Verb Classification using Distributional Similarity in Syntactic and Semantic Structures

Danilo Croce
University of Tor Vergata
00133 Roma, Italy
`croce@info.uniroma2.it`

Alessandro Moschitti
University of Trento
38123 Povo (TN), Italy
`moschitti@disi.unitn.it`

Roberto Basili
University of Tor Vergata
00133 Roma, Italy
`basili@info.uniroma2.it`

Martha Palmer
University of Colorado at Boulder
Boulder, CO 80302, USA
`mpalmer@colorado.edu`

Abstract

In this paper, we propose innovative representations for automatic classification of verbs according to mainstream linguistic theories, namely VerbNet and FrameNet. First, syntactic and semantic structures capturing essential lexical and syntactic properties of verbs are defined. Then, we design advanced similarity functions between such structures, i.e., semantic tree kernel functions, for exploiting distributional and grammatical information in Support Vector Machines. The extensive empirical analysis on VerbNet class and frame detection shows that our models capture meaningful syntactic/semantic structures, which allows for improving the state-of-the-art.

1 Introduction

Verb classification is a fundamental topic of computational linguistics research given its importance for understanding the role of verbs in conveying semantics of natural language (NL). Additionally, generalization based on verb classification is central to many NL applications, ranging from shallow semantic parsing to semantic search or information extraction. Currently, a lot of interest has been paid to two verb categorization schemes: VerbNet (Schuler, 2005) and FrameNet (Baker et al., 1998), which has also fostered production of many automatic approaches to predicate argument extraction.

Such work has shown that syntax is necessary for helping to predict the roles of verb arguments and consequently their verb sense (Gildea and Jurasfky, 2002; Pradhan et al., 2005; Gildea and Palmer, 2002). However, the definition of models for optimally combining lexical and syntactic constraints is still far for being accomplished. In particular, the exhaustive design and experimentation of lexical and syntactic features for learning verb classification appears to be computationally problematic. For example, the verb **order** can belongs to the two VerbNet classes:

– The class 60.1, i.e., *order someone to do something* as shown in: *The Illinois Supreme Court* ***ordered*** *the commission to audit Commonwealth Edison 's construction expenses and refund any unreasonable expenses .*

– The class 13.5.1: *order or request something* like in: ... *Michelle blabs about it to a sandwich man while* ***ordering*** *lunch over the phone .*

Clearly, the syntactic realization can be used to discern the cases above but it would not be enough to correctly classify the following verb occurrence: .. *ordered the lunch to be delivered* .. in Verb class 13.5.1. For such a case, selectional restrictions are needed. These have also been shown to be useful for semantic role classification (Zapirain et al., 2010). Note that their coding in learning algorithms is rather complex: we need to take into account syntactic structures, which may require an exponential number of syntactic features (i.e., all their possible substructures). Moreover, these have to be enriched with lexical information to trig lexical preference.

In this paper, we tackle the problem above by studying innovative representations for automatic verb classification according to VerbNet and FrameNet. We define syntactic and semantic structures capturing essential lexical and syntactic properties of verbs. Then, we apply similarity between such structures, i.e., kernel functions, which can also exploit distributional lexical semantics, to train automatic classifiers. The basic idea of such functions is to compute the similarity between two verbs in terms of all the possible substructures of their syntactic frames. We define and automatically extract a lexicalized approximation of the latter. Then, we apply kernel functions that jointly model structural and lexical similarity so that syntactic properties are combined with generalized lexemes. The nice property of kernel functions is that they can be used in place of the scalar product of feature vectors to train algorithms such as Support Vector Machines (SVMs). This way SVMs can learn the association between syntactic (sub-) structures whose lexical arguments are generalized and target verb classes, i.e., they can also learn selectional restrictions.

We carried out extensive experiments on verb class and frame detection which showed that our models greatly improve on the state-of-the-art (up to about 13% of relative error reduction). Such results are nicely assessed by manually inspecting the most important substructures used by the classifiers as they largely correlate with syntactic frames defined in VerbNet.

