Hierarchical Semantic Role Labeling

Alessandro Moschitti[◇] **Ana-Maria Giuglea**[◇] **Bonaventura Coppola**[†‡] **Roberto Basili**[◇]
moschitti@info.uniroma2.it ana-maria.giuglea@topex.ro coppolab@itc.it basili@info.uniroma2.it

[◇] DISP - University of Rome "Tor Vergata", Rome, Italy
[†] ITC-Irst, [‡] DIT - University of Trento, Povo, Trento, Italy

Abstract

We present a four-step hierarchical SRL strategy which generalizes the classical two-level approach (boundary detection and classification). To achieve this, we have split the classification step by grouping together roles which share linguistic properties (e.g. Core Roles versus Adjuncts). The results show that the non-optimized hierarchical approach is computationally more efficient than the traditional systems and it preserves their accuracy.

1 Introduction

For accomplishing the CoNLL 2005 Shared Task on Semantic Role Labeling (Carreras and Màrquez, 2005), we capitalized on our experience on the semantic shallow parsing by extending our system, widely experimented on PropBank and FrameNet (Giuglea and Moschitti, 2004) data, with a two-step boundary detection and a hierarchical argument classification strategy.

Currently, the system can work in both basic and enhanced configuration. Given the parse tree of an input sentence, the basic system applies (1) a boundary classifier to select the nodes associated with correct arguments and (2) a multi-class labeler to assign the role type. For such models, we used some of the linear (e.g. (Gildea and Jurasfky, 2002; Pradhan et al., 2005)) and structural (Moschitti, 2004) features developed in previous studies.

In the enhanced configuration, the boundary annotation is subdivided in two steps: a first pass in which we label argument boundary and a second pass in which we apply a simple heuristic to eliminate the argument overlaps. We have also tried some strategies to learn such heuristics automatically. In order to do this we used a tree kernel to classify the subtrees associated with correct predicate argument structures (see (Moschitti et al., 2005)). The rationale behind such an attempt was to exploit the correlation among potential arguments.

Also, the role labeler is divided into two steps: (1) we assign to the arguments one out of four possible class labels: *Core Roles*, *Adjuncts*, *Continuation Arguments* and *Co-referring Arguments*, and (2) in each of the above class we apply the set of its specific classifiers, e.g. A0,..,A5 within the Core Role class. As such grouping is relatively new, the traditional features may not be sufficient to characterize each class. Thus, to generate a large set of features automatically, we again applied tree kernels.

Since our SRL system exploits the PropBank formalism for internal data representation, we developed ad-hoc procedures to convert back and forth to the CoNLL Shared Task format. This conversion step gave us useful information about the amount and the nature of the parsing errors. Also, we could measure the frequency of the mismatches between syntax and role annotation.

In the remainder of this paper, Section 2 describes the basic system configuration whereas Section 3 illustrates its enhanced properties and the hierarchical structure. Section 4 describes the experimental setting and the results. Finally, Section 5 summarizes

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 201–204, Ann Arbor, June 2005.

our conclusions.

2 The Basic Semantic Role Labeler

In the last years, several machine learning approaches have been developed for automatic role labeling, e.g. (Gildea and Jurasfky, 2002; Pradhan et al., 2005). Their common characteristic is the adoption of flat feature representations for predicate-argument structures. Our basic system is similar to the one proposed in (Pradhan et al., 2005) and it is described hereafter.

We divided the predicate argument labeling in two subtasks: (a) the detection of the arguments related to a target, i.e. all the compounding words of such argument, and (b) the classification of the argument type, e.g. A0 or AM. To learn both tasks we used the following algorithm:

1. Given a sentence from the *training-set*, generate a full syntactic parse-tree;
2. Let $\mathcal{P}$ and $\mathcal{A}$ be respectively the set of predicates and the set of parse-tree nodes (i.e. the potential arguments);
3. For each pair $<p, a> \in \mathcal{P} \times \mathcal{A}$:
   - extract the feature representation set, $F_{p,a}$;
   - if the subtree rooted in a covers exactly the words of one argument of p, put $F_{p,a}$ in T^+ (positive examples), otherwise put it in T^- (negative examples).

We trained the SVM boundary classifier on T^+ and T^- sets and the role labeler i on the T_i^+, i.e. its positive examples and T_i^-, i.e. its negative examples, where $T^+ = T_i^+ \cup T_i^-$, according to the ONE-vs.-ALL scheme. To implement the multi-class classifiers we select the argument associated with the maximum among the SVM scores.

To represent the $F_{p,a}$ pairs we used the following features:
- the *Phrase Type*, *Predicate Word*, *Head Word*, *Governing Category*, *Position* and *Voice* defined in (Gildea and Jurasfky, 2002);
- the *Partial Path*, *Compressed Path*, *No Direction Path*, *Constituent Tree Distance*, *Head Word POS*, *First and Last Word/POS in Constituent*, *SubCategorization* and *Head Word of Prepositional Phrases* proposed in (Pradhan et al., 2005); and
- the *Syntactic Frame* designed in (Xue and Palmer, 2004).

