Verb subcategorization kernels for automatic semantic labeling

Alessandro Moschitti and **Roberto Basili**
Department of Computer Science
University of Rome ”Tor Vergata”
Rome, Italy
{moschitti,basili}@info.uniroma2.it

Abstract

Recently, many researches in natural language learning have considered the representation of complex linguistic phenomena by means of structural kernels. In particular, tree kernels have been used to represent verbal subcategorization frame (SCF) information for predicate argument classification. As the SCF is a relevant clue to learn the relation between syntax and semantic, the classification algorithm accuracy was remarkable enhanced. In this article, we extend such work by studying the impact of the SCF tree kernel on both PropBank and FrameNet semantic roles. The experiments with Support Vector Machines (SVMs) confirm a strong link between the SCF and the semantics of the verbal predicates as well as the benefit of using kernels in diverse and complex test conditions, e.g. classification of unseen verbs.

1 Introduction

Some theories of verb meaning are based on syntactic properties, e.g. the alternations of verb arguments (Levin, 1993). In turn, Verb Subcategorization Frame (SCF) characterizes different syntactic alternations, thus, it plays a central role in the linking theory between verb semantics and their syntactic structures.

Figure 1 shows the parse tree for the sentence `"John rented a room in Boston"` along with the semantic shallow information embodied by the verbal predicate *to rent* and its three arguments: Arg0, Arg1 and ArgM. The SCF of such verb, i.e. `NP-PP`, provides a synthesis of the predicate argument structure.

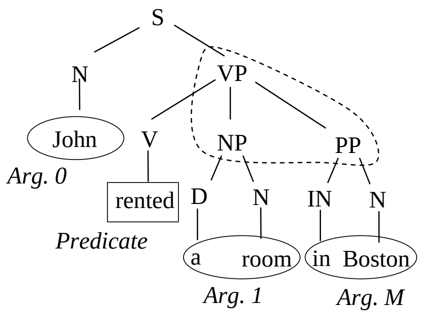

Figure 1: A predicate argument structure in a parse-tree representation.

Currently, the systems which aim to derive semantic shallow information from texts recognize the SCF of a target verb and represent it as a flat feature (e.g. (Xue and Palmer, 2004; Pradhan et al., 2004)) in the learning algorithm. To achieve this goal, a lexicon which describes the SCFs for each verb, is required. Such a resource is difficult to find especially for specific domains, thus, several methods to automatically extract SCF have been proposed (Korhonen, 2003). In (Moschitti, 2004), an alternative to the SCF extraction was proposed, i.e. the SCF kernel (SK). The subcategorization frame of verbs was implicitly represented by means of the syntactic subtrees which include the predicate with its arguments. The similarity between such syntactic structures was evaluated by means of convolution kernels.

Convolution kernels are machine learning approaches which aim to describe structured data in terms of its substructures. The similarity between two structures is carried out by kernel functions which determine the number of common substructures without evaluating the overall substructure space. Thus, if we associate two SCFs with two subtrees, we can measure their similarity with such functions applied to the two trees. This approach determines a more syntactically motivated verb partition than the traditional method based on flat SCF representations (e.g. the `NP-PP` of Figure 1). The subtrees associated with SCF group the verbs which have similar syntactic realizations, in turn, according to Levin's theories, this would suggest that they are semantically related.

A preliminary study on the benefit of such kernels was measured on the classification accuracy of semantic arguments in (Moschitti, 2004). In such work, the improvement on the PropBank arguments (Kingsbury and Palmer, 2002) classification suggests that SK adds information to the prediction of semantic structures. On the contrary, the performance decrease on the FrameNet data classification shows the limit of such approach, i.e. when the syntactic structures are shared among several semantic roles SK seems to be useless.

In this article, we use Support Vector Machines (SVMs) to deeply analyze the role of SK in the automatic predicate argument classification. The major novelty of the article relates to the extensive experimentation carried out on the PropBank (Kingsbury and Palmer, 2002) and FrameNet (Fillmore, 1982) corpora with diverse levels of task complexity, e.g. test instances of unseen predicates (typical of free-text processing). The results show that: (1) once a structural representation of a linguistic object, e.g. SCF, is available we can use convolution kernels to study its connections with another linguistic phenomenon, e.g. the semantic predicate arguments. (2) The tree kernels automatically derive the features (structures) which support also a sort of back-off estimation in case of unseen verbs. (3) The structural features are in general robust in all testing conditions.

