A Tree Kernel-based Shallow Semantic Parser for Thematic Role Extraction

Daniele Pighin[1,2] and Alessandro Moschitti[1]

[1] University of Trento, DIT
[2] FBK-irst - Trento, Italy

Abstract. We present a simple, two-steps supervised strategy for the identification and classification of thematic roles in natural language texts. We employ no external source of information but automatic parse trees of the input sentences. We use a few attribute-value features and tree kernel functions applied to specialized structured features. Different configurations of our thematic role labeling system took part in 2 tasks of the SemEval 2007 evaluation campaign, namely the closed tasks on semantic role labeling for the English and the Arabic languages. In this paper we present and discuss the system configuration that participated in the English semantic role labeling task and present new results obtained after the end of the evaluation campaign.

1 Introduction

The availability of large scale data sets of manually annotated predicate argument structures has recently favored the use of Machine Learning approaches to the design of automated Semantic Role Labeling (SRL) systems.

Research in this area is largely focused in two directions, namely the decomposition of the SRL task in a proper set of possibly disjoint problems and the selection and design of the features that can provide an effective and accurate model for the above learning problems. Though many different task decompositions have been attempted with more or less success, it is largely agreed that full syntactic information about the input free text sentences provides relevant clues about the position of an argument and the role it plays with respect to the predicate [1].

In this paper we present a system for the labeling of semantic roles that produces VerbNet [2] like annotations of free text sentences using only full syntactic parses of the input sentences. The labeling process is modeled as a cascade of two distinct classification steps: (1) boundary detection (BD), in which the word sequences that encode a thematic role for a given predicate are recognized, and (2) role classification (RC), in which the thematic role label is assigned with respect to the predicate. In order to be consistent with the underlying linguistic model, at the end of the process a set of simple heuristics are applied to ensure that only well formed annotations are output.

We use Support Vector Machines (SVMs) as our learning algorithm, and combine 2 different views of the incoming syntactic data: a) an explicit representation of a few relevant features in the form of attribute-value pairs, evaluated by a polynomial kernel, and b) structural features derived by applying canonical transformations to the sentence parse trees, evaluated by a tree kernel function.

All of this aspects will be discussed top-down in the remainder of this paper: Section 2 describes the architecture of our labeling system; Section 3 discusses the kernel function that we employ for the learning task; Section 4 discusses the linear and structural features that we use to represent the classifier examples; Section 5 describes the experimental setting and reports the accuracy of the system on the SemEval2007 closed task on semantic role labeling, along with the evaluation of different system configurations carried out after the end of the challenge; finally, Section 6 discusses the results that we obtained and presents our conclusions.

2 System Description

Given a target predicate word in a natural language sentence, a SRL system is meant to correctly identify all the arguments of the predicate. This problem is usually divided in two sub-tasks:

– the detection of the boundaries (i. e. the word span) of each argument, and
– the classification of the argument type, e.g. *Arg0* or *ArgM* in PropBank or *Agent* and *Goal* in FrameNet or VerbNet.

The standard approach to learn both the detection and the classification of predicate arguments is summarized by the following steps:

1. Given a sentence from the *training-set*, generate a full syntactic parse-tree;
2. let $\mathcal{P}$ and $\mathcal{A}$ be the set of predicates and the set of parse-tree nodes (i.e. the potential arguments), respectively;
3. for each pair $\langle p, a \rangle \in \mathcal{P} \times \mathcal{A}$:
   – extract the feature representation set, $F_{p,a}$;
   – if the sub-tree 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).

For instance, in Figure 2.a, for each combination of the predicate *approve* with any other tree node a that do not overlap with the predicate, a classifier example $F_{\text{approve},a}$ is generated. If a exactly covers one of the predicate arguments (in this case: *The charter*, *by the EC Commission* or *on Sept. 21*) it is regarded as a positive instance, otherwise it will be a negative one, e. g. $F_{\text{approve},(\text{NN charter})}$.

