Semantic Tree Kernels to classify Predicate Argument Structures

Alessandro Moschitti [1] and **Bonaventura Coppola** [2] and **Daniele Pighin** [3] and **Roberto Basili** [4]

Abstract. Recent work on Semantic Role Labeling (SRL) has shown that syntactic information is critical to detect and extract predicate argument structures. As syntax is expressed by means of structured data, i.e. parse trees, its encoding in learning algorithms is rather complex.

In this paper, we apply tree kernels to encode the whole predicate argument structure in Support Vector Machines (SVMs). We extract from the sentence syntactic parse the subtrees that span potential argument structures of the target predicate and classify them in incorrect or correct structures by means of tree kernel based SVMs. Experiments on the PropBank collection show that the classification accuracy of correct/incorrect structures is remarkably high and helps to improve the accuracy of the SRL task. This is a piece of evidence that tree kernels provide a powerful mechanism to learn the complex relation between syntax and semantics.

1 INTRODUCTION

The design of features for natural language processing tasks is, in general, a critical problem. The inherent complexity of linguistic phenomena, often characterized by structured data, makes difficult to find effective attribute-value representations for the target learning models.

In many cases, the traditional feature selection techniques [8] are not very useful since the critical problem relates to feature generation rather than selection. For example, the design of features for a natural language syntactic parse-tree re-ranking problem [2] cannot be carried out without a deep knowledge about automatic syntactic parsing. The modeling of syntax/semantics-based features should take into account linguistic aspects to detect the interesting context, e.g. ancestor nodes or semantic dependencies [15].

A viable alternative has been proposed in [3], where convolution kernels were used to implicitly define a tree substructure space. The selection of the relevant structural features was left to the voted perceptron learning algorithm. Such successful experimentation shows that tree kernels are very promising for automatic feature engineering, especially when the available knowledge about the phenomenon is limited.

In a similar way, automatic learning tasks that rely on syntactic information may take advantage of a tree kernel approach. One of such tasks is the Semantic Role Labeling (SRL), as defined e.g. in [1] over the PropBank corpus [7]. Most literature work models SRL as the classification of tree nodes of the sentence parse containing the target predicate. Indeed, a node can uniquely determine the set of words that compose an argument (boundaries) and provide, along with the local tree structure, information useful to the classification of the role. Accordingly, most SRL systems split the labeling process into two different steps: Boundary Detection (i.e. determine the text boundaries of predicate arguments) and Role Classification (i.e. labeling such arguments with a semantic role, e.g. Arg0 or Arg1).

Both the above steps require the design and extraction of features from the parse tree. Capturing the interconnected relationships among a predicate and its arguments is a hard task. To decrease such complexity we can design features considering a predicate with only one argument at a time, but this limits our ability to capture the semantics of the whole predicate structure. An alternative approach to engineer syntactic features is the use of tree kernels as the substructures that they generate potentially correspond to relevant syntactic clues.

In this paper we use tree kernels to model classifiers that decide if a predicate argument structure is correct or not. We apply a traditional boundary classifier (TBC) [11] to label all parse tree nodes that are potential arguments, then we classify the syntactic subtrees which span the predicate-argument dependencies, i.e. Predicate Argument Spanning Trees ($PAST$s). Since the design of effective features to encode such information is not simple, tree kernels are a very useful method. To validate our approach we experimented tree kernels with Support Vector Machines for the classification of $PAST$s. The results show that this classification problem can be learned with high accuracy (about 88% of F_1-measure[5]) and the impact on the overall SRL labeling accuracy is also relevant.

The paper is organized as follows: Section 2 introduces the Semantic Role Labeling based on SVMs and the tree kernel spaces; Section 3 formally defines the $PAST$s and the algorithm to classify them; Section 4 shows the comparative results between our approach and the traditional one; Section 5 presents the related work; and finally, Section 6 summarizes the conclusions.