In the rest of the paper, Sec. 2 reports on related work, Sec. 3 and Sec. 4 describe previous and our models for syntactic and semantic similarity, respectively, Sec. 5 illustrates our experiments, Sec. 6 discusses the output of the models in terms of error analysis and important structures and finally Sec. 7 derives the conclusions.

2 Related work

Our target task is verb classification but at the same time our models exploit distributional models as well as structural kernels. The next three subsections report related work in such areas.

Verb Classification. The introductory verb classification example has intuitively shown the complexity of defining a comprehensive feature representation. Hereafter, we report on analysis carried out in previous work.

It has been often observed that verb senses tend to show different selectional constraints in a specific argument position and the above verb *order* is a clear example. In the direct object position of the example sentence for the first sense 60.1 of $order$, we found *commission* in the role PATIENT of the predicate. It clearly satisfies the +ANIMATE/+ORGANIZATION restriction on the PATIENT role. This is not true for the direct object dependency of the alternative sense 13.5.1, which usually expresses the THEME role, with unrestricted type selection. When properly generalized, the direct object information has thus been shown highly predictive about verb sense distinctions.

In (Brown et al., 2011), the so called *dynamic dependency neighborhoods* (DDN), i.e., the set of verbs that are typically collocated with a direct object, are shown to be more helpful than lexical information (e.g., WordNet). The set of typical verbs taking a noun n as a direct object is in fact a strong characterization for semantic similarity, as all the nouns m similar to n tend to collocate with the same verbs. This is true also for other syntactic dependencies, among which the direct object dependency is possibly the strongest cue (as shown for example in (Dligach and Palmer, 2008)).

In order to generalize the above DDN feature, distributional models are ideal, as they are designed to model all the collocations of a given noun, according to large scale corpus analysis. Their ability to capture lexical similarity is well established in WSD tasks (e.g. (Schutze, 1998)), thesauri harvesting (Lin, 1998), semantic role labeling (Croce et al., 2010)) as well as information retrieval (e.g. (Furnas et al., 1988)).

Distributional Models (DMs). These models follow the distributional hypothesis (Firth, 1957) and characterize lexical meanings in terms of *context of use*, (Wittgenstein, 1953). By inducing geometrical notions of vectors and norms through corpus analysis, they provide a topological definition of semantic similarity, i.e., distance in a space. DMs can capture the similarity between words such as *delegation, deputation* or *company* and *commission*. In case of sense 60.1 of the verb $order$, DMs can be used to suggest that the role PATIENT can be inherited by all these words, as suitable *Organisations*.

In supervised language learning, when few examples are available, DMs support cost-effective lexical generalizations, often outperforming knowledge based resources (such as WordNet, as in (Pantel et al., 2007)). Obviously, the choice of the context type determines the type of targeted semantic properties. Wider contexts (e.g., entire documents) are shown to suggest topical relations. Smaller contexts tend to capture more specific semantic aspects, e.g. the syntactic behavior, and better capture paradigmatic relations, such as synonymy. In particular, word space models, as described in (Sahlgren, 2006), define contexts as the words appearing in a n-sized window, centered around a target word. Co-occurrence counts are thus collected in a words-by-words matrix, where each element records the number of times two words co-occur within a single window of word tokens. Moreover, robust weighting schemas are used to smooth counts against too frequent co-occurrence pairs: Pointwise Mutual Information (PMI) scores (Turney and Pantel, 2010) are commonly adopted.