The T^+ and T^- sets are used to train the boundary classifier (BC). To train the role multi-class classifier (RM), T^+ can be reorganized as positive $T^+_{\text{arg}_i}$ and negative $T^-_{\text{arg}_i}$ examples for each argument i. In this way, an individual One-vs-All classifier for each argument i can be trained. We adopted this solution, according to [3], since it is simple and effective. In the classification phase, given an unseen sentence, all its $F_{p,a}$ are generated and classified by each individual role classifier. The role label associated with the maximum among the scores provided by the individual classifiers is eventually selected.

To make the annotations consistent with the underlying linguistic model, we employ a few simple heuristics to resolve the overlap situations that may occur, e. g. both *charter* and *the charter* in Figure 2 may be assigned a role:

- if more than two nodes are involved, i. e. a node d and two or more of its descendants n_i are classified as arguments, then assume that d is not an argument. This choice is justified by previous studies [4] showing that the accuracy of classification is higher for nodes located lower in the tree;
- if only two nodes are involved, i. e. they dominate each other, then keep the one with the highest classification score.

More complex, and generally more accurate, solutions can be adopted to improve the accuracy of the final annotation output by a SRL system[3]. Among other interesting strategies, [6] used a probabilistic joint evaluation over the whole predicate argument structure in order to establish a global relation between the local decisions of the role classifiers; [7] described a method based on Levenshtein-distance to *correct* the inconsistencies in the output sequence of role labels; [8] used a voting mechanism over multiple syntactic views in order to reduce the effect of parsing errors on the labeling accuracy.

Many supervised learning algorithms have more or less successfully been employed for SRL. We chose to use Support Vector Machines (SVMs) as our learning algorithm as they provide both a state-of-the-art learning model (in terms of accuracy) and the possibility of using kernel functions [9]. The kernels that we employ are described in the next section, whereas Section 4 presents the linear and structural features that we use to characterize the learning problem.

3 Kernel Functions for Semantic Role Labeling

In this study we adopted Support Vector Machines (SVMs) to exploit our new kernel functions. SVMs are learning algorithms which take training examples labeled with the class information as input and generate classification models. Each example e_i is represented in the feature space as a vector $\boldsymbol{x_i} \in \Re^n$ by means of a feature function

$$\phi : \mathcal{E} \rightarrow \Re^n \ ,$$

where $\mathcal{E}$ is the set of examples.

The generated model is a hyperplane $H(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ which separates positive from negative examples, where $\boldsymbol{w} \in \Re^n$ and $b \in \Re$ are parameters

[3] Indeed, previous versions of our SRL system sported a joint-inference model and a re-ranker mechanism based on tree kernels, as described in [5], which is currently offline due to changes in the interface of our feature extraction software module.

learned from data by applying the *Structural Risk Minimization principle* [9]. An example e_i is categorized in the target class only if $H(\boldsymbol{x}_i) \geq 0$.

The kernel trick allows the evaluation of the similarity between example pairs, $K(e_1, e_2)$, to be carried out without an explicit representation of the whole feature space, i.e. $K(e_1, e_2) = \phi(e_1) \cdot \phi(e_2) = \boldsymbol{x_1} \cdot \boldsymbol{x_2}$.

A traditional example is given by the polynomial kernel:

$$K_P(e_i, e_j) = (c + \boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d \ , \tag{1}$$

where c is a constant and d is the degree of the polynomial. This kernel generates the space of all conjunctions of feature groups up to d elements.

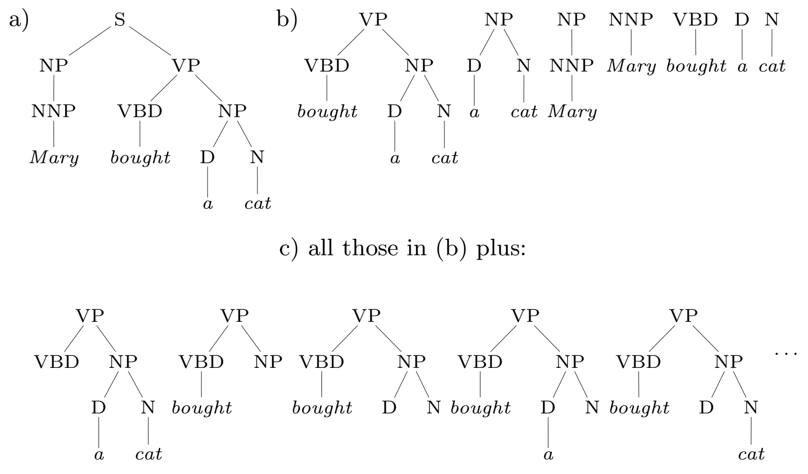

Fig. 1. Fragment space generated by an ST (b) and an SST (c) kernel from an example sub-tree (a).