2 SEMANTIC ROLE LABELING

In the last years, several machine learning approaches have been developed for automatic role labeling, e.g. [5, 11]. Their common characteristic is the adoption of attribute-value representations for predicate-argument structures. Accordingly, our basic system is similar to the one proposed in [11] and is hereby described.

We use a boundary detection classifier (for any role type) to derive the words compounding an argument and a multiclassifier to assign the role (e.g. `ARG0` or `ARGM`) described in PropBank [7]). To prepare the training data for both classifiers, we used the following algorithm:

1. Given a sentence from the *training-set*, generate a full syntactic parse tree;

[1] University of Rome "Tor Vergata", moschitti@info.uniroma2.it
[2] ITC-Irst and University of Trento, coppolab@itc.it
[3] University of Rome "Tor Vergata", daniele.pighin@gmail.com
[4] University of Rome "Tor Vergata", basili@info.uniroma2.it

[5] F_1 assigns equal importance to Precision P and Recall R, i.e. $F_1 = \frac{2P \times R}{P+R}$.

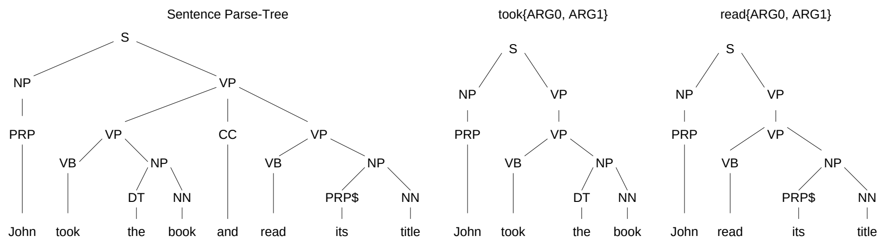

Figure 1. A sentence parse tree with two predicative subtree structures ($PAST$s)

2. Let P and A be respectively the set of predicates and the set of parse-tree nodes;
3. For each pair $<p, a> \in P \times A$:
- 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 the T^+ set (positive examples), otherwise put it in the T^- set (negative examples).

The outputs of the above algorithm are the T^+ and T^- sets. For the subtask of Boundary Detection these can be directly used to train a boundary classifier (e.g. an SVM). Concerning the subtask of Role Classification, the generic binary role labeler for role r (e.g. an SVM) can be trained on the T_r^+, i.e. its positive examples and T_r^-, i.e. its negative examples, where $T^+ = T_r^+ \cup T_r^-$, according to the ONE-vs-ALL scheme. The binary classifiers are then used to build a general role multiclassifier by simply selecting the argument associated with the maximum among the classification scores resulting from the individual binary SVM classifiers.

Regarding the design of features for predicate-argument pairs, we can use the attribute-values defined in [5] or tree structures [10]. Although we focus on the latter approach, a short description of the former is still relevant as they are used by TBC. They include the *Phrase Type*, *Predicate Word*, *Head Word*, *Governing Category*, *Position* and *Voice* features. For example, the *Phrase Type* indicates the syntactic type of the phrase labeled as a predicate argument and the *Parse Tree Path* contains the path in the parse tree between the predicate and the argument phrase, expressed as a sequence of nonterminal labels linked by direction (up or down) symbols, e.g. V $\uparrow$ VP $\downarrow$ NP.

A viable alternative to manual design of syntactic features is the use of tree-kernel functions. These implicitly define a feature space based on all possible tree substructures. Given two trees T_1 and T_2, instead of representing them with the whole fragment space, we can apply the kernel function to evaluate the number of common fragments.

Formally, given a tree fragment space $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, the indicator function $I_i(n)$ is defined, which is equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is $K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively. In turn $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \lambda^{l(f_i)} I_i(n_1) I_i(n_2)$, where $0 \leq \lambda \leq 1$ and $l(f_i)$ is the number of levels of the subtree f_i. Thus $\lambda^{l(f_i)}$ assigns a lower weight to larger fragments. When $\lambda = 1$, Δ is equal to the number of common fragments rooted at nodes n_1 and n_2. As described in [3], Δ can be computed in $O(|N_{T_1}| \times |N_{T_2}|)$.