Structural Kernels. Tree and sequence kernels have been successfully used in many NLP applications, e.g., parse reranking and adaptation, (Collins and Duffy, 2002; Shen et al., 2003; Toutanova et al., 2004; Kudo et al., 2005; Titov and Henderson, 2006), chunking and dependency parsing, e.g., (Kudo and Matsumoto, 2003; Daumé III and Marcu, 2004), named entity recognition, (Cumby and Roth, 2003), text categorization, e.g., (Cancedda et al., 2003; Gliozzo et al., 2005), and relation extraction, e.g., (Zelenko et al., 2002; Bunescu and Mooney, 2005; Zhang et al., 2006).

Recently, DMs have been also proposed in integrated syntactic-semantic structures that feed advanced learning functions, such as the semantic tree kernels discussed in (Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b; Mehdad et al., 2010; Croce et al., 2011).

3 Structural Similarity Functions

In this paper we model verb classifiers by exploiting previous technology for kernel methods. In particular, we design new models for verb classification by adopting algorithms for structural similarity, known as Smoothed Partial Tree Kernels (SPTKs) (Croce et al., 2011). We define new innovative structures and similarity functions based on LSA.

The main idea of SPTK is rather simple: (i) measuring the similarity between two trees in terms of the number of shared subtrees; and (ii) such number also includes similar fragments whose lexical nodes are just related (so they can be different). The contribution of (ii) is proportional to the lexical similarity of the tree lexical nodes, where the latter can be evaluated according to distributional models or also lexical resources, e.g., WordNet.

In the following, we define our models based on previous work on LSA and SPTKs.

3.1 LSA as lexical similarity model

Robust representations can be obtained through intelligent dimensionality reduction methods. In LSA the original word-by-context matrix M is decomposed through Singular Value Decomposition (SVD) (Landauer and Dumais, 1997; Golub and Kahan, 1965) into the product of three new matrices: U, S, and V so that S is diagonal and $M = USV^T$. M is then approximated by $M_k = U_k S_k V_k^T$, where only the first k columns of U and V are used, corresponding to the first k greatest singular values. This approximation supplies a way to project a generic term w_i into the k-dimensional space using $W = U_k S_k^{1/2}$, where each row corresponds to the representation vectors $\vec{w_i}$. The original statistical information about M is captured by the new k-dimensional space, which preserves the global structure while removing low-variant dimensions, i.e., distribution noise. Given two words w_1 and w_2, the term similarity function σ is estimated as the cosine similarity between the corresponding projections $\vec{w_1}, \vec{w_2}$ in the LSA space, i.e $\sigma(w_1, w_2) = \frac{\vec{w_1} \cdot \vec{w_2}}{\|\vec{w_1}\| \|\vec{w_2}\|}$. This is known as *Latent Semantic Kernel* (LSK), proposed in (Cristianini et al., 2001), as it defines a positive semi-definite Gram matrix $\boldsymbol{G} = \sigma(w_1, w_2)\ \forall w_1, w_2$ (Shawe-Taylor and Cristianini, 2004). σ is thus a valid kernel and can be combined with other kernels, as discussed in the next session.

3.2 Tree Kernels driven by Semantic Similarity

To our knowledge, two main types of tree kernels exploit lexical similarity: the syntactic semantic tree kernel defined in (Bloehdorn and Moschitti, 2007a) applied to constituency trees and the smoothed partial tree kernels (SPTKs) defined in (Croce et al., 2011), which generalizes the former. We report the definition of the latter as we modified it for our purposes. SPTK computes the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. Its

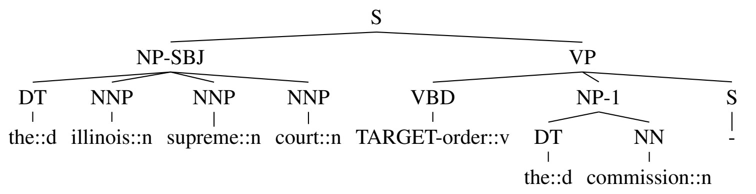

Figure 1: Constituency Tree (CT) representation of verbs.