A more abstract class of kernel functions evaluate the similarity between two discrete structures in terms of their overlap, generally measured as a function of the number of common substructures [10]. The kernels that we consider here represent trees in terms of their substructures (fragments). The kernel function detects if a tree sub-part (common to both trees) belongs to the feature space that we intend to generate. For such purpose, the desired fragments need to be described. As we consider syntactic parse trees, each node with its children is associated with a grammar production rule, where the symbol at the left-hand side corresponds to the parent and the symbols at the right-hand side are associated with the children. The terminal symbols of the grammar are always associated with tree leaves.

We define a SubTree (ST) [11] as a tree rooted in any non-terminal node along with all its descendants. For example, Figure 1 shows the parse tree of the sentence *Mary brought a cat* (a) together with its 7 STs (b). A SubSet

Tree (SST) [10] is a more general structure since its leaves can be non-terminal symbols. Figure 1(c) shows some of the SSTs for the same example sentence. The SSTs satisfy the constraint that grammatical rules cannot be broken. For example, `[VP [V NP]]` is an SST which has two non-terminal symbols, `V` and `NP`, as leaves. On the contrary, `[VP [V]]` is not an SST as it violates the production VP$\rightarrow$V NP.

The main idea underlying tree kernels is to compute the number of common substructures between two trees t_1 and t_2 without explicitly considering the whole fragment space. Let $\{f_1, f_2, ..\} = \mathcal{F}$ be the set of fragments and let the indicator function $I_i(n)$ be equal to 1 if the target f_i is rooted at node n and 0 otherwise. A tree kernel function $K_T(\cdot)$ over two trees is defined as:

$$K_T(t_1, t_2) = \sum_{n_1 \in N_{t_1}} \sum_{n_2 \in N_{t_2}} \Delta(n_1, n_2) \tag{2}$$

where N_{t_1} and N_{t_2} are the sets of nodes of t_1 and t_2, respectively. The function $\Delta(\cdot)$ evaluates the number of common fragments rooted in n_1 and n_2:

$$\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2) \tag{3}$$

We can compute Δ as follows:

1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (i. e. they are pre-terminal symbols) then $\Delta(n_1, n_2) = 1$;
3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\sigma + \Delta(c_{n_1}^j, c_{n_2}^j)) \tag{4}$$

where $\sigma \in \{0, 1\}$, $nc(n_1)$ is the number of the children of n_1 and c_n^j is the j-th child of node n. Note that, since the productions are the same, $nc(n_1) = nc(n_2)$.

When $\sigma = 0$, $\Delta(n_1, n_2)$ is equal to 1 only if $\forall j \quad \Delta(c_{n_1}^j, c_{n_2}^j) = 1$, i. e. all the productions associated with the children are identical. By recursively applying this property, it follows that the sub-trees in n_1 and n_2 are identical. Thus, Eq. 2 evaluates the subtree (ST) kernel. When $\sigma = 1$, $\Delta(n_1, n_2)$ evaluates the number of SSTs common to n_1 and n_2 as shown in [10].