3 AUTOMATIC CLASSIFICATION OF PREDICATE ARGUMENT STRUCTURES

Most semantic role labeling models rely only on the features extracted from the current candidate argument node. To consider a complete predicate argument structure, the classifier should formulate a hypothesis on the potential parse-tree node subsets which include the argument nodes of the target predicate. Without the boundary information, we should consider all possible tree node subsets, i.e. an exponential number.

To solve such problems we apply a traditional boundary classifier (TBC) to select the set of potential arguments $\mathcal{PA}$. Such a subset can be associated with a subtree which in turn can be classified by means of a tree kernel function. Intuitively, such a function measures to what extent a given candidate subtree is *compatible* with the subtree of a correct predicate argument structure.

3.1 The Predicate Argument Spanning Trees ($PAST$s)

We consider the predicate argument structures annotated in PropBank along with the corresponding TreeBank data as our object space. Given the target predicate p in a sentence parse tree T and a subset $s = \{n_1, .., n_k\}$ of its nodes, N_T, we define as the spanning tree root r the lowest common ancestor of $n_1, .., n_k$. The node set spanning tree p_s is the subtree rooted in r from which the nodes that are neither ancestors nor descendants of any n_i are removed.

Since predicate arguments are associated with tree nodes (i.e. they exactly fit into syntactic constituents), we can define the *Predicate Argument Spanning Tree* ($PAST$) of a predicate argument set, $\{a_1, .., a_n\}$, as the node set spanning tree (NST) over such nodes, i.e. $p_{\{a_1,..,a_n\}}$. A $PAST$ corresponds to the *minimal* subparse tree whose leaves are all and only the words compounding the arguments. For example, Figure 1 shows the parse tree of the sentence `"John took the book and read its title"`. $took_{\{ARG_0,ARG_1\}}$ and $read_{\{ARG_0,ARG_1\}}$ are two $PAST$ structures associated with the two predicates *took* and *read*, respectively. All the other possible NSTs are not valid $PAST$s for these predicates. Note that labeling $p_s, \forall s \subseteq N_T$ with a $PAST$ Classifier is equivalent to solve the boundary detection problem.

The critical points for the application of $PASTs$ are: (1) how to design suitable features for the characterization of $PASTs$. This new structure requires a careful linguistic investigation about its significant properties. (2) How to deal with the exponential number of NSTs.

For the first problem, the use of tree kernels over the $PAST$s can be an alternative to the manual feature design as the learning machine, (e.g. SVMs) can select the most relevant features from a high dimensional feature space. In other words, we can use a tree kernel function to estimate the similarity between two $PAST$s (see Section 2), hence avoiding to define explicit features.

For the second problem there are two main approaches: (1) We can consider the classification confidence provided by TBC [11] and evaluate the m most probable argument node sequences $\{n_1, .., n_k\}$. On the m NSTs derived from such sequences, we can apply a re-ranking approach based on SVMs with tree kernel. (2) We can use only the set of nodes $\mathcal{PA}$ decided by TBC (i.e. those classified as