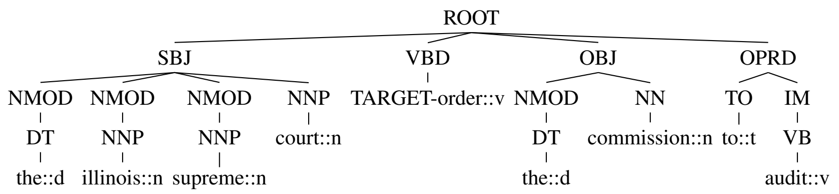

Figure 2: Representation of verbs according to the Grammatical Relation Centered Tree (GRCT)

general equations are reported hereafter:

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

where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2)$ is equal to the number of common fragments rooted in the n_1 and n_2 nodes[1]. The Δ function determines the richness of the kernel space and thus induces different tree kernels, for example, the syntactic tree kernel (STK) (Collins and Duffy, 2002) or the partial tree kernel (PTK) (Moschitti, 2006).

The algorithm for SPTK's Δ is the following: if n_1 and n_2 are leaves then $\Delta_\sigma(n_1, n_2) = \mu\lambda\sigma(n_1, n_2)$; else

$$\Delta_\sigma(n_1, n_2) = \mu\sigma(n_1, n_2) \times \Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_\sigma(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big), \qquad (2)$$

where (1) σ is any similarity between nodes, e.g., between their lexical labels; (2) $\lambda, \mu \in [0, 1]$ are decay factors; (3) $c_{n_1}(h)$ is the h^{th} child of the node n_1; (4) $\vec{I}_1$ and $\vec{I}_2$ are two sequences of indexes, i.e., $\vec{I} = (i_1, i_2, .., l(I))$, with $1 \le i_1 < i_2 < .. < i_{l(I)}$; and (5) $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11} + 1$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21} + 1$. Note that, as shown in (Croce et al., 2011), the average running time of SPTK is sub-quadratic in the number of the tree nodes. In the next section we show how we exploit the class of SPTKs, for verb classification.

[1]To have a similarity score between 0 and 1, a normalization in the kernel space, i.e. $\frac{TK(T_1,T_2)}{\sqrt{TK(T_1,T_1) \times TK(T_2,T_2)}}$ is applied.

4 Verb Classification Models

The design of SPTK-based algorithms for our verb classification requires the modeling of two different aspects: (i) a tree representation for the verbs; and (ii) the lexical similarity suitable for the task. We also modified SPTK to apply different similarity functions to different nodes to introduce flexibility.

4.1 Verb Structural Representation

The implicit feature space generated by structural kernels and the corresponding notion of similarity between verbs obviously depends on the input structures. In the cases of STK, PTK and SPTK different tree representations lead to engineering more or less expressive linguistic feature spaces.

With the aim of capturing syntactic features, we started from two different parsing paradigms: phrase and dependency structures. For example, for representing the first example of the introduction, we can use the constituency tree (CT) in Figure 1, where the target verb node is enriched with the TARGET label. Here, we apply tree pruning to reduce the computational complexity of tree kernels as it is proportional to the number of nodes in the input trees. Accordingly, we only keep the subtree dominated by the target `VP` by pruning from it all the S-nodes along with their subtrees (i.e, all nested sentences are removed). To further improve generalization, we lemmatize lexical nodes and add generalized POS-Tags, i.e., noun (n::), verb (v::), adjective (::a), determiner (::d) and so on, to them. This is useful for constraining similarity to be only contributed by lexical pairs of the same grammatical category.

