using AdaBoost for the most common argument labels. To reduce the sample space, Model 3 selects training examples (both positive and negative) only from: (a) the first clause that includes the predicate, or (b) from phrases that appear to the left of the predicate in the sentence. More than 98% of the argument constituents fall into one of these classes.

At prediction time the classifiers are combined using a simple greedy technique that iteratively assigns to each predicate the argument classified with the highest confidence. For each predicate we consider as candidates all `AM` attributes, but only numbered attributes indicated in the corresponding PropBank frame. Additionally, this greedy strategy enforces a limited number of domain knowledge constraints in the generated solution: (a) arguments can not overlap in any form, and (b) no duplicate arguments are allowed for `ARG0-5`.

5 The Inference Model

The most important component of our inference model is candidate filtering, which decides if a candidate argument should be maintained in the global solution or not. Candidate filtering is implemented as a single binary classifier that uses only features extracted from the solutions provided by the individual systems. For robustness, we do not use any features that are specific to any of the individual models, nor the confidence value of their classifiers.

Table 7 lists the features extracted from each candidate argument by the filtering classifier. For simplicity we have focused only on attributes that: (a) are readily available in the solutions proposed by the individual classifiers, and (b) allow the gathering of simple and robust statistics. For example, the filtering classifier might learn that a candidate is to be trusted if: (a) two individual systems proposed it, (b) if its label is `ARG2` and it was generated by Model 1, or (c) if it was proposed by Model 2 within a certain argument sequence.

The candidate arguments that pass the filtering stage are incorporated in the global solution by the conflict resolution module, which enforces several domain specific constraints. We have currently implemented two constraints: (a) arguments can not overlap or embed other arguments; and (b) no duplicate arguments are allowed for the numbered arguments `ARG0-5`. Theoretically, the set of constraints can be extended with any other rules, but in our particular case, we know that some constraints, e.g. providing only arguments indicated in the corresponding PropBank frame, are already guaranteed by the individual models. Conflicts are solved with a straightforward greedy strategy: the pool of candidate arguments is inspected in descending order of the confidence values assigned by the filtering classifier, and candidates are appended to the global solution only if they do not violate any of the domain constraints with the arguments already selected. Our inference system currently has a sequential architecture, i.e. no feedback is sent from the conflict resolution module to candidate filtering.

The *label* of the candidate argument.
The *number of systems* that generated an argument with this label and span.
The *unique ids*, e.g. M1 and M2, of all the systems that generated an argument with this label and span.
The *argument sequence* for this predicate for all the systems that generated an argument with this label and span. For example, the argument sequence for the proposition illustrated in Figure 1 is: `ARG0 - ARGM-TMP - P - ARG1 - ARGM-LOC`.
The *number* and *unique ids* of all the systems that generated an argument with the *same span but different label*.
The *number* and *unique ids* of all the systems that generated an argument *included* in the current argument.
The *number* and *unique ids* of all the systems that generated an argument that *contains* the current argument.
The *number* and *unique ids* of all the systems that generated an argument that *overlaps* the current argument.

Table 7: Features used by the candidate filtering classifier.

6 Experimental Results

We trained the individual models using the complete CoNLL-2005 training set (PropBank/TreeBank sections 2 to 21). All models were developed using AdaBoost with decision trees of depth 4 (i.e. each branch may represent a conjunction of at most 4 basic features). Each classification model was trained for up to 2,000 rounds.

We applied some simplifications to keep training times and memory requirements inside admissible bounds: (a) we have limited the number of negative examples in Model 3 to the first 500,000; (b) we have trained only the most frequent argument labels: top 41 for Model 1, top 35 for Model 2, and top 24 for Model 3; and (c) we discarded all features occurring less than 15 times in the training set.