The remainder of this article is organized as follows: Section 2 defines the Predicate Argument Extraction problem and the standard solution to solve it. In Section 3 we present our kernels whereas in Section 4 we show comparative results among SVMs using standard features and the proposed kernels. Finally, Section 5 summarizes the conclusions.

2 Parsing of Semantic Roles and Semantic Arguments

There are two main resources that relate to predicate argument structures: PropBank (PB) and FrameNet (FN). PB is a 300,000 word corpus annotated with predicative information on top of the Penn Treebank 2 Wall Street Journal texts. For any given predicate, the expected arguments are labeled sequentially from Arg 0 to Arg 9, ArgA and ArgM. The Figure 1 shows an example of the PB predicate annotation. Predicates in PB are only embodied by verbs whereas most of the times Arg 0 is the *subject*, Arg 1 is the *direct object* and ArgM may indicate *locations*, as in our example.

FrameNet also describes predicate/argument structures but for this purpose it uses richer semantic structures called frames. These latter are schematic representations of situations involving various participants, properties and roles, in which a word may be typically used. Frame elements or semantic roles are arguments of target words that can be verbs or nouns or adjectives. In FrameNet, the argument names are local to the target frames. For example, assuming that *attach* is the target word and *Attaching* is the target frame, a typical sentence annotation is the following.

[$_{Agent}$ `They`] `attach`$_{Tgt}$ [$_{Item}$ `themselves`] [$_{Connector}$ `with their mouthparts`] `and then release a digestive enzyme secretion which eats into the skin.`

Several machine learning approaches for argument identification and classification have been developed, e.g. (Gildea and Jurasfky, 2002; Gildea and Palmer, ; Gildea and Hockenmaier, 2003; Pradhan et al., 2004). Their common characteristic is the adoption of feature spaces that model predicate-argument structures in a flat feature representation. In the next section we present the common parse tree-based approach to this problem.

2.1 Predicate Argument Extraction

Given a sentence in natural language, all the predicates associated with the verbs have to be identified along with their arguments. This problem can be divided into two subtasks: (a) the detection of the target argument boundaries, i.e. all its compounding words, and (b) the classification of the argument type, e.g. *Arg0* or *ArgM* in PropBank or *Agent* and *Goal* in FrameNet.

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 $<p, a> \in \mathcal{P} \times \mathcal{A}$:
   - 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).

For instance, in Figure 1, for each combination of the predicate *rent* with the nodes `N`, `S`, `VP`, `V`, `NP`, `PP`, `D` or `IN` the instances $F_{rent,a}$ are generated. In case the node a exactly covers "Paul", "a room" or "in Boston", it will be a positive instance otherwise it will be a negative one, e.g. $F_{rent,IN}$.

The T^+ and T^- sets can be re-organized as positive $T^+_{arg_i}$ and negative $T^-_{arg_i}$ 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 as it is simple and effective (Pradhan et al., 2004). In the classification phase, given a sentence of the *test-set*, all its $F_{p,a}$ are generated and classified by each individual classifier C_i. As a final decision, we select the argument associated with the maximum value among the scores provided by the individual classifiers.

2.2 Standard feature space

The discovery of relevant features is, as usual, a complex task, nevertheless, there is a common consensus on the basic features that should be adopted. These standard features, firstly proposed in (Gildea and Jurasfky, 2002), refer to a flat information derived from parse trees, i.e. *Phrase Type*, *Predicate Word*, *Head Word*, *Governing Category*, *Position* and *Voice*. For example, the *Phrase Type* indicates the syntactic type of the phrase labeled as a predicate argument, e.g. NP for Arg_1 in Figure 1. The *Parse Tree Path* contains the path in the parse tree between the predicate and the argument phrase, expressed as a sequence of non-terminal labels linked by direction (up or down) symbols, e.g. `V` $\uparrow$ `VP` $\downarrow$ `NP` for Arg_1 in Figure 1. The *Predicate Word* is the surface form of the verbal predicate, e.g. *rent* for all arguments.

In the next section we describe the SVM approach and the basic kernel theory for the predicate argument classification.

2.3 Learning with Support Vector Machines

Given a vector space in $\Re^n$ and a set of positive and negative points, SVMs classify vectors according to a separating hyperplane, $H(\vec{x}) = \vec{w} \times \vec{x} + b = 0$, where $\vec{w} \in \Re^n$ and $b \in \Re$ are learned by applying the *Structural Risk Minimization principle* (Vapnik, 1995).