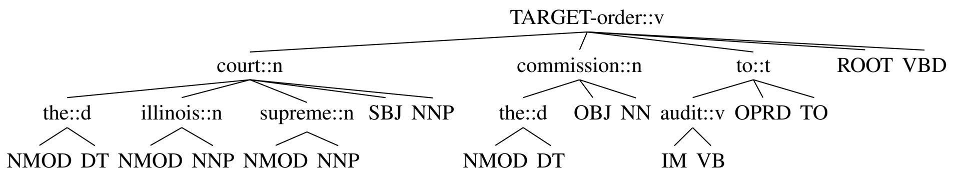

Figure 3: Representation of verbs according to the Lexical Centered Tree (LCT)

To encode dependency structure information in a tree (so that we can use it in tree kernels), we use (i) lexemes as nodes of our tree, (ii) their dependencies as edges between the nodes and (iii) the dependency labels, e.g., grammatical functions (GR), and POS-Tags, again as tree nodes. We designed two different tree types: (i) in the first type, GR are central nodes from which dependencies are drawn and all the other features of the central node, i.e., lexical surface form and its POS-Tag, are added as additional children. An example of the GR Centered Tree (GRCT) is shown in Figure 2, where the POS-Tags and lexemes are children of GR nodes. (ii) The second type of tree uses lexicals as central nodes on which both GR and POS-Tag are added as the rightmost children. For example, Figure 3 shows an example of a Lexical Centered Tree (LCT). For both trees, the pruning strategy only preserves the verb node, its direct ancestors (father and siblings) and its descendants up to two levels (i.e., direct children and grandchildren of the verb node). Note that, our dependency tree can capture the semantic head of the verbal argument along with the main syntactic construct, e.g., *to audit*.

4.2 Generalized node similarity for SPTK

We have defined the new similarity σ_τ to be used in Eq. 2, which makes SPTK more effective as shown by Alg. 1. σ_τ takes two nodes n_1 and n_2 and applies a different similarity for each node type. The latter is derived by τ and can be: GR (i.e., SYNT), POS-Tag (i.e., POS) or a lexical (i.e., LEX) type. In our experiment, we assign 0/1 similarity for SYNT and POS nodes according to string matching. For LEX type, we apply a lexical similarity learned with LSA to only pairs of lexicals associated with the same POS-Tag. It should be noted that the type-based similarity allows for potentially applying a different similarity for each node. Indeed, we also tested an amplification factor, namely, leaf weight (lw), which amplifies the matching values of the leaf nodes.

Algorithm 1 $\sigma_\tau(n_1, n_2, lw)$

```
$\sigma_\tau \leftarrow 0$,
if $\tau(n_1) = \tau(n_2) = \text{SYNT} \wedge label(n_1) = label(n_2)$ then
    $\sigma_\tau \leftarrow 1$
end if
if $\tau(n_1) = \tau(n_2) = \text{POS} \wedge label(n_1) = label(n_2)$ then
    $\sigma_\tau \leftarrow 1$
end if
if $\tau(n_1) = \tau(n_2) = \text{LEX} \wedge pos(n_1) = pos(n_2)$ then
    $\sigma_\tau \leftarrow \sigma_{LEX}(n_1, n_2)$
end if
if leaf$(n_1) \wedge$ leaf$(n_2)$ then
    $\sigma_\tau \leftarrow \sigma_\tau \times lw$
end if
return $\sigma_\tau$
```

5 Experiments

In these experiments, we tested the impact of our different verb representations using different kernels, similarities and parameters. We also compared with simple bag-of-words (BOW) models and the state-of-the-art.

5.1 General experimental setup

We consider two different corpora: one for VerbNet and the other for FrameNet. For the former, we used the same verb classification setting of (Brown et al., 2011). Sentences are drawn from the Semlink corpus (Loper et al., 2007), which consists of the PropBanked Penn Treebank portions of the Wall Street Journal. It contains 113K verb instances, 97K of which are verbs represented in at least one VerbNet class. Semlink includes 495 verbs, whose instances are labeled with more than one class (including one single VerbNet class or none). We used all instances of the corpus for a total of 45,584 instances for 180 verb classes. When instances labeled with the *none* class are not included, the number of examples becomes 23,719.