In our case, each classifier example e_i is represented by a set of attribute-value features $\mathcal{L}_i$ and a structural feature t_i. The similarity between to examples e_i and e_j is evaluated by applying a polynomial kernel $K_P(\cdot)$ of degree $d = 3$ to the attribute-value features and an SST kernel $K_{SST}(\cdot)$ to the structured representation of the examples. The contribution of each kernel function is individually normalized and the tree kernel output is weighted by the w_k factor, which is set to 0.3. The resulting kernel function is the following:

$$K(e_i, e_j) = \frac{K_P(\mathcal{L}_i, \mathcal{L}_j)}{\|K_P(\mathcal{L}_j, \mathcal{L}_j)\|} + w_k \times \frac{K_{SST}(t_i, t_j)}{\|K_{SST}(t_i, t_j)\|} \ , \tag{5}$$

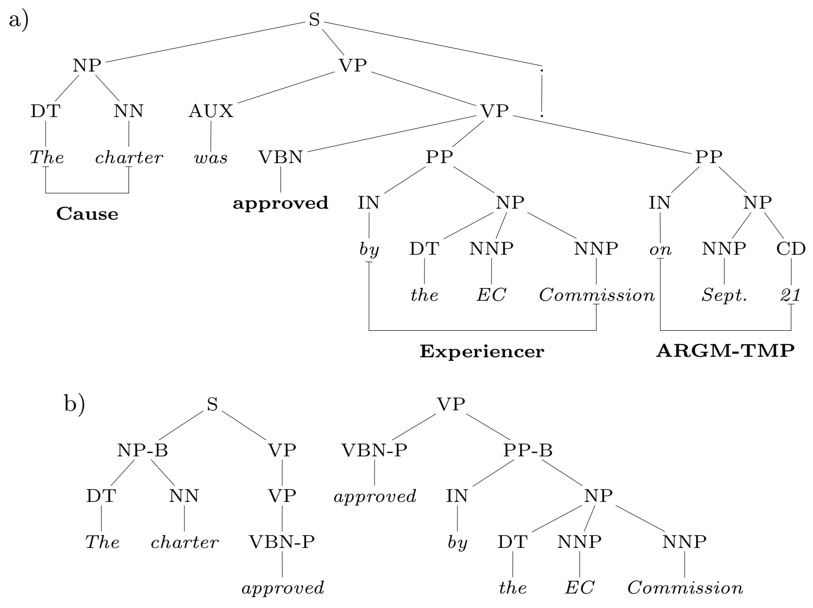

Fig. 2. A sentence parse tree (a) and two example AST_1^m structures relative to the predicate *approve* (b).

where

$$\|K_{SST}(t_i, t_j)\| = \sqrt{K_{SST}(t_i, t_i) \times K_{SST}(t_j, t_j)}\ ,$$

$$\|K_P(\mathcal{L}_j, \mathcal{L}_j)\| = \sqrt{K_P(\mathcal{L}_j, \mathcal{L}_j) \times K_P(\mathcal{L}_j, \mathcal{L}_j)}\ .$$

4 Features for Semantic Role Labeling

We explicitly represent as attribute-value pairs the following features of each $F_{p,a}$ pair:

- *Phrase Type*, *Predicate Word*, *Head Word*, *Position* and *Voice* as defined in [12];
- *Partial Path*, *No Direction Path*, *Head Word POS*, *First and Last Word/POS in Constituent* and *SubCategorization* as proposed in [3];
- *Syntactic Frame* as designed in [13].

We also employ structured features derived by the full parses in an attempt to capture relevant aspects that may not be emphasized by the explicit feature representation.

We indicate with structured features the basic syntactic structures extracted or derived from the sentence-parse tree by means of some canonical transformation. [14] and [4] defined several classes of structured features that were successfully employed with tree kernels for the different stages of an SRL process.

Set	Props	T	T^+	T^-
Train	15,838	793,104	45,157	747,947
Dev	1,606	75,302	4,291	71,011
Train - Dev	14,232	717,802	40,866	676,936
Test	3,094	144,965	6,931	138,034

Table 1. Composition of the dataset in terms of: number of annotations (Props); number of candidate argument nodes (T); positive (T^+) and negative (T^-) boundary classifier examples. The annotated test set has been released after the end of the evaluation period.

Figure 2 shows an example of the AST_1^m structures that we used for both the boundary detection and the argument classification stages.

5 Experiments

In this section we discuss the setup and the results of the experiments carried out on the dataset of the SemEval2007 closed task on SRL.