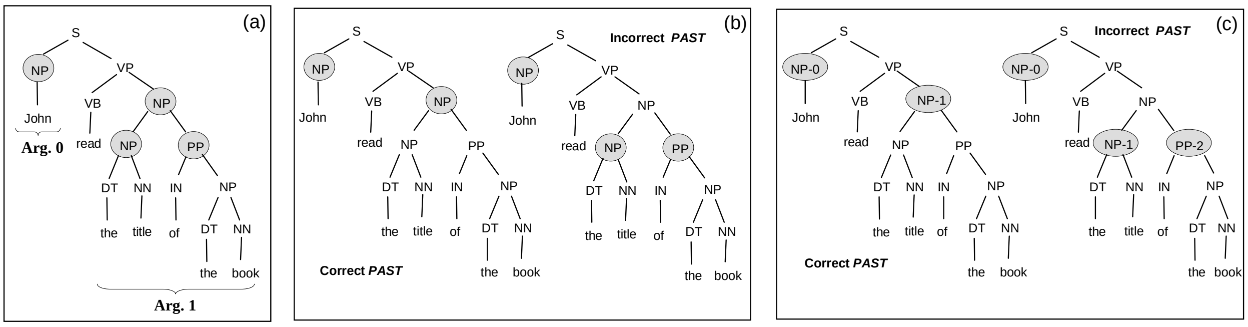

Figure 2. Two-step boundary classification. a) Sentence tree; b) Two candidate $PASTs$; c) Extended $PAST$-Ord labeling

arguments). Thus we need to classify only the set $\mathcal{P}$ of NSTs associated with any subset of $\mathcal{PA}$, i.e. $\mathcal{P} = \{p_s : s \subseteq \mathcal{PA}\}$.

As a re-ranking task would not give an explicit and clear indication of the classifier ability to distinguish between correct and incorrect predicate argument structures, we preferred to apply the second approach. However, also the classification of $\mathcal{P}$ may be computationally problematic, since theoretically there are $|\mathcal{P}| = 2^{|\mathcal{PA}|}$ members.

In order to develop a very efficient procedure, we applied the $PAST$ Classifier only to structures that we know that are incorrect. A simple way to detect them is to look for node pairs $<n_1, n_2> \in \mathcal{PA} \times \mathcal{PA}$ that overlap, i.e. either n_1 is ancestor of n_2 or vice versa. Note that structures that contain overlapping nodes often also contain correct substructures, i.e. subsets of $\mathcal{PA}$ can be associated with correct $PAST$. Assuming the above hypothesis, we create two node sets $\mathcal{PA}_1 = \mathcal{PA} - \{n_1\}$ and $\mathcal{PA}_2 = \mathcal{PA} - \{n_2\}$ and classify them with the $PAST$ Classifier to select the correct set of argument boundaries. This procedure can be generalized to a set of overlapping nodes greater than 2 by selecting a maximal set of non-overlapping nodes. Additionally, as the Precision of TBC is generally high, the number of overlapping nodes is very small. Thus we can explore the whole space.

Figure 2 shows a working example of the multi-stage classifier. In Frame (a), TBC labels as potential arguments (gray color) three overlapping nodes related to `ARG1`. This leads to two possible solutions (Frame (b)) of which only the first is correct. In fact, according to the second one, the propositional phrase *of the book* would be incorrectly attached to the verbal predicate, i.e. in contrast with the parse tree. The $PAST$ Classifier, applied to the two NSTs, is expected to detect this inconsistency and provide the correct output.

3.2 Designing Features with Tree Fragments

The Frame (b) of Figure 2 shows two perfectly identical NSTs. Therefore, it is not possible to discern between them using only their fragments. To solve the problem we can enrich the $NSTs$ by marking their argument nodes with a progressive number, starting from the leftmost argument. For example, in the first NST of Frame (c), we mark as `NP-0` and `NP-1` the first and second argument nodes whereas in the second NST we transform the three argument node labels in `NP-0`, `NP-1` and `PP-2`. We will refer to the resulting structure as a $PAST$-Ord (ordinal number). This simple modification allows the tree kernel to generate different argument structures for the above NSTs. For example, from the first NST in Figure 2.c, the fragments `[NP-1 [NP][PP]]`, `[NP [DT][NN]]` and `[PP [IN][NP]]` are generated. They do not match anymore with the `[NP-0 [NP][PP]]`, `[NP-1 [DT][NN]]` and `[PP-2 [IN][NP]]` fragments generated from the second NST in Figure 2.c.