The models were tuned on a separate development partition (TreeBank section 24) and evaluated on two corpora: (a) TreeBank section 23, which consists of Wall Street Journal (WSJ) documents, and (b) on three sections of the Brown corpus, semantically annotated by the PropBank team for the CoNLL 2005 shared task evaluation. Table 8 sum-

WSJ	PProps	Precision	Recall	$F_{\beta=1}$
Model 1	48.45%	78.76%	72.44%	75.47 ± 0.8
Model 2	**52.04**%	79.65%	**74.92**%	**77.21** ± 0.8
Model 3	45.28%	**80.32**%	72.95%	76.46 ± 0.6
Brown	PProps	Precision	Recall	$F_{\beta=1}$
Model 1	30.85%	67.72%	58.29%	62.65 ± 2.1
Model 2	**36.44**%	71.82%	**64.03**%	**67.70** ± 1.9
Model 3	29.48%	**72.41**%	59.67%	65.42 ± 2.1

Table 8: Overall results of the individual models on the WSJ and Brown test sets.

marizes the results of the three models on the WSJ and Brown corpora. In that table we include the percentage of perfect propositions detected by each model ("PProps"), i.e. predicates recognized with all their arguments, the overall precision, recall, and $F_{\beta=1}$ measure[2].

The results summarized in Table 8 indicate that all individual systems have a solid performance. Although none of them would rank in the top 3 in this year's CoNLL evaluation (Carreras and Màrquez, 2005), their performance is comparable to the best individual systems presented at that evaluation exercise[3]. As expected, the models based on full parsing (2 and 3) perform better than the model based on partial syntax. But, interestingly, the difference is not large (e.g., less than 2 points in $F_{\beta=1}$ in the WSJ corpus), evincing that having base syntactic chunks and clause boundaries is enough to obtain a competitive performance with a simple system.

Consistently with other systems evaluated on the Brown corpus, all our models experience a severe performance drop in this corpus, due to the lower performance of the linguistic processors.

6.1 Performance of Combination Systems

We have trained the candidate filtering binary classifier on one third of the training partition. Its training data was generated using individual models trained on the other two thirds of the training partition. The classifier was developed using Support Vector Machines (SVM) with a polynomial kernel of degree 2. We trained combined models for all 4 possible combinations of our 3 individual models.

[2]The signifi cance intervals for the F_1 measure have been obtained using bootstrap resampling (Noreen, 1989). F_1 rates outside of these intervals are assumed to be signifi cantly different from the related F_1 rate ($p < 0.05$).

[3]The best performing SRL systems at CoNLL were a combination of several subsystems. See section 7 for details.

Table 9 summarizes the performance of the combined systems on the WSJ and Brown corpora.[4] The combined systems are compared against a baseline combination system, which merges *all* the arguments generated by the individual systems. For conflict resolution, the baseline uses the greedy strategy introduced in Section 5, but using as argument ordering criterion a radix sort that orders the candidate arguments in descending order of: number of models that agreed on this argument; argument length in tokens; and performance of the individual system[5].

Table 9 indicates that our combination strategy is always successful: the results of all combined systems improve upon their individual models and they are better the baseline when using the same number of individual models. As expected, the highest scoring combined system includes all three individual models. Its $F_{\beta=1}$ measure is 2.35 points higher than the best individual model (Model 2) in the WSJ test set and 1.30 points higher in the Brown test set. Somewhat surprisingly, the highest percentage of perfect propositions is not obtained by the overall best combination, but by the system that combines the two models based on full parsing (Models 2 and 3). This happens because Model 1 is the weakest performing of the bunch, hence its arguments, while providing useful information to the filtering classifier, decrease the number of perfect propositions when selected.

We consider these results encouraging given the simplicity of our inference model and the limited amount of training data used to train the candidate filtering classifier. Additionally, they compare favorably with respect to the best performing systems at CoNLL-2005 shared task (see Section 7).

6.2 Upper Limit of the Combination Strategy

To explore the potential of our approach we have constructed a hypothetical system where our candidate filtering module is replaced with a perfect classifier that selects only correct arguments and discards all others. Table 10 lists the results obtained on the WSJ and Brown corpora by this hypothetical system using all three individual models.