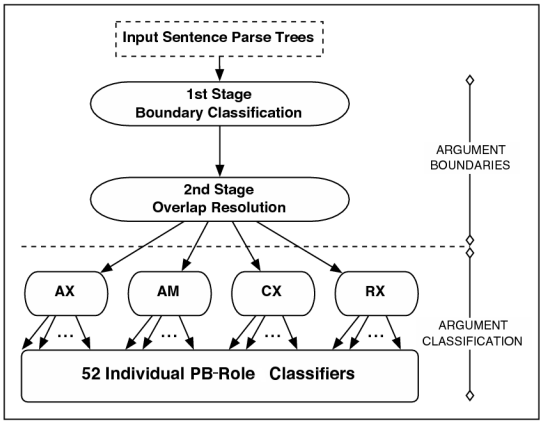

Figure 1: Architecture of the Hierarchical Semantic Role Labeler.

3 Hierarchical Semantic Role Labeler

Having two phases for argument labeling provides two main advantages: (1) the efficiency is increased as the negative boundary examples, which are almost all parse-tree nodes, are used with one classifier only (i.e. the boundary classifier), and (2) as arguments share common features that do not occur in the non-arguments, a preliminary classification between arguments and non-arguments advantages the boundary detection of roles with fewer training examples (e.g. A4). Moreover, it may be simpler to classify the type of roles when the not-argument nodes are absent.

Following this idea, we generalized the above two level strategy to a four-step role labeling by grouping together the arguments sharing similar properties. Figure 1, shows the hierarchy employed for argument classification:

During the first phase, we select the parse tree nodes which are likely predicate arguments. An SVM with moderately high recall is applied for such purpose.

In the second phase, a simple heuristic which selects *non-overlapping* nodes from those derived in the previous step is applied. Two nodes n_1 and n_2 do not overlap if n_1 is not ancestor of n_2 and viceversa. Our heuristic simply eliminates the nodes that cause the highest number of overlaps. We have also studied how to train an overlap resolver by means of tree kernels; the promising approach and results can be found in (Moschitti et al., 2005).

In the third phase, we classify the detected arguments in the following four classes: AX, i.e. *Core*

Arguments, AM, i.e. *Adjuncts*, CX, i.e. *Continuation Arguments* and RX, i.e. the *Co-referring Arguments*. The above classification relies on linguistic reasons. For example *Core arguments* class contains the arguments specific to the verb frames while *Adjunct Arguments* class contains arguments that are shared across all verb frames.

In the fourth phase, we classify the members within the classes of the previous level, e.g. `A0` vs. `A1`, ..., `A5`.

4 The Experiments

We experimented our approach with the CoNLL 2005 Shared Task standard dataset, i.e. the PennTree Bank, where sections from 02 to 21 are used as training set, Section 24 as development set (Dev) and Section 23 as the test set (WSJ). Additionally, the Brown corpus' sentences were also used as the test set (Brown). As input for our feature extractor we used only the Charniak's parses with their POSs.

The evaluations were carried out with the SVM-light-TK software (Moschitti, 2004) available at `http://ai-nlp.info.uniroma2.it/moschitti/` which encodes the tree kernels in the SVM-light software (Joachims, 1999). We used the default polynomial kernel (degree=3) for the linear feature representations and the tree kernels for the structural feature processing.

As our feature extraction module was designed to work on the PropBank project annotation format (i.e. the *prop.txt* index file), we needed to generate it from the CoNLL data. Each PropBank annotation refers to a parse tree node which exactly covers the target argument but when using automatic parses such node may not exist. For example, on the CoNLL Charniak's parses, (sections 02-21 and 24), we discovered that this problem affects 10,293 out of the 241,121 arguments (4.3%) and 9,741 sentences out of 87,257 (11.5%). We have found out that most of the errors are due to wrong parsing attachments. This observation suggests that the capability of discriminating between correct and incorrect parse trees is a key issue in the boundary detection phase and it must be properly taken into account.

4.1 Basic System Evaluation

For the boundary classifier we used a SVM with the polynomial kernel of degree 3. We set the regularization parameter, `c`, to 1 and the cost factor, `j` to 7 (to have a slightly higher recall). To reduce the learning time, we applied a simple heuristic which removes the nodes covering the target predicate node. From the initial 4,683,777 nodes (of sections 02-21), the heuristic removed 1,503,100 nodes with a loss of 2.6% of the total arguments. However, as we started the experiments in late, we used only the 992,819 nodes from the sections 02-08. The classifier took about two days and half to converge on a 64 bits machine (2.4 GHz and 4Gb Ram).

The multiclassifier was built with 52 binary argument classifiers. Their training on all arguments from sec 02-21, (i.e. 242,957), required about a half day on a machine with 8 processors (32 bits, 1.7 GHz and overll 4Gb Ram).