We also explored another relevant direction in enriching the feature space. It should be noted that the semantic information provided by the role type can remarkably help the detection of correct or incorrect predicate argument structures. Thus, we enrich the argument node label with the role type, e.g. the `NP-0` and `NP-1` of the correct $PAST$ of Figure 2.c becomes `NP-Arg0` and `NP-Arg1` (not shown). We refer to this structure as $PAST$-Arg. Of course, to apply the $PAST$-Arg Classifier, we need a traditional role multiclassifier (TRM) which labels the arguments detected by TBC.

4 THE EXPERIMENTS

The experiments were carried out within the setting defined in the CoNLL-2005 Shared Task [1]. We used the PropBank corpus available at `www.cis.upenn.edu/∼ace`, along with the Penn TreeBank 2 for the gold trees (`www.cis.upenn.edu/∼treebank`) [9], which includes about 53,700 sentences.

Since the experiments over gold parse trees inherently overestimate the accuracy in the semantic role labeling task, e.g. 93% vs. 79% [11], we also adopted Charniak parse trees from the CoNLL 2005 Shared Task data (available at `www.lsi.upc.edu/∼srlconll/`) along with the official performance evaluator.

All the experiments were performed with the SVM-light software [6] available at `svmlight.joachims.org`. For TBC and TRM, we used the linear kernel with a regularization parameter (option `-c`) equal to 1. A cost factor (option `-j`) of 10 was adopted for TBC to have a higher Recall, whereas for TRM, the cost factor was parameterized according to the maximal accuracy of each argument class on the validation set. For the $PAST$ Classifier, we implemented the tree kernel defined [3] inside SVM-light with a λ equal to 0.4 (see [10]).

4.1 Gold Standard Tree Evaluations

In these experiments, we used the sections from 02 to 08 of the TreeBank/PropBank (54,443 argument nodes and 1,343,046 non-argument nodes) to train the traditional boundary classifier (TBC). Then, we applied it to classify the sections from 09 to 21 (125,443 argument nodes vs. 3,010,673 non-argument nodes). We obtained 2,988 NSTs containing at least one overlapping node pair out of the total 65,212 predicate structures (according to the TBC decisions). From the 2,988 overlapping structures, we derived 3,624 positive and 4,461 negative NSTs, that we used to train the $PAST$-Ord Classifier.

The performance was evaluated through the F_1 measure over Section 23, which includes 10,406 argument nodes out of 249,879 parse

	TBC			TBC+RND			TBC+HEU			TBC+$PAST$-Ord		
	P.	R.	F_1	P.	R.	F_1	P.	R.	F_1	P.	R.	F_1
All Struct.	92.2	98.8	95.4	93.6	97.3	95.4	93.0	97.3	95.1	94.4	98.4	96.4
Overl. Struct.	98.3	65.8	78.8	74.0	72.3	73.1	68.1	75.2	71.5	89.6	92.7	91.1

Table 1. Two-step boundary classification performance using the TBC, RND and HEU baselines, and the $PAST$-Ord classifier.

	Section 21								Section 23							
	bnd				bnd+class				bnd				bnd+class			
	PAST Classifier			RND	PAST Classifier			RND	PAST Classifier			RND	PAST Classifier			RND
	-	Ord	Arg		-	Ord	Arg		-	Ord	Arg		-	Ord	Arg	
P.	87.5	88.3	88.3	86.9	85.5	86.3	86.4	85.0	78.6	79.0	79.3	77.8	73.1	73.5	73.4	72.3
R.	87.3	88.1	88.3	87.1	85.7	86.5	86.8	85.6	78.1	78.4	78.7	77.9	73.8	74.1	74.4	73.6
F_1	87.4	88.2	88.3	87.0	85.6	86.4	86.6	85.3	78.3	78.7	79.0	77.9	73.4	73.8	73.9	72.9

Table 2. Semantic Role Labeling performance on automatic trees using $PAST$ classifiers.

tree nodes. After applying the TBC classifier, we detected 235 overlapping $NSTs$, from which we extracted 204 correct $PAST$s and 385 incorrect ones. On such gold standard trees, we measured only the performance of the $PAST$-Arg Classifier which was very high, i.e. 87.1% in Precision and 89.2% in Recall (88.1% of F_1).