To apply the SVM algorithm to Predicate Argument Classification, we need a function $\phi : \mathcal{F} \rightarrow \Re^n$ to map our features space $\mathcal{F} = \{f_1, .., f_{|\mathcal{F}|}\}$ and our predicate/argument pair representation, $F_{p,a} = F_z$, into $\Re^n$, such that:

$$F_z \rightarrow \phi(F_z) = (\phi_1(F_z), .., \phi_n(F_z))$$

From the kernel theory we have that:

$$H(\vec{x}) = \Big(\sum_{i=1..l} \alpha_i \vec{x}_i \Big) \cdot \vec{x} + b =$$

$$\sum_{i=1..l} \alpha_i \vec{x}_i \cdot \vec{x} + b = \sum_{i=1..l} \alpha_i \phi(F_i) \cdot \phi(F_z) + b.$$

where, $F_i \ \forall i \in \{1, .., l\}$ are the training instances and the product $K_T(F_i, F_z) = <\phi(F_i) \cdot \phi(F_z)>$ is the kernel function associated with the mapping ϕ.

The simplest mapping that we can apply is $\phi(F_z) = \vec{z} = (z_1, ..., z_n)$ where $z_i = 1$ *if* $f_i \in F_z$ and $z_i = 0$ otherwise, i.e. the characteristic vector of the set F_z with respect to $\mathcal{F}$. If we choose the scalar product as a kernel function we obtain the linear kernel $K_L(F_x, F_z) = \vec{x} \cdot \vec{z}$.

Another function that has shown high accuracy for the predicate argument classification (Pradhan et al., 2004) is the polynomial kernel:

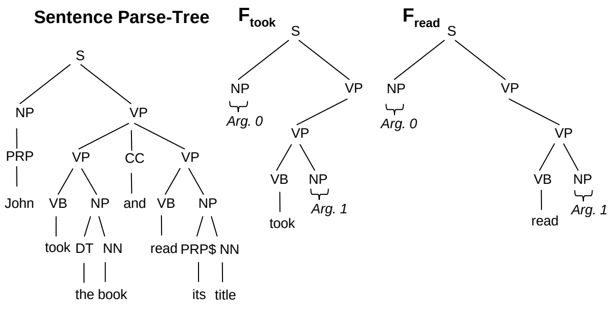

Figure 2: Subcategorization frame structure for two predicate argument structures.

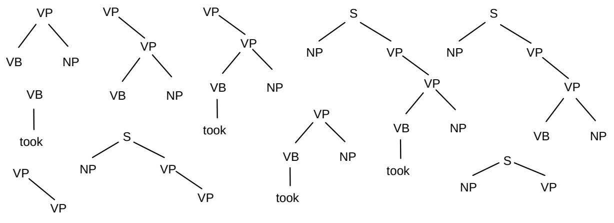

Figure 3: All 10 valid fragments of the SCFS associated with the arguments of F_{took} of Figure 2.

$K_{Poly}(F_x, F_z) = (c + \vec{x} \cdot \vec{z})^d$, where c is a constant and d is the degree of the polynom.

The interesting property is that we do not need to evaluate the ϕ function to compute the above vector; only the $K(\vec{x}, \vec{z})$ values are required. This allows us to define efficient classifiers in a huge (possible infinite) feature set, provided that the kernel is processed in an efficient way. In the next section, we introduce the convolution kernel that we used to represent subcategorization structures.

3 Subcategorization Frame Kernel (SK)

The convolution kernel that we have experimented was devised in (Moschitti, 2004) and is characterized by two aspects: the semantic space of the subcategorization structures and the kernel function that measure their similarities.

3.1 Subcategorization Frame Structure (SCFS)

We consider the predicate argument structures annotated in PropBank or FrameNet as our semantic space. As we assume that semantic structures are correlated to syntactic structures, we used a kernel that selects semantic information according to the syntactic structure of a predicate. The subparse tree which describes the subcategorization frame of the target verbal predicate defines the target Subcategorization Frame Structure (SCFS). For example, Figure 2 shows the parse tree of the sentence `"John took the book and read its title"` together with two SCFS structures, F_{took} and F_{read} associated with the two predicates *took* and *read*, respectively. Note that SCFS includes also the external argument (i.e. the subject) although some linguistic theories do not consider it being part of the SCFs.