5.1 Setup

The training set comprises 15,838[4] training annotations organized on a per-verb basis. In order to build a development set (Dev), we sampled about one tenth, i. e. 1,606 annotations, of the original training set. For the final evaluation on the test set (Test), consisting of 3,094 annotations, we trained our classifiers on the whole training data. Statistics on the dataset composition are shown in Table 1.

The evaluations were carried out with the SVMLight-TK[5] software [15] which extends the SVMLight[6] package [16] with tree kernel functions. We used the default polynomial kernel (degree=3) for the linear features and a SubSet Tree (SST) kernel [10] for the comparison of AST_1^m structured features. The kernels are normalized and summed by assigning a weight of 0.3 to the TK, as explained in Section 3.

5.2 SemEval2007 Evaluation

The configuration presented at the SemEval2007 closed task on SRL comprised 50 boundary classifiers (BC), and 619 distinct role classifiers combined into 50 role multi-classifiers (RM) with an One-vs-All strategy, i. e. one BC and a RM per predicate word. The whole training took about 4 hours on a 64 bits machine (2.2GHz, 1GB RAM)[7].

[4] A bunch of unaligned annotations were removed from the dataset.

[5] `http://ai-nlp.info.uniroma2.it/moschitti/`

[6] `http://svmlight.joachims.org/`

[7] In order to have a faster development cycle, we limited to 60 thousands the training examples of the boundary classifier for the verb *say*. The accuracy on this relation

Task	Kernel(s)	Precision	Recall	$F_{\beta=1}$
BD	poly	94.34%	71.26%	81.19
	poly + TK	92.89%	76.09%	83.65
BD + RC	poly	88.72%	68.76%	77.47
	poly + TK	86.60%	72.40%	78.86

Table 2. SRL accuracy on the development test for the boundary detection (BD) and the complete SRL task (BD+RC) using the polynomial kernel alone (poly) or combined with a tree kernel function (poly + TK).

All the evaluations were carried out using the CoNLL2005 evaluator tool available at `http://www.lsi.upc.es/~srlconll/soft.html`.

Table 2 shows the aggregate results on boundary detection (BD) and the complete SRL task (BD+RC) on the development set using the polynomial kernel alone (poly) or in conjunction with the tree kernel and structured features (poly+TK). For both tasks, tree kernel functions do trigger automatic feature selection and improve the polynomial kernel by 2.46 and 1.39 F_1 points, respectively.

The SRL accuracy for each of the 47 distinct role labels is shown in Table 3 under Column *Split*. Column *#TI* lists the number of instances of each role in the test set. Many roles have very few positive examples both in the training and the test sets, and therefore have little or no impact on the overall accuracy which is dominated by the few roles which are very frequent, such as *Theme*, *Agent*, *Topic* and *ARGM-TMP* which account for almost 80% of all the test roles.

Table 4 shows the results of the two systems that participated in the SemEval 2007 closed task on English SRL. The winning system (labeled *UBC-UPC*, [17]) uses a sequential approach to role labeling and unlike our own exploits a) verb-sense information to restrict the possibile sequences of output roles, and b) WordNet-based selectional preferences on the potential arguments. The result is a much more accurate SRL system, i. e. 83.66 vs 75.44 F1 points in the official evaluation.

5.3 Further Evaluation

After the official evaluation was over we run another series of experiments in order to evaluate the effect of different configurations of the SRL system. We did not change any parameter in the setup of the SVMs but used the poly+TK kernel to compare different strategies for training the BC and the RM. These strategies are:

- training both the BC and the RM by splitting the data on a per-predicate basis, i. e. the configuration that participated in the SemEval evaluation;

is still very high, as we measured an overall F_1 of 87.18 on the development set and of 85.13 on the test set