Using the $PAST$-Ord Classifier we removed from the TBC outcome the nodes that caused overlaps. To measure the impact on the boundary detection task, we compared it with three different boundary classification baselines:
1. TBC: overlaps are ignored and no decision is taken. This provides an upper bound for the recall as no potential argument is rejected for later labeling. Notice that, in presence of overlapping nodes, the sentence cannot be annotated correctly.
2. RND: one among the non-overlapping structures with maximal number of arguments is randomly selected.
3. HEU (heuristic): one of the NSTs which contains the nodes with the lowest overlapping score is chosen. This score counts the number of overlapping node pairs in the NST. For example, in Figure 2.a we have an NP that overlaps with two nodes NP and PP, thus it is assigned a score of 2.

The third row of Table 1 shows the results of TBC, $TBC + RND$, $TBC + HEU$ and TBC+$PAST$-Ord in the columns 2,3,4 and 5, respectively. We note that: First, the TBC F_1 is slightly higher than the result obtained in [11], i.e. 95.4% vs. 93.8% under the same training/testing conditions (i.e. same PropBank version, same training and testing split and same machine learning algorithm). This is explained by the fact that we did not include continuations and coreferring arguments that are more difficult to detect. Second, both RND and HEU do not improve the TBC result. This can be explained by observing that in the 50% of the cases a correct node is removed. Third, when the $PAST$-Ord Classifier is used to select the correct node, the F_1 increases of 1.49%, i.e. (96.86 vs. 95.37). This is a relevant result as it is difficult to increase the very high baseline given by TBC. Finally, we tested the above classifiers on the overlapping structures only, i.e. we measured the $PAST$-Ord Classifier improvement on all and only the structures that required its application. Such reduced test set contains 642 argument nodes and 15,408 non-argument nodes. The fourth row of Table 1 reports the classifier performance on such task. We note that the $PAST$-Ord Classifier improves the other heuristics of about 20%.

4.2 Automatic Tree Evaluations

In these experiments we used the automatic trees generated by Charniak's parser and the predicate argument annotations defined in the CoNLL 2005 shared task. Again, we trained TBC on sections 02-08 whereas, to achieve a very accurate role classifier, we trained TRM on all sections 02-21. Then, we trained the $PAST$, $PAST$-Ord, and $PAST$-Arg Classifiers on the output of TBC and TRM over sections 09-20 for a total of 183,642 arguments, 30,220 $PAST$s and 28,143 incorrect $PAST$s.

	Section 21			Section 23		
PAST Class.	P.	R.	F_1	P.	R.	F_1
–	69.8	77.9	73.7	62.2	77.1	68.9
Ord	73.7	81.2	77.3	63.7	80.6	71.2
Arg	73.6	84.7	78.7	64.2	82.3	72.1

Table 3. $PAST$, $PAST$-Ord, and $PAST$-Arg performances on sections 21 and 23.

