Semantic Role Labeling as Sequential Tagging

Lluís Màrquez, **Pere Comas**, **Jesús Giménez** and **Neus Català**
TALP Research Centre
Technical University of Catalonia (UPC)
{lluism,pcomas,jgimenez,ncatala}@lsi.upc.edu

Abstract

In this paper we present a semantic role labeling system submitted to the CoNLL-2005 shared task. The system makes use of partial and full syntactic information and converts the task into a sequential BIO-tagging. As a result, the labeling architecture is very simple . Building on a state-of-the-art set of features, a binary classifier for each label is trained using AdaBoost with fixed depth decision trees. The final system, which combines the outputs of two base systems performed F_1=76.59 on the official test set. Additionally, we provide results comparing the system when using partial vs. full parsing input information.

1 Goals and System Architecture

The goal of our work is twofold. On the one hand, we want to test whether it is possible to implement a competitive SRL system by reducing the task to a sequential tagging. On the other hand, we want to investigate the effect of replacing partial parsing information by full parsing. For that, we built two different individual systems with a shared sequential strategy but using UPC chunks-clauses, and Charniak's parses, respectively. We will refer to those systems as $\mathrm{PP_{UPC}}$ and $\mathrm{FP_{CHA}}$, hereinafter.

Both partial and full parsing annotations provided as input information are of hierarchical nature. Our system navigates through these syntactic structures in order to select a subset of constituents organized sequentially (i.e., non embedding). Propositions are treated independently, that is, each target verb generates a sequence of tokens to be annotated. We call this pre-processing step **sequentialization**.

The sequential tokens are selected by exploring the sentence spans or regions defined by the clause boundaries[1]. The top-most syntactic constituents falling inside these regions are selected as tokens. Note that this strategy is independent of the input syntactic annotation explored, provided it contains clause boundaries. It happens that, in the case of full parses, this node selection strategy is equivalent to the pruning process defined by Xue and Palmer (2004), which selects sibling nodes along the path of ancestors from the verb predicate to the root of the tree[2]. Due to this pruning stage, the upper-bound recall figures are 95.67% for $\mathrm{PP_{UPC}}$ and 90.32% for $\mathrm{FP_{CHA}}$. These values give F_1 performance upper bounds of 97.79 and 94.91, respectively, assuming perfect predictors (100% precision).

The nodes selected are labeled with B-I-O tags depending if they are at the beginning, inside, or outside of a verb argument. There is a total of 37 argument types, which amount to 37*2+1=75 labels.

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 prob-

[1] Regions to the right of the target verb corresponding to ancestor clauses are omitted in the case of partial parsing.

[2] With the unique exception of the exploration inside sibling PP constituents proposed by (Xue and Palmer, 2004).

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 193–196, Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

lems to address multi-class classification.

AdaBoost binary classifiers are used for **labeling** test sequences in a left-to-right tagging scheme using a recurrent sliding window approach with information about the tag assigned to the preceding token. This tagging module ensures some basic constraints, e.g., BIO correct structure, arguments do not cross clause boundaries nor base chunk boundaries, `A0-A5` arguments not present in PropBank frames for a certain verb are not allowed, etc. We also tried beam search on top of the classifiers' predictions to find the sequence of labels with highest sentence-level probability (as a summation of individual predictions). But the results did not improve the basic greedy tagging.

Regarding **feature representation**, we used all input information sources, with the exception of verb senses and Collins' parser. We did not contribute with significantly original features. Instead, we borrowed most of them from the existing literature (Gildea and Jurafsky, 2002; Carreras et al., 2004; Xue and Palmer, 2004). Broadly speaking, we considered features belonging to four categories[3]:

(1) On the verb predicate:

- **Form**; **Lemma**; **POS tag**; **Chunk type** and **Type of verb phrase** in which verb is included: *single-word* or *multi-word*; **Verb voice**: *active*, *passive*, *copulative*, *infinitive*, or *progressive*; Binary flag indicating if the verb is a **start/end** of a clause.
- **Subcategorization**, i.e., the phrase structure rule expanding the verb parent node.

(2) On the focus constituent:

- **Type**; **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 in the constituent.
- **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 described 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).

(3) Context of the focus constituent:

- **Previous and following words and POS tags** 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.