The second corpus refers to FrameNet frame classification. The training and test data are drawn from the FrameNet 1.5 corpus[2], which consists of 135K sentences annotated according the frame semantics

[2]http://framenet.icsi.berkeley.edu

(Baker et al., 1998). We selected the subset of frames containing more than 100 sentences annotated with a verbal predicate for a total of 62,813 sentences in 187 frames (i.e., very close to the VerbNet datasets). For both the datasets, we used 70% of instances for training and 30% for testing.

Our verb (multi) classifier is designed with the *one-vs-all* (Rifkin and Klautau, 2004) multi-classification schema. This uses a set of binary SVM classifiers, one for each verb class (frame) i. The sentences whose verb is labeled with the class i are positive examples for the classifier i. The sentences whose verbs are compatible with the class i but evoking a different class or labeled with *none* (no current verb class applies) are added as negative examples. In the classification phase the binary classifiers are applied by (i) only considering classes that are compatible with the target verbs; and (ii) selecting the class associated with the maximum positive SVM margin. If all classifiers provide a negative score the example is labeled with *none*.

To learn the binary classifiers of the schema above, we coded our modified SPTK in SVM-Light-TK[3] (Moschitti, 2006). The parameterization of each classifier is carried on a held-out set (30% of the training) and is concerned with the setting of the trade-off parameter (option -c) and the leaf weight (lw) (see Alg. 1), which is used to linearly scale the contribution of the leaf nodes. In contrast, the cost-factor parameter of SVM-Light-TK is set as the ratio between the number of negative and positive examples for attempting to have a balanced Precision/Recall.

Regarding SPTK setting, we used the lexical similarity σ defined in Sec. 3.1. In more detail, LSA was applied to ukWak (Baroni et al., 2009), which is a large scale document collection made up of 2 billion tokens. M is constructed by applying POS tagging to build rows with pairs $\langle$lemma, ::POS$\rangle$ (lemma::POS in brief). The contexts of such items are the columns of M and are short windows of size $[-3, +3]$, centered on the items. This allows for better capturing syntactic properties of words. The most frequent 20,000 items are selected along with their 20k contexts. The entries of M are the point-wise mutual information between them. SVD reduction is then applied to M, with a dimensionality cut of $l = 250$.

For generating the CT, GRCT and LCT structures, we used the constituency trees generated by the Charniak parser (Charniak, 2000) and the dependency structures generated by the LTH syntactic parser (described in (Johansson and Nugues, 2008)).

The classification performance is measured with accuracy (i.e., the percentage of correct classification). We also derive statistical significance of the results by using the model described in (Yeh, 2000) and implemented in (Padó, 2006).

[3](Structural kernels in SVMLight (Joachims, 2000)) available at http://disi.unitn.it/moschitti/Tree-Kernel.htm

	STK		PTK		SPTK	
	lw	Acc.	lw	Acc.	lw	Acc.
CT	-	83.83%	8	**84.57%**	8	84.46%
GRCT	-	84.83%	8	85.15%	8	**85.28%**
LCT	-	77.73%	0.1	86.03%	0.2	**86.72%**
Br. et Al.				84.64%		
BOW				79.08%		
SK				82.08%		

Table 1: VerbNet accuracy with the *none* class

	STK		PTK		SPTK	
	lw	Acc.	lw	Acc.	lw	Acc.
GRCT	-	92.67%	6	92.97%	0.4	93.54%
LCT	-	90.28%	6	92.99%	0.3	93.78%
BOW				91.13%		
SK				91.84%		

Table 2: FrameNet accuracy without the *none* class

5.2 VerbNet and FrameNet Classification Results

To assess the performance of our settings, we also derive a simple baseline based on the bag-of-words (BOW) model. For it, we represent an instance of a verb in a sentence using all words of the sentence (by creating a special feature for the predicate word).

We also used sequence kernels (SK), i.e., PTK applied to a tree composed of a fake root and only one level of sentence words. For efficiency reasons[4], we only consider the 10 words before and after the predicate with subsequence features of length up to 5.