First, to test the TBC, TRM and the $PAST$ classifiers, we used Section 23 (17,429 arguments, 2,159 $PAST$s and 3,461 incorrect $PAST$s) and Section 21 (12,495 arguments, 1,975 $PAST$s and 2,220 incorrect $PAST$s). The performance derived on Section 21 corresponds to an upper bound of our classifiers, i.e. the results using an ideal syntactic parser (Charniak's parser was trained also on Section 21) and an ideal role classifier. They provide the $PAST$ family classifiers with accurate syntactic and semantic information. Table 3 shows Precision, Recall and F_1 measures of the $PAST$ classifiers over the NSTs of sections 21 and 23. Rows 2, 3 and 4 report the performance of $PAST$, $PAST$-Ord, and $PAST$-Arg Classifiers, respectively. Several points should be remarked: (a) the general performance is lower than the one achieved on gold trees with $PAST$-Ord, i.e. 88.1% (see Section 4.1). The impact of parsing accuracy is also confirmed by the gap of about 6% points between sections 21 and 23. (b) The ordinal numbering of arguments (Ord) and the role type information (Arg) provide the tree kernel with more meaningful fragments since they improve the basic model of about 4%. (c) The deeper semantic information generated by the Arg labels provides useful clues to select correct predicate argument structures, since it improves the Ord model on both sections.

Second, we measured the impact of the $PAST$ classifiers on both phases of semantic role labeling. Table 2 reports the results on the sections 21 and 23. For each of them, the Precision, Recall and F_1 of different approaches to the boundary identification (bnd) and to the complete task, i.e. boundary and role classification (bnd+class), is shown. Such approaches are based on different strategies to remove the overlaps, i.e. $PAST$, $PAST$-Ord, $PAST$-Arg and the baseline (RND) which uses a random selection of non-overlapping structures. We needed to remove the overlaps from the baseline in order to apply the CoNLL evaluator.

We note that: (a) for any model, the boundary detection F_1 on Section 21 is about 10 points higher than the F_1 on Section 23 (e.g. 87.0% vs. 77.9% for RND). As expected, the parse tree quality is very important to detect argument boundaries. (b) On the real test (Section 23) the classification introduces labeling errors which decrease the accuracy of about 5% (77.9 vs 72.9 for RND). (c) The Ord and Arg approaches constantly improve the baseline F_1 of about 1%. Such a result does not surprise as it is similar to the one obtained on Gold Trees: the overlapping structures are a small percentage of the test set thus the overall impact cannot be very high.

Third, the comparison with the CoNLL 2005 results [1] can only be carried out with respect to the whole SRL task (bnd+class in table 2) since boundary detection versus role classification is generally not provided in CoNLL 2005. Moreover, our best global result, i.e. 73.9%, was obtained under two severe experimental factors: a) the use of just 1/3 of the available training data, and b) the usage of the linear SVM model for the TBC classifier, which is much faster than the polynomial SVMs but also less accurate. However, we note the promising results of the $PAST$ meta-classifier, which can be used with any of the best figure CoNLL systems.

Finally, kernel outcome suggests that: (a) it is robust to parse tree errors since it preserves the same improvement across trees derived with different accuracy, i.e. the gold trees of Penn TreeBank and the automatic trees of Section 21 and Section 23. (b) It shows a high accuracy for the classification of correct and incorrect predicate argument structures. This last property is quite interesting considering the important findings of a recent paper [13]. The winning strategy to improve semantic role labeling relates to the exploiting of different labeling hypotheses, i.e. several $\mathcal{PA}_i$ sets derived from different parsing alternatives. A joint inference procedure was used to select the most likely set $s \subseteq \cup_i \mathcal{PA}_i$. In our opinion, the $PAST$ Classifiers seem very well suited to select such set.

5 RELATED WORK

Recently, many kernels for natural language applications have been designed. In what follows, we highlight their difference and properties.

The tree kernel used in this article was proposed in [3] for syntactic parsing reranking. It was experimented with the Voted Perceptron and was shown to improve the syntactic parsing. In [4], a feature description language was used to extract structural features from the syntactic shallow parse trees associated with named entities. The experiments on the named entity categorization showed that when the description language selects an adequate set of tree fragments the Voted Perceptron algorithm increases its classification accuracy. The explanation was that the complete tree fragment set contains many irrelevant features and may cause overfitting. In [13], a set of different syntactic parse trees, e.g. Charniak n best parse trees, were used to improve the SRL accuracy. These different sources of syntactic information were used to generate a set of different SRL outputs. A joint inference stage was applied to resolve the inconsistency of the different outputs. This approach may be applied to our tree kernel strategies to design a joint tree kernel model. In [14], it was observed that there are strong dependencies among the labels of the semantic argument nodes of a verb. Thus, to approach the problem as the classification of an overall role sequences, a re-ranking method is applied to the assignments generated by a TRM. This approach is in line with our $PAST$ Classifier that can be used to refine such reranking strategy. In [12], some experiments were conducted on SRL systems trained using different syntactic views. Again, our approach may be used in conjunction with this model to provide a further syntactic view related to the whole predicate argument structure.