(4) Relation between predicate and constituent:

- **Relative position**; **Distance** in words and chunks; **Level of embedding** with respect to the constituent: in number of clauses.
- **Constituent path** as described in (Gildea and Jurafsky, 2002); 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); 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)

2 Experimental Setting and Results

We trained the classification models using the complete training set (sections from 02 to 21). Once converted into one sequence per target predicate, the resulting set amounts 1,049,049 training examples in the PP_{UPC} model and 828,811 training examples in the FP_{CHA} model. The average number of labels per argument is 2.071 and 1.068, respectively. This fact makes "I" labels very rare in the FP_{CHA} model.

When running AdaBoost, we selected as weak rules decision trees of fixed depth 4 (i.e., each branch may represent a conjunction of at most 4 basic features) and trained a classification model per label for up to 2,000 rounds.

We applied some simplifications to keep training times and memory requirements inside admissible bounds. First, we discarded all the argument labels that occur very infrequently and trained only the 41 most frequent labels in the case of PP_{UPC} and the 35 most frequent in the case of FP_{CHA}. The remaining labels where joined in a new label "`other`" in training and converted into "`O`" whenever the SRL system assigns a "`other`" label during testing. Second, we performed a simple frequency filtering by discarding those features occurring less than 15 times in the training set. As an

[3] Features extracted from partial parsing and Named Entities are common to PP_{UPC} and FP_{CHA} models, while features coming from Charniak parse trees are implemented exclusively in the FP_{CHA} model.

exception, the frequency threshold for the features referring to the verb predicate was set to 3. The final number of features we worked with is 105,175 in the case of PP_{UPC} and 80,742 in the case of FP_{CHA}.

Training with these very large data and feature sets becomes an issue. Fortunately, we could split the computation among six machines in a Linux cluster. Using our current implementation combining Perl and C++ we could train the complete models in about 2 days using memory requirements between 1.5GB and 2GB. Testing with the ensembles of 2,000 decision trees per label is also not very efficient, though the resulting speed is admissible, e.g., the development set is tagged in about 30 minutes using a standard PC.

The overall results obtained by our individual PP_{UPC} and FP_{CHA} SRL systems are presented in table 1, with the best results in boldface. As expected, the FP_{CHA} system significantly outperformed the PP_{UPC} system, though the results of the later can be considered competitive. This fact is against the belief, expressed as one of the conclusions of the CoNLL-2004 shared task, that full-parsing systems are about 10 F_1 points over partial-parsing systems. In this case, we obtain a performance difference of 2.18 points in favor of FP_{CHA}.

Apart from resulting performance, there are additional advantages when using the FP_{CHA} approach. Due to the coarser granularity of sequence tokens, FP_{CHA} sequences are shorter. There are 21% less training examples and a much lower quantity of "I" tags to predict (the mapping between syntactic constituents and arguments is mostly one-to-one). As a consequence, FP_{CHA} classifiers train faster with less memory requirements, and achieve competitive results (near the optimal) with much less rounds of boosting. See figure 1. Also related to the token granularity, the number of completely correct outputs is 4.13 points higher in FP_{CHA}, showing that the resulting labelings are structurally better than those of PP_{UPC}.

Interestingly, the PP_{UPC} and FP_{CHA} systems make quite different argument predictions. For instance, FP_{CHA} is better at recognizing `A0` and `A1` arguments since parse constituents corresponding to these arguments tend to be mostly correct. Comparatively, PP_{UPC} is better at recognizing `A2-A4` arguments since they are further from the verb predicate

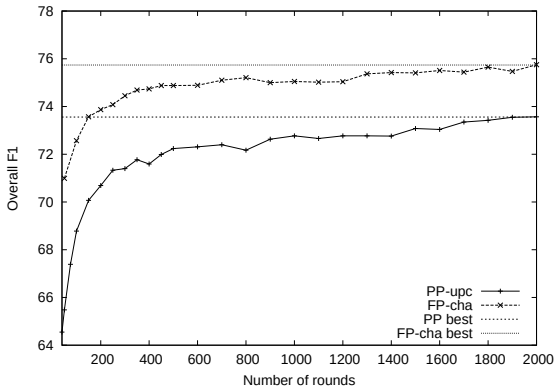

Figure 1: Overall F_1 performance of individual systems on the development set with respect to the number of learning rounds

	Perfect props	Precision	Recall	$F_{\beta=1}$
PP_{UPC}	47.38%	76.86%	70.55%	73.57
FP_{CHA}	**51.51**%	78.08%	73.54%	75.75
Combined	51.39%	**78.39**%	**75.53**%	**76.93**

Table 1: Overall results of the individual systems on the development set.

and tend to accumulate more parsing errors, while the fine granularity of the PP_{UPC} sequences still allow to capture them[4]. Another interesting observation is that the precision of both systems is much higher than the recall.