		Split			**Monolithic**		
Role	**#TI**	**Precision**	**Recall**	$\mathbf{F}_{\beta=1}$	**Precision**	**Recall**	$\mathbf{F}_{\beta=1}$
Overall	6931	81.58%	70.16%	75.44	81.25%	74.42%	77.69
ARG2	4	100.00%	25.00%	40.00	50.00%	25.00%	33.33
ARG3	17	61.11%	64.71%	62.86	52.63%	58.82%	55.56
ARG4	4	0.00%	0.00%	0.00	0.00%	0.00%	0.00
ARGM-ADV	188	55.14%	31.38%	40.00	52.45%	39.89%	45.32
ARGM-CAU	13	50.00%	23.08%	31.58	66.67%	30.77%	42.11
ARGM-DIR	4	100.00%	25.00%	40.00	100.00%	25.00%	40.00
ARGM-EXT	3	0.00%	0.00%	0.00	0.00%	0.00%	0.00
ARGM-LOC	151	51.66%	51.66%	51.66	59.12%	62.25%	60.65
ARGM-MNR	85	41.94%	15.29%	22.41	46.81%	25.88%	33.33
ARGM-PNC	28	38.46%	17.86%	24.39	53.33%	28.57%	37.21
ARGM-PRD	9	83.33%	55.56%	66.67	83.33%	55.56%	66.67
ARGM-REC	1	0.00%	0.00%	0.00	0.00%	0.00%	0.00
ARGM-TMP	386	55.65%	35.75%	43.53	64.79%	59.59%	62.08
Actor1	12	85.71%	50.00%	63.16	90.91%	83.33%	86.96
Actor2	1	100.00%	100.00%	100.00	100.00%	100.00%	100.00
Agent	2551	91.38%	77.34%	83.78	90.42%	81.81%	85.90
Asset	21	42.42%	66.67%	51.85	48.39%	71.43%	57.69
Attribute	17	60.00%	70.59%	64.86	63.16%	70.59%	66.67
Beneficiary	24	65.00%	54.17%	59.09	66.67%	50.00%	57.14
Cause	48	75.56%	70.83%	73.12	71.11%	66.67%	68.82
Experiencer	132	86.49%	72.73%	79.01	83.33%	68.18%	75.00
Location	12	83.33%	41.67%	55.56	80.00%	33.33%	47.06
Material	7	100.00%	14.29%	25.00	100.00%	28.57%	44.44
Patient	37	76.67%	62.16%	68.66	88.89%	64.86%	75.00
Patient1	20	72.73%	40.00%	51.61	78.57%	55.00%	64.71
Predicate	181	63.75%	56.35%	59.82	64.12%	60.22%	62.11
Product	106	70.79%	59.43%	64.62	69.79%	63.21%	66.34
R-ARGM-LOC	2	0.00%	0.00%	0.00	0.00%	0.00%	0.00
R-ARGM-MNR	2	0.00%	0.00%	0.00	0.00%	0.00%	0.00
R-ARGM-TMP	4	0.00%	0.00%	0.00	0.00%	0.00%	0.00
R-Agent	74	70.15%	63.51%	66.67	65.93%	81.08%	72.73
R-Experiencer	5	100.00%	20.00%	33.33	0.00%	0.00%	0.00
R-Patient	2	0.00%	0.00%	0.00	0.00%	0.00%	0.00
R-Predicate	1	0.00%	0.00%	0.00	0.00%	0.00%	0.00
R-Product	2	0.00%	0.00%	0.00	0.00%	0.00%	0.00
R-Recipient	8	100.00%	87.50%	93.33	100.00%	87.50%	93.33
R-Theme	7	75.00%	42.86%	54.55	42.86%	42.86%	42.86
R-Theme1	7	100.00%	85.71%	92.31	100.00%	85.71%	92.31
R-Theme2	1	50.00%	100.00%	66.67	0.00%	0.00%	0.00
R-Topic	14	66.67%	42.86%	52.17	70.00%	50.00%	58.33
Recipient	48	75.51%	77.08%	76.29	76.00%	79.17%	77.55
Source	25	65.22%	60.00%	62.50	62.50%	60.00%	61.22
Stimulus	21	33.33%	19.05%	24.24	46.67%	33.33%	38.89
Theme	650	79.22%	68.62%	73.54	79.04%	70.77%	74.68
Theme1	69	77.42%	69.57%	73.28	81.36%	69.57%	75.00
Theme2	60	74.55%	68.33%	71.30	76.47%	65.00%	70.27
Topic	1867	84.26%	82.27%	83.25	84.12%	82.59%	83.35

Table 3. Evaluation of the semantic role labeling accuracy on the SemEval2007 - Task 17 test set. Column *#TI* reports the number of instances of each role label in the test set. The results under Column *Split* are relative to the configuration that participated in the official evaluation (one BC and one RM per predicate word); the results under Column *Monolithic* are those of the most accurate configuration that we tried (one BC, one RM).