Acknowledgments

This research is partially supported by the PrestoSpace EU Project#: FP6-507336.

6 CONCLUSIONS

The feature design for new natural language learning tasks is difficult. We can take advantage of the kernel methods to model our intuitive knowledge about the target linguistic phenomenon. In this paper we have shown that we can exploit the properties of tree kernels to engineer syntactic features for the semantic role labeling task.

The experiments on gold standard trees as well as on automatic trees suggest that (1) the information related to the whole predicate argument structure is important and (2) tree kernels can be used to generate syntactic/semantic features. The remarkable result is that such structures are robust with respect to parse tree errors.

In the future, we would like to use an approach similar to the $PAST$ classifier to select the best predicate argument annotation from those carried out on several parse trees provided by one or more parsing models.

REFERENCES

[1] Xavier Carreras and Lluís Màrquez, 'Introduction to the CoNLL-2005 shared task: Semantic role labeling', in *Proceedings of CoNLL-2005*, (2005).

[2] Michael Collins, 'Discriminative reranking for natural language parsing', in *In Proceedings of ICML 2000*, (2000).

[3] Michael Collins and Nigel Duffy, 'New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron', in *ACL02*, (2002).

[4] Chad Cumby and Dan Roth, 'Kernel methods for relational learning', in *Proceedings of ICML 2003*, Washington, DC, USA, (2003).

[5] Daniel Gildea and Daniel Jurasfky, 'Automatic labeling of semantic roles', *Computational Linguistic*, **28**(3), 496–530, (2002).

[6] T. Joachims, 'Making large-scale SVM learning practical.', in *Advances in Kernel Methods - Support Vector Learning*, eds., B. Schölkopf, C. Burges, and A. Smola, (1999).

[7] Paul Kingsbury and Martha Palmer, 'From Treebank to PropBank', in *Proceedings of LREC'02)*, Las Palmas, Spain, (2002).

[8] Ron Kohavi and Dan Sommerfield, 'Feature subset selection using the wrapper model: Overfitting and dynamic search space topology', in *1st KDD Conference*, (1995).

[9] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, 'Building a large annotated corpus of english: The Penn Treebank', *Computational Linguistics*, **19**, 313–330, (1993).

[10] Alessandro Moschitti, 'A study on convolution kernel for shallow semantic parsing', in *Proceedings of ACL'04*, Barcelona, Spain, (2004).

[11] Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky, 'Support vector learning for semantic argument classification', *Machine Learning Journal*, (2005).

[12] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky, 'Semantic role labeling using different syntactic views', in *Proceedings of ACL'05*, (2005).

[13] V. Punyakanok, D. Roth, and W. Yih, 'The necessity of syntactic parsing for semantic role labeling', in *Proceedings of IJCAI 2005*, (2005).

[14] Kristina Toutanova, Aria Haghighi, and Christopher Manning, 'Joint learning improves semantic role labeling', in *Proceedings of ACL'05*, (2005).

[15] Kristina Toutanova, Penka Markova, and Christopher D. Manning, 'The leaf projection path view of parse trees: Exploring string kernels for hpsg parse selection', in *In Proceedings of EMNLP 2004*, (2004).

[16] Nianwen Xue and Martha Palmer, 'Calibrating features for semantic role labeling', in *Proceedings of EMNLP 2004*, (2004).