Once the semantic representation is defined, we need to design a tree kernel function to estimate the similarity between our objects.

3.2 The tree kernel function

The main idea of tree kernels is to model a $K(T_1, T_2)$ function which computes the number of the common substructures between two trees T_1 and T_2. For example, Figure 3 shows all the fragments of the argument structure F_{took} (see Figure 2) which will be matched against the fragment of another SCFS.

Given the set of fragments $\{f_1, f_2, ..\} = \mathcal{F}$ extracted from all SCFSs of the training set, we define the indicator function $I_i(n)$ which is equal 1 if the target f_i is rooted at node n and 0 otherwise. It follows that:

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \qquad (1)$$

where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. This latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes. 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-terminals 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

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(c^j_{n_1}, c^j_{n_2})) \qquad (2)$$

where $\sigma \in \{0, 1\}$, $nc(n_1)$ is the number of the children of n_1 and c^j_n is the j-th child of the node n.

Note that, as the productions are the same $nc(n_1) = nc(n_2)$.

The above kernel has the drawback of assigning higher weights to larger structures[1]. To overcome this problem we can scale the relative importance of the tree fragments using a parameter λ in the conditions 2 and 3 as follows: $\Delta(n_x, n_z) = \lambda$ and $\Delta(n_x, n_z) = \lambda \prod_{j=1}^{nc(n_x)} (\sigma + \Delta(c_{n_1}^j, c_{n_2}^j))$.

The set of fragments that belongs to SCFs are derived by human annotators according to semantic considerations, thus they generate a semantic subcategorization frame kernel (SK). We also note that SK estimates the similarity between two SCFSs by counting the number of fragments that are in common. For example, in Figure 2, $K_T(\phi(F_{took}), \phi(F_{read}))$ is quite high (i.e. 6 out 10 substructures) as the two verbs have the same syntactic realization.

In other words the fragments encode semantic information which is measured by SK. This provides the argument classifiers with important clues about the possible set of arguments suited for a target verbal predicate. To support this hypothesis the next section presents the experiments on the predicate argument type of FrameNet and ProbBank.

4 The Experiments

A clustering algorithm which uses SK would group together verbs that show a similar syntactic structure. To study the properties of such clusters we experimented SK in combination with the traditional kernel used for the predicate argument classification. As the polynomial kernel with degree 3 was shown to be the most accurate for the argument classification (Pradhan et al., 2004; Moschitti, 2004) we use it to build two kernel combinations:

- $Poly + SK$: $\frac{K_{Poly}}{|K_{Poly}|} + \gamma \frac{K_T}{|K_T|}$, i.e. the sum between the normalized polynomial kernel (see Section 2.3) and the normalized SK[2].

- $Poly \times SK$: $\frac{K_{Poly} \times K_T}{|K_{Poly}| \times |K_T|}$, i.e. the normalized product between the polynomial kernel and SK.

For the experiments we adopted two corpora PropBank (PB) and FrameNet (FN). PB, available at `www.cis.upenn.edu/∼ace`, is used along with the Penn TreeBank 2 (`www.cis.upenn.edu /∼treebank`) (Marcus et al., 1993). This corpus contains about 53,700 sentences and a fixed split between training and testing which has been used in other researches, e.g. (Pradhan et al., 2004; Gildea and Palmer,). In this split, Sections from 02 to 21 are used for training, section 23 for testing and sections 1 and 22 as development set. We considered all 12 arguments from *Arg0* to *Arg9*, *ArgA* and *ArgM* for a total of 123,918 and 7,426 arguments in the training and test sets, respectively. It is worth noting that in the experiments we used the gold standard parsing from the Penn TreeBank, thus our kernel structures are derived with high precision.

The second corpus was obtained by extracting from FrameNet (`www.icsi.berkeley.edu/ ∼framenet/`) all 24,558 sentences from 40 frames of the Senseval 3 (`http://www.senseval.org`) Automatic Labeling of Semantic Role task. We considered 18 of the most frequent roles for a total of 37,948 arguments[3]. Only verbs are selected to be predicates in our evaluations. Moreover, as there is no fixed split between training and testing, we randomly selected 30% of the sentences for testing and 30% for *validation-set*, respectively. Both training and testing sentences were processed using Collins' parser (Collins, 1997) to generate parse-tree automatically. This means that our shallow semantic parser for FrameNet is fully automated.

4.1 The Classification set-up

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).