We run the role multiclassifier on the output of the boundary classifier. The results on the Dev, WSJ and Brown test data are shown in Table 1. Note that, the overlapping nodes cause the generation of overlapping constituents in the sentence annotation. This prevents us to use the CoNLL evaluator. Thus, we used the overlap resolution algorithm also for the basic system.

4.2 Hierarchical Role Labeling Evaluation

As the first two phases of the hierarchical labeler are identical to the basic system, we focused on the last two phases. We carried out our studies over the Gold Standard boundaries in the presence of arguments that do not have a *perfect-covering* node in the Charniak trees.

To accomplish the third phase, we re-organized the flat arguments into the AX, AM, CX and RX classes and we built a single multi-classifier. For the fourth phase, we built a multi-classifier for each of the above classes: only the examples related to the target class were used, e.g. the AX mutliclassifier was designed with the A0,..,A5 ONE-vs-ALL binary classifiers.

In rows 2 and 3, Table 2 shows the numbers of training and development set instances. Row 4 contains the F_1 of the binary classifiers of the third phase whereas Row 5 reports the F_1 of the resulting multi-classifier. Row 6 presents the F_1s of the multi-classifiers of the fourth phase.

Row 7 illustrates the F_1 measure of the fourth phase classifier applied to the third phase output. Fi-

	Precision	Recall	$F_{\beta=1}$
Development	74.95%	73.10%	74.01
Test WSJ	76.55%	75.24%	75.89
Test Brown	65.92%	61.83%	63.81
Test WSJ+Brown	75.19%	73.45%	74.31

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	76.55%	75.24%	75.89
A0	81.05%	84.37%	82.67
A1	77.21%	74.12%	75.63
A2	67.02%	68.11%	67.56
A3	69.63%	54.34%	61.04
A4	74.75%	72.55%	73.63
A5	100.00%	40.00%	57.14
AM-ADV	55.23%	55.34%	55.28
AM-CAU	66.07%	50.68%	57.36
AM-DIR	50.62%	48.24%	49.40
AM-DIS	77.71%	78.44%	78.07
AM-EXT	68.00%	53.12%	59.65
AM-LOC	59.02%	63.09%	60.99
AM-MNR	67.67%	52.33%	59.02
AM-MOD	98.65%	92.56%	95.51
AM-NEG	97.37%	96.52%	96.94
AM-PNC	42.28%	45.22%	43.70
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	81.90%	74.52%	78.03
R-A0	79.50%	84.82%	82.07
R-A1	62.23%	75.00%	68.02
R-A2	100.00%	31.25%	47.62
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	100.00%	50.00%	66.67
R-AM-EXT	100.00%	100.00%	100.00
R-AM-LOC	85.71%	85.71%	85.71
R-AM-MNR	22.22%	33.33%	26.67
R-AM-TMP	67.69%	84.62%	75.21
V	97.34%	97.30%	97.32

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

nally, in Row 8, we report the F_1 of the basic system on the gold boundary nodes. We note that the basic system shows a slightly higher F_1 but is less computational efficient than the hierarchical approach.

5 Final Remarks

In this paper we analyzed the impact of a hierarchical categorization on the semantic role labeling task. The results show that such approach produces an accuracy similar to the flat systems with a higher efficiency. Moreover, some preliminary experiments show that each node of the hierarchy requires different features to optimize the associated multiclassifier. For example, we found that the SCF tree kernel (Moschitti, 2004) improves the AX multiclassifier

	AX	AM	CX	RX
# train. examples	172,457	59,473	2,954	7,928
# devel. examples	5,930	2,132	105	284
Phase III: binary class.	97.29	97.35	70.86	93.15
Phase III	95.99			
Phase IV	92.50	85.88	91.43	91.55
Phase III & IV	88.15			
Basic System	88.61			

Table 2: Hierarchical Semantic Role Labeler Results

whereas the PAF tree kernel seems more suited for the classification within the other classes, e.g. AM.

Future work on the optimization of each phase is needed to study the potential accuracy limits of the proposed hierarchical approach.

Acknowledgements

We wish to thank Daniele Pighin for his valuable support in the development of the SRL system.

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *proceedings of CoNLL'05*.

Daniel Gildea and Daniel Jurasfky. 2002. Automatic labeling of semantic roles. *Computational Linguistic*.

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge Discovering using FrameNet, VerbNet and PropBank. In *proceedings of the Workshop on Ontology and Knowledge Discovering at ECML'04, Pisa, Italy*.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin, and Roberto Basili. 2005. Engineering of syntactic features for shallow semantic parsing. In *proceedings of the Feature Engineering Workshop at ACL'05*, Ann Arbor, USA.

Alessandro Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. In *proceedings of ACL-2004*, Barcelona, Spain.

Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *to appear in Machine Learning Journal*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP'04*, Barcelona, Spain.