Table 1 reports the accuracy of different models for VerbNet classification. It should be noted that: first, SK produces a much higher accuracy than BOW, i.e., 82.08 vs. 79.08. On one hand, this is

[4]The average running time of the SK is much higher than the one of PTK. When a tree is composed by only one level PTK collapses to SK.

	STK		PTK		SPTK	
	lw	Acc.	*lw*	Acc.	*lw*	Acc.
CT	-	91.14%	8	91.66%	6	91.66%
GRCT	-	91.71%	8	92.38%	4	92.33%
LCT	-	89.20%	0.2	92.54%	0.1	92.55%
BOW				88.16%		
SK				89.86%		

Table 3: VerbNet accuracy without the *none* class

generally in contrast with standard text categorization tasks, for which n-gram models show accuracy comparable to the simpler BOW. On the other hand, it simply confirms that verb classification requires the dependency information between words (i.e., at least the sequential structure information provided by SK).

Second, SK is 2.56 percent points below the state-of-the-art achieved in (Brown et al., 2011) (BR), i.e, 82.08 vs. 84.64. In contrast, STK applied to our representation (CT, GRCT and LCT) produces comparable accuracy, e.g., 84.83, confirming that syntactic representation is needed to reach the state-of-the-art.

Third, PTK, which produces more general structures, improves over BR by almost 1.5 (statistically significant result) when using our dependency structures GRCT and LCT. CT does not produce the same improvement since it does not allow PTK to directly compare the lexical structure (lexemes are all leaf nodes in CT and to connect some pairs of them very large trees are needed).

Finally, the best model of SPTK (i.e, using LCT) improves over the best PTK (i.e., using LCT) by almost 1 point (statistically significant result): this difference is only given by lexical similarity. SPTK improves on the state-of-the-art by about 2.08 absolute percent points, which, given the high accuracy of the baseline, corresponds to 13.5% of relative error reduction.

We carried out similar experiments for frame classification. One interesting difference is that SK improves BOW by only 0.70, i.e., 4 times less than in the VerbNet setting. This suggests that word order around the predicate is more important for deriving the VerbNet class than the FrameNet frame. Additionally, LCT or GRCT seems to be invariant for both PTK and SPTK whereas the lexical similarity still produces a relevant improvement on PTK, i.e., 13% of relative error reduction, for an absolute accuracy of 93.78%. The latter improves over the state-of-the-art, i.e., 92.63% derived in (Giuglea and Moschitti, 2006), by using STK on CT on 133 frames.

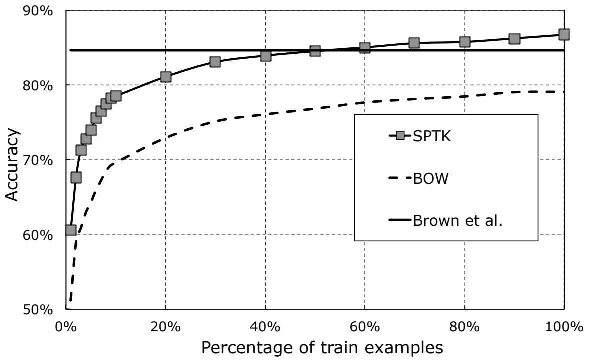

Figure 4: Learning curves: VerbNet accuracy with the *none* Class

We also carried out experiments to understand the role of the *none* class. Table 3 reports on the VerbNet classification without its instances. This is of course an unrealistic setting as it would assume that the current VerbNet release already includes all senses for English verbs. In the table, we note that the overall accuracy highly increases and the difference between models reduces. The similarities play no role anymore. This may suggest that SPTK can help in complex settings, where verb class characterization is more difficult. Another important role of SPTK models is their ability to generalize. To test this aspect, Figure 4 illustrates the learning curves of SPTK with respect to BOW and the accuracy achieved by BR (with a constant line). It is impressive to note that with only 40% of the data SPTK can reach the state-of-the-art.