The classification performance was measured using the F_1 measure[4] for the individual arguments and the accuracy for the final multi-class classifier. This latter choice allows us to compare the results

[1] With a similar aim and to have a similarity score between 0 and 1, we also apply the normalization in the kernel space, i.e. $K'(T_1, T_2) = \frac{K(T_1,T_2)}{\sqrt{K(T_1,T_1) \times K(T_2,T_2)}}$.

[2] To normalize a kernel $K(\vec{x}, \vec{z})$ we can divide it by $\sqrt{K(\vec{x}, \vec{x}) \times K(\vec{z}, \vec{z})}$.

[3] We mapped together roles having the same name

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

with previous literature works, e.g. (Gildea and Jurasfky, 2002; Pradhan et al., 2004; Gildea and Palmer,).

For the evaluation of SVMs, we used the default regularization parameter (e.g., $C = 1$ for normalized kernels) and we tried a few cost-factor values (i.e., $j \in \{1, 2, 3, 5, 7, 10, 100\}$) to adjust the rate between Precision and Recall. We chose the parameters by evaluating the SVMs using the K_{Poly} kernel (degree = 3) over the *validation-set*. Both λ (see Section 3.2) and γ parameters were evaluated in a similar way by maximizing the performance of SVM using *Poly+SK*. We found that the best values were 0.4 and 0.3, respectively.

4.2 Comparative results

To study the impact of the subcategorization frame kernel we experimented the three models $Poly$, $Poly + SK$ and $Poly \times SK$ on different training conditions.

First, we run the above models using all the verbal predicates available in the training and test sets. Tables 1 and 2 report the F_1 measure and the global accuracy for PB and FN, respectively. Column 2 shows the accuracy of $Poly$ (90.5%) which is substantially equal to the accuracy obtained in (Pradhan et al., 2004) on the same training and test sets with the same SVM model. Columns 3 and 4 show that the kernel combinations $Poly + SK$ and $Poly \times SK$ remarkably improve $Poly$ accuracy, i.e. 2.7% (93.2% vs. 90.5%) whereas on FN only $Poly + SK$ produces a small accuracy increase, i.e. 0.7% (86.2% vs. 85.5%).

This outcome is lower since the FN classification requires dealing with a higher variability of its semantic roles. For example, in ProbBank most of the time, the PB *Arg0* and *Arg1* corresponds to the *logical subject* and *logical direct object*, respectively. On the contrary, the FN *Cause* and *Agent* roles are often both associated with the *logical subject* and share similar syntactic realizations, making SCFS less effective to distinguish between them. Moreover, the training data available for FrameNet is smaller than that used for PropBank, thus, the tree kernel may not have enough examples to generalize, correctly.

Second, we carried out other experiments using a subset of the total verbs for training and another disjoint subset for testing. In these conditions, the impact of SK is amplified: on PB, $SK \times Poly$ outperforms *Poly* by 4.8% (86.9% vs. 82.1%), whereas, on FN, SK increases *Poly* of about 2%, i.e. 74.6% vs. 72.8%. These results suggest that (a) when test-set verbs are not observed during training, the classification task is harder, e.g. 82.1% vs. 90.5% on PB and (b) the syntactic structures of the verbs, i.e. the SCFSs, allow the SVMs to better generalize on unseen verbs.

Args	All Verbs			Disjoint Verbs		
	$Poly$	$+SK$	$\times SK$	$Poly$	$+SK$	$\times SK$
Arg0	90.8	94.6	94.7	86.8	90.9	91.1
Arg1	91.1	92.9	94.1	81.7	86.8	88.3
Arg2	80.0	77.4	82.0	49.9	49.5	47.6
Arg3	57.9	56.2	56.4	20.3	22.9	20.6
Arg4	70.5	69.6	71.1	0	0	0
ArgM	95.4	96.1	96.3	90.3	93.4	93.7
Acc.	90.5	92.4	93.2	82.1	86.3	86.9

Table 1: Kernel accuracies on PropBank.

Role	All Verbs			Disjoint Verbs		
	$Poly$	$+SK$	$\times SK$	$Poly$	$+SK$	$\times SK$
agent	91.7	94.4	94.0	82.5	84.8	84.7
cause	57.4	60.6	56.4	29.1	28.1	26.9
degree	77.1	77.2	60.9	40.6	44.6	22.6
depict.	85.8	86.2	85.9	73.6	74.0	71.2
instrum.	67.1	69.1	64.6	13.3	13.0	12.8
manner	80.5	79.7	77.7	74.8	74.3	72.3
Acc.	85.5	86.2	85.0	72.8	74.6	74.2

Table 2: Kernel accuracies on 18 FrameNet semantic roles.