– training a monolithic BC and an RM for each predicate, i.e. BC is trained with the data of all predicates;

– training a split BC and a monolithic RM, i.e. the latter classifier is trained putting together all different predicates. This means that there is only one

Rank	Team	P	R	$\mathbf{F}_{\beta=1}$
1	UBC-UPC	85.31%	82.08%	83.66
2	RTV	81.58%	70.16%	75.44

Table 4. Results of the two teams that took part in the closed task on English SRL. Our system (labeled *RTV*) ranked second out of two.

Task	Classifier		Precision	Recall	$\mathbf{F}_{\beta=1}$
	bnd	role			
BD	s	-	87.09%	72.96%	79.40
	m	-	87.42%	77.36%	82.09
BD + RC	s	s	81.58%	70.16%	75.44
	s	m	81.72%	70.57%	75.74
	m	s	81.05%	73.64%	77.17
	m	m	81.25%	74.42%	77.69

Table 5. Accuracy comparison between the split (s) and the monolithic (m) boundary (bnd) and role (role) classifiers on the boundary detection (BD) and the complete SRL task (BD + RC) on the test set. Of course, no role classifier is employed for the boundary detection task.

classifier for each role type. For example, the *Agent* role classifier will be unique for all predicates.
– Using both a monolithic BC and a monolithic RM.

The whole training set was used to learn the models and the evaluation was carried out on the test set. Training the monolithic BC and role classifiers took almost one week on the same hardware platform. Table 5 shows the results of this comparison with respect to the boundary detection (BD) and the complete SRL task. The flags "s" and "m" in column 2 and 3 indicate the strategy ("split" or "monolithic") employed to train the BC and the RM, respectively.

For the boundary detection task, the data show that the accuracy of the monolithic approach is much higher than the split one, i. e. 82.09 vs 79.40. While the precision is almost the same, the monolithic BC improves by almost 4.5 percent points, i. e. 77.36% vs 72.96%, the recall over the split configuration. A similar trend can be observed on the complete SRL task. Here the overall accuracy improves by 2.25 F_1 points if an all-monolithic configuration is used instead of an all-split configuration. And also in this case, while there is little or no difference in terms of precision (the precision values measured for the four combinations range from 81.05% to 81.72%), the recall of the RM improves by 3-4 percent points when a monolithic boundary classifier is used, i. e. 73.64% vs 70.16% and 74.42% vs 70.57% for the split and the monolithic RM, respectively. The right side of Table 3 (Column *Monolithic*) details the accuracy of the all-monolithic configuration on the classification of each thematic role label from the SemEval test set.

6 Final Remarks

In this paper we presented a system that employs tree kernels and a basic set of flat features for the classification of thematic roles.

The basic approach that we adopted is meant to be as general and fast as possible. The issue of generality is addressed by training the boundary and role classifiers on a per-predicate basis and by employing tree kernels and structured features in the learning algorithm. The resulting architecture can also be used to learn the classification of roles of non-verbal predicates since the automatic feature selection triggered by the tree kernel compensate for the lack of *ad-hoc*, well established explicit features for some classes of non-verbal predicates, such as adverbial or prepositional ones.

Splitting the learning problem also has the clear advantage of noticeably improving the efficiency of the classifiers, thus reducing training and classification time. On the other hand, this split results in some classifiers having too few training instances and therefore being very inaccurate. This is especially true for the boundary classifiers, which conversely need to be very accurate in order to positively support the following stages of the SRL process. The solution of a monolithic boundary classifier that we previously employed [4] is noticeably more accurate though much less efficient, especially for training.