[4]For conciseness, in Table 9 we introduced the notation M1+2+3 to indicate the combination of Models 1, 2, and 3

[5]This combination produced the highest-scoring baseline model.

WSJ	PProps	Prec.	Recall	$F_{\beta=1}$
M1+2	51.30%	81.30%	74.13%	77.55 ± 0.7
M1+3	47.26%	81.21%	73.36%	77.08 ± 0.8
M2+3	**52.65**%	81.55%	75.30%	78.30 ± 0.7
M1+2+3	51.64%	**84.89**%	74.87%	**79.56** ± 0.7
baseline	51.09%	77.29%	**78.67**%	77.98 ± 0.7
Brown	**PProps**	**Prec.**	**Recall**	**$F_{\beta=1}$**
M1+2	35.95%	73.70%	62.93%	67.89 ± 2.0
M1+3	28.98%	72.83%	58.84%	65.09 ± 2.2
M2+3	**37.06**%	73.89%	63.30%	68.18 ± 2.2
M1+2+3	34.20%	**78.66**%	61.46%	**69.00** ± 2.1
baseline	33.58%	67.66%	**66.01**%	66.82 ± 1.8

Table 9: Overall results of the combination models on the WSJ and Brown test sets.

	Perfect props	Precision	Recall	$F_{\beta=1}$
WSJ	70.76%	99.12%	85.22%	91.64
Brown	51.87%	99.63%	74.32%	85.14

Table 10: Performance upper limit on the test sets.

Table 10 indicates that the upper limit of proposed approach is relatively high: the $F_{\beta=1}$ of this hypothetical system is over 12 points higher than our best combined system in the WSJ test set, and over 16 points higher in the Brown corpus. These results indicate that the potential of our combination strategy is high, especially when compared with re-ranking strategies, which are limited to the performance of the best *complete* solution in the candidate pool. By allowing the re-combination of arguments from the individual candidate solutions we raise this threshold significantly. Table 11 lists the contribution of the individual models to this upper limit on the WSJ corpus. For conciseness, we list only the "core" numbered arguments. "$\cap$ of 3" indicates the percentage of correct arguments where all 3 models agreed, "$\cap$ of 2" indicates the percentage of correct arguments where any 2 models agreed, and the other columns indicate the percentage of correct arguments detected by a single model. Table 11 indicates that, as expected, two or more individual models agreed on a large percentage of the correct arguments. Nevertheless, a significant number of correct arguments, e.g. over 22% of ARG3, come from a *single* individual system. This proves that, in order to achieve maximum performance, one has to look beyond simple voting strategies that favor arguments with high agreement between individual systems.

	$\cap$ of 3	$\cap$ of 2	M1	M2	M3
ARG0	80.45%	12.10%	3.47%	2.14%	1.84%
ARG1	69.82%	17.83%	7.45%	2.77%	2.13%
ARG2	56.04%	22.32%	12.20%	4.95%	4.49%
ARG3	56.03%	21.55%	12.93%	5.17%	4.31%
ARG4	65.85%	20.73%	6.10%	2.44%	4.88%

Table 11: Contribution of the individual systems to the upper limit, for ARG0–ARG4 in the WSJ test set.

	WSJ		Brown	
	PProps	$F_{\beta=1}$	PProps	$F_{\beta=1}$
koomen	53.79%	**79.44** ± 0.8	32.34%	**67.75** ± 1.8
haghighi	**56.52**%	78.45 ± 0.8	**37.06**%	67.71 ± 2.0
pradhan	50.14%	77.37 ± 0.7	36.44%	67.07 ± 2.0

Table 12: Results of the best combined systems at CoNLL-2005.

7 Related Work

The best performing systems at the CoNLL-2005 shared task included a combination of different base subsystems to increase robustness and to gain coverage and independence from parse errors. Therefore, they are closely related to the work of this paper. Table 12 summarizes their results under exactly the same experimental setting.