To verify that the kernel representation is superior to the traditional representation we carried out an experiment using a flat feature representation of the SCFs, i.e. we used the syntactic frame feature described (Xue and Palmer, 2004) in place of SK. The result as well as other literature findings, e.g. (Pradhan et al., 2004) show an improvement on PB of about 0.7% only. Evidently flat features cannot derive the same information of a convolution kernel.

Finally, to study how the verb complexity impacts on the usefulness of SK, we carried out additional experiments with different verb sets. One dimension of complexity is the frequency of the verbs in the target corpus. Infrequent verbs are associated with predicate argument structures poorly represented in the training set thus they are more difficult to classify. Another dimension of the verb complexity is the number of different SCFs that they show in different contexts. Intuitively, the higher is the number

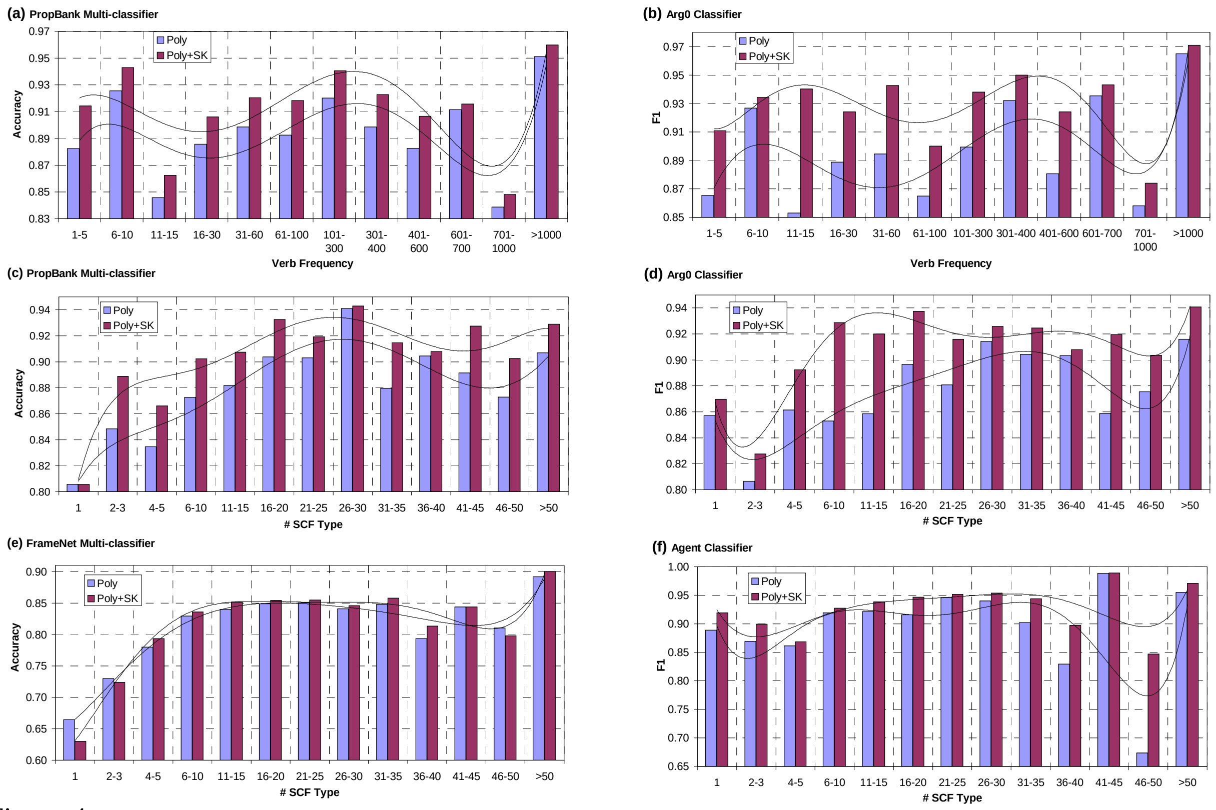

Figure 4: The impact of SCF on the classification accuracy of the semantic arguments and semantic roles according to the verb complexity.

of verb's SCF types the more difficult is the classification of its arguments.