The previous two facts suggest that combining the outputs of the two systems may lead to a significant improvement. We experimented with a greedy combination scheme for joining the maximum number of arguments from both solutions in order to increase coverage and, hopefully, recall. It proceeds departing from an empty solution by: First, adding all the arguments from FP_{CHA} in which this method performs best; Second, adding all the arguments from PP_{UPC} in which this method performs best; and Third, making another loop through the two methods adding the arguments not considered in the first loop. At each step, we require that the added arguments do not overlap/embed with arguments in the current solution and also that they do not introduce repetitions of `A0-A5` arguments. The results on the

[4] As an example, the F_1 performance of PP_{UPC} on `A0` and `A2` arguments is 79.79 and 65.10, respectively. The performance of FP_{CHA} on the same arguments is 84.03 and 62.36.

	Precision	Recall	$F_{\beta=1}$
Development	78.39%	75.53%	76.93
Test WSJ	79.55%	76.45%	77.97
Test Brown	70.79%	64.35%	67.42
Test WSJ+Brown	78.44%	74.83%	76.59

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	79.55%	76.45%	77.97
A0	87.11%	86.28%	86.69
A1	79.60%	76.72%	78.13
A2	69.18%	67.75%	68.46
A3	76.38%	56.07%	64.67
A4	79.78%	69.61%	74.35
A5	0.00%	0.00%	0.00
AM-ADV	59.15%	52.37%	55.56
AM-CAU	73.68%	57.53%	64.62
AM-DIR	71.43%	35.29%	47.24
AM-DIS	77.14%	75.94%	76.54
AM-EXT	63.64%	43.75%	51.85
AM-LOC	62.74%	54.27%	58.20
AM-MNR	54.33%	52.91%	53.61
AM-MOD	96.16%	95.46%	95.81
AM-NEG	99.13%	98.70%	98.91
AM-PNC	53.49%	40.00%	45.77
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	77.68%	78.75%	78.21
R-A0	86.84%	88.39%	87.61
R-A1	75.32%	76.28%	75.80
R-A2	54.55%	37.50%	44.44
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	69.81%	71.15%	70.48
V	99.16%	99.16%	99.16

Table 2: Overall results (top) and detailed results on the WSJ test (bottom).

development set (presented in table 1) confirm our expectations, since a performance increase of 1.18 points over the best individual system was observed, mainly caused by recall improvement. The final system we presented at the shared task performs exactly this solution merging procedure. When applied on the WSJ test set, the combination scheme seems to generalize well, since an improvement is observed with respect to the development set. See the official results of our system, which are presented in table 2. Also from that table, it is worth noting that the F_1 performance drops by more than 9 points when tested on the Brown test set, indicating that the results obtained on the WSJ corpora do not generalize well to corpora with other genres. The study of the sources of this lower performance deserves further investigation, though we do not believe that it is attributable to the greedy combination scheme.

3 Conclusions

We have presented a simple SRL system submitted to the CoNLL-2005 shared task, which treats the SRL problem as a sequence tagging task (using a BIO tagging scheme). Given the simplicity of the approach, we believe that the results are very good and competitive compared to the state-of-the-art. We also provided a comparison between two SRL systems sharing the same architecture, but build on partial vs. full parsing, respectively. Although the full parsing approach obtains better results and has some implementation advantages, the partial parsing system shows also a quite competitive performance. The results on the development set differ in 2.18 points, but the outputs generated by the two systems are significantly different. The final system, which scored F_1=76.59 in the official test set, is a combination of both individual systems aiming at increasing coverage and recall.

Acknowledgements

This research has been partially supported by the European Commission (CHIL project, IP-506909). Jesús Giménez is a research fellow from the Spanish Ministry of Science and Technology (ALIADO project, TIC2002-04447-C02). We would like to thank also Xavier Carreras for providing us with many software components and Mihai Surdeanu for fruitful discussions on the problem and feature engineering.

References

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

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

R. E. Schapire and Y. Singer. 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3).

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.