Koomen et al. (2005) used a 2 layer architecture similar to ours. The pool of candidates is generated by running a full syntax SRL system on alternative input information (Collins parsing, and 5-best trees from Charniak's parser). The combination of candidates is performed in an elegant global inference procedure as constraint satisfaction, which, formulated as Integer Linear Programming, can be solved efficiently. Interestingly, the generalized inference layer allows to include in the objective function, jointly with the candidate argument scores, a number of linguistically-motivated constraints to obtain a coherent solution. Differing from the strategy presented in this paper, their inference layer does not include learning. Also, they require confidence values from individual classifiers. This is the best performing system at CoNLL-2005.

Haghighi et al. (2005) implemented a double re-ranking model on top of the base SRL models to select the most probable solution among a set of candidates. The re-ranking is performed, first, on a set of n-best solutions obtained by the base system run on a single parse tree, and, then, on the set of best-candidates coming from the n-best parse trees. The re-ranking approach allows to define global complex features applying to complete candidate solutions to train the rankers. The main drawback, compared to our approach, is that re-ranking does not permit to combine different solutions since it is forced to select a complete candidate solution. This fact implies that the performance upper limit strongly depends on the ability of the base model to generate the complete correct solution in the set of n-best candidates.

Finally, Pradhan et al. (2005b) followed a stacking approach by learning two individual systems based on full syntax, whose outputs are used to generate features to feed the training stage of a final chunk-by-chunk SRL system. Although the fine granularity of the chunking-based system allows to recover from parsing errors, we find this combination scheme quite ad-hoc because it forces to break argument candidates into chunks in the last stage.

8 Conclusions

This paper introduces a novel, robust combination strategy for semantic role labeling. Our approach is separated in two stages: a candidate generation phase, which combines the solutions generated by several individual models into a pool of candidate arguments, followed by a simple inference model that filters the candidate arguments using a single binary classifier and then enforces an arbitrary number of domain-specific constraints.

The proposed approach has several advantages. First, because it combines the solutions provided by the individual models, the inference model can recover from errors produced in the generation phase. Second, due to the diversity of the individual models employed, the candidate pool contains a high percentage of the correct arguments. And lastly, our approach is flexible and robust: it can incorporate any SRL model in the candidate generation stage because it does not require that the individual SRL models provide any information, e.g. classification confidence values, other than an argument solution.

Our results are better than the state of the art using automatically-generated syntactic information. These results are encouraging considering the simplicity of the proposed approach.

Acknowledgments

This research has been partially supported by the European Commission (CHIL project, IP-506909). Mihai Surdeanu is a research fellow within the Ramón y Cajal program of the Spanish Ministry of Education and Science.

References

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

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

X. Carreras, L. Màrquez, and G. Chrupała. 2004. Hierarchical recognition of propositional arguments with perceptrons. In *Proceedings of CoNLL 2004 shared task*.

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

A. Haghighi, K. Toutanova, and C. Manning. 2005. A joint model for semantic role labeling. In *Proceedings of CoNLL-2005 shared task*.

P. Kingsbury, M. Palmer, and M. Marcus. 2002. Adding semantic annotation to the Penn Treebank. In *Proceedings of the Human Language Technology Conference*.

P. Koomen, V. Punyakanok, D. Roth, and W. Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL-2005 shared task*.

L. Màrquez, P. Comas, J. Giménez, and N. Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of CoNLL-2005 shared task*.

S. Narayanan and S. Harabagiu. 2004. Question answering based on semantic structures. In *International Conference on Computational Linguistics (COLING 2004)*.

E. W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005a. Support vector learning for semantic argument classifi cation. *Machine Learning, to appear*.

S. Pradhan, K. Hacioglu, W. Ward, J. H. Martin, and D. Jurafsky. 2005b. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL-2005*.

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confi dence-rated predictions. *Machine Learning*, 37(3).

M. Surdeanu and J. Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *Proceedings of CoNLL-2005 shared task*.

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

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