Figure 4.a, reports the accuracy along with the trend line plot of $Poly$ and $SK + Poly$ according to subsets of different verb frequency. For example, the label 1-5 refers to the class of verbal predicates whose frequency ranges from 1 to 5. The associated accuracy is evaluated on the portions of the training and test-sets which contain only the verbs in such class. We note that SK improves $Poly$ for any verb frequency. Such improvement decreases when the frequency becomes very high, i.e. when there are many training instances that can suggest the correct classification. A similar behavior is shown in Figure 4.b where the F_1 measure for Arg0 of PB is reported.

Figures 4.c and 4.d illustrate the accuracy and the F_1 measure for all arguments and Arg0 of PB according to the number of SCF types, respectively. We observe that the Semantic Kernel does not produce any improvement on the verbs which are syntactically expressed by only one type of SCF. As the number of SCF types increases (> 1) $Poly + SK$ outperforms $Poly$ for any verb class, i.e. when the verb is *enough* complex SK always produces useful information independently of the number of the training set instances. On the one hand, a high number of verb instances reduces the complexity of the classification task. On the other hand, as the number of verb type increases the complexity of the task increases as well.

A similar behavior can be noted on the FN data (Figure 4.e) even if the not so strict correlation between syntax and semantics prevents SK to produce high improvements. Figure 4.f shows the impact of SK on the *Agent* role. We note that, the F_1 increases more than the global accuracy (Figure 4.e) as the *Agent* most of the time corresponds to Arg0. This is confirmed by the Table 2 which shows an improvement for the *Agent* of up to 2% when SK is used along with the polynomial kernel.

5 Conclusive Remarks

In this article, we used Support Vector Machines (SVMs) to deeply analyze the role of the subcategorization frame kernel (SK) in the automatic predicate argument classification of PropBank and

FrameNet corpora. To study the SK's verb classification properties we have combined it with the polynomial kernel on standard flat features.

We run the SVMs on diverse levels of task complexity. The results show that: (1) in general SK remarkably improves the classification accuracy. (2) When there are no training instances of the test-set verbs the improvement of SK is almost double. This suggests that tree kernels automatically derive features which support also a sort of back-off estimation in case of unseen verbs. (3) In all complexity conditions the structural features are in general very robust, maintaining a high improvement over the basic accuracy. (4) The semantic role classification in FrameNet is affected with more noisy data as it is based on the output of a statistical parser. As a consequence the improvement is lower. Anyway, the systematic superiority of SK suggests that it is less sensible to parsing errors than other models. This opens promising direction for a more weakly supervised application of the statistical semantic tagging supported by SK.

In summary, the extensive experimentation has shown that the SK provides information robust with respect to the complexity of the task, i.e. verbs with richer syntactic structures and sparse training data.

An important observation on the use of tree kernels has been pointed out in (Cumby and Roth, 2003). Both computational efficiency and classification accuracy can often be superior if we select the most informative tree fragments and carry out the learning in the feature space. Nevertheless, the case studied in this paper is well suited for using kernels as: (1) it is difficult to guess which fragment from an SCF should be retained and which should be discarded, (2) it may be the case that all fragments are useful as SCFs are small structures and all theirs substructures may serve as different back-off levels and (3) small structures do not heavily penalize efficiency.

Future research may be addressed to (a) the use of SK kernel to explicitly generate verb clusters and (b) the use of convolution kernels to study other linguistic phenomena: we can use tree kernels to investigate which syntactic features are suited for an unknown phenomenon.

Acknowledgement

We would like to thank the anonymous reviewers for their competence and commitment showed in the review of this paper.

References

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the ACL'97*,Somerset, New Jersey.

Chad Cumby and Dan Roth. 2003. Kernel methods for relational learning. In *Proceedings of ICML'03*, Washington, DC, USA.

Charles J. Fillmore. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of EMNLP'03*, Sapporo, Japan.

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

Daniel Gildea and Martha Palmer. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL'02*.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Sch$\ddot{o}$lkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC'02*.

Anna Korhonen. 2003. *Subcategorization Acquisition*. Ph.D. thesis, Techical Report UCAM-CL-TR-530. Computer Laboratory, University of Cambridge.

Beth Levin. 1993. *English Verb Classes and Alternations A Preliminary Investigation*. Chicago: University of Chicago Press.

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

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *proceedings ACL'04*, 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 the Machine Learning Journal*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP'04*, Barcelona, Spain, July.