Indeed, after the SemEval2007 evaluation period was over, we ran another series of experiments comparing the split and the monolithic approaches both for boundary detection and the complete SRL task. The results show that the monolithic approach always outperforms the split one, especially for cases like this in which the training instances of some split may be too few and too sparse to generalize properly. In particular, the monolithic boundary classifier grants a noticeable improvement in accuracy, both for the boundary detection and the complete labeling task. Nevertheless, training and classification time with the split configuration is much lower. This issue should be taken into account when we need to: (a) carry out an extensive experimentation with different feature sets; (b) design time-constrained applications, e. g. on-line services; or (c) annotate very large datasets.

Although it was provided as part of both the training and test data, we chose not to use the verb sense information. This choice is motivated by our intention to depend on as less external resources as possible in order to be able to port our SRL system to other linguistic models and languages, for which such resources may not exist. Still, identifying the predicate sense is a key issue especially for role classification, as the argument structure of a predicate is largely determined by its sense. Indeed, the results of the UBC-UPC system clearly show that semantic information can boost the accuracy of SRL, and that it should be employed when dealing with languages, domains or tasks for which it is available and dependable. In the near feature we plan to use larger structured features, i. e. spanning all the potential arguments of a predicate, to improve the accuracy of our role classifiers. We also plan to reintroduce the joint-model evaluation and the TK-based re-ranking mechanism that we presented in [5], which is currently offline due to changes in our feature extraction software component.

References

1. Carreras, X., Màrquez, L.: Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In: Proceedings of CoNLL-2005. (2005)
2. Kipper, K., Dang, H.T., Palmer, M.: Class-based construction of a verb lexicon. In: Proceedings of AAAI-2000 Seventeenth National Conference on Artificial Intelligence, Austin, TX. (2000)
3. Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J.H., Jurafsky, D.: Support vector learning for semantic argument classification. to appear in Machine Learning Journal (2005)
4. Moschitti, A., Pighin, D., Basili, R.: Tree kernel engineering in semantic role labeling systems. In: Proceedings of the Workshop on Learning Structured Information in Natural Language Applications, EACL 2006, Trento, Italy, European Chapter of the Association for Computational Linguistics (2006) 49–56
5. Moschitti, A., Pighin, D., Basili, R.: Tree kernel engineering for proposition reranking. In: Proceedings of the International Workshop on Mining and Learning with Graphs, ECML/PKDD 2006. (2006)
6. Toutanova, K., Haghighi, A., Manning, C.: Joint learning improves semantic role labeling. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor, Michigan, Association for Computational Linguistics (2005) 589–596
7. Sang, E.T.K., Canisius, S., van den Bosch, A., Bogers, T.: Applying spelling error correction techniques for improving semantic role labelling. In: In Proceedings of the Ninth Conference on Natural Language Learning, CoNLL-2005, Ann Arbor, MI (2005)
8. Punyakanok, V., Koomen, P., Roth, D., Yih, W.t.: Generalized inference with multiple semantic role labeling systems. In: Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), Ann Arbor, Michigan, Association for Computational Linguistics (2005) 181–184
9. Vapnik, V.N.: Statistical Learning Theory. John Wiley and Sons (1998)
10. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: ACL02. (2002)
11. Vishwanathan, S., Smola, A.: Fast kernels on strings and trees. In: Proceedings of Neural Information Processing Systems. (2002)
12. Gildea, D., Jurasfky, D.: Automatic labeling of semantic roles. Computational Linguistic **28**(3) (2002) 496–530
13. Xue, N., Palmer, M.: Calibrating features for semantic role labeling. In: Proceedings of EMNLP 2004, Barcelona, Spain (2004) 88–94
14. Moschitti, A., Pighin, D., Basili, R.: Semantic role labeling via tree kernel joint inference. In: Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X. (2006)
15. Moschitti, A.: A study on convolution kernel for shallow semantic parsing. In: proceedings of ACL-2004, Barcelona, Spain (2004)
16. Joachims, T.: Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. (1999)
17. Zapirain, B., Agirre, E., Márquez, L.: UBC-UPC: Sequential SRL Using Selectional Preferences. An approach with Maximum Entropy Markov Models. In: Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval 2007), Prague, Czech Republic, Association for Computational Linguistics (2007)