6 Model Analysis and Discussion

We carried out analysis of system errors and its induced features. These can be examined by applying the reverse engineering tool[5] proposed in (Pighin and Moschitti, 2010; Pighin and Moschitti, 2009a; Pighin and Moschitti, 2009b), which extracts the most important features for the classification model. Many mistakes are related to false positives and negatives of the *none* class (about 72% of the errors). This class also causes data imbalance. Most errors are also due to lack of lexical information available to the SPTK kernel: (i) in 30% of the errors, the argument heads were proper nouns for which the lexical generalization provided by the DMs was not

[5]http://danielepighin.net/cms/software/flink

VerbNet class 13.5.1
(IM(VB(target))(OBJ))
(VC(VB(target))(OBJ))
(VC(VBG(target))(OBJ))
(OPRD(TO)(IM(VB(target))(OBJ)))
(PMOD(VBG(target))(OBJ))
(VB(target))
(VC(VBN(target)))
(PRP(TO)(IM(VB(target))(OBJ)))
(IM(VB(target))(OBJ)(ADV(IN)(PMOD)))
(OPRD(TO)(IM(VB(target))(OBJ)(ADV(IN)(PMOD))))

VerbNet class 60
(VC(VB(target))(OBJ))
(NMOD(VBG(target))(OPRD))
(VC(VBN(target))(OPRD))
(NMOD(VBN(target))(OPRD))
(PMOD(VBG(target))(OBJ))
(ROOT(SBJ)(VBD(target))(OBJ)(P(,)))
(VC(VB(target))(OPRD))
(ROOT(SBJ)(VBZ(target))(OBJ)(P(,)))
(NMOD(SBJ(WDT))(VBZ(target))(OPRD))
(NMOD(SBJ)(VBZ(target))(OPRD(SBJ)(TO)(IM)))

Table 4: GRCT fragments

available; and (ii) in 76% of the errors only 2 or less argument heads are included in the extracted tree, therefore tree kernels cannot exploit enough lexical information to disambiguate verb senses. Additionally, ambiguity characterizes errors where the system is linguistically consistent but the learned selectional preferences are not sufficient to separate verb senses. These errors are mainly due to the lack of contextual information. While error analysis suggests that further improvement is possible (e.g. by exploiting proper nouns), the type of generalizations currently achieved by SPTK are rather effective. Table 4 and 5 report the tree structures characterizing the most informative training examples of the two senses of the verb *order*, i.e. the VerbNet classes 13.5.1 (*make a request for something*) and 60 (*give instructions to or direct somebody to do something with authority*).

In line with the method discussed in (Pighin and Moschitti, 2009b), these fragments are extracted as they appear in most of the support vectors selected during SVM training. As easily seen, the two classes are captured by rather different patterns. The typical accusative form with an explicit direct object emerges as characterizing the sense 13.5.1, denoting the THEME role. All fragments of the sense 60 emphasize instead the sentential complement of the verb that in fact expresses the standard PROPOSITION role in VerbNet. Notice that tree fragments correspond to syntactic patterns. The a posteriori

VerbNet class 13.5.1
(VP(VB(target))(NP))
(VP(VBG(target))(NP))
(VP(VBD(target))(NP))
(VP(TO)(VP(VB(target))(NP)))
(S(NP-SBJ)(VP(VBP(target))(NP)))
VerbNet class 60
(VBN(target))
(VP(VBD(target))(S))
(VP(VBZ(target))(S))
(VBP(target))
(VP(VBD(target))(NP-1)(S(NP-SBJ)(VP)))

Table 5: CT fragments

analysis of the learned models (i.e. the underlying support vectors) confirm very interesting grammatical generalizations, i.e. the capability of tree kernels to implicitly trigger useful linguistic inductions for complex semantic tasks. When SPTK are adopted, verb arguments can be lexically generalized into word classes, i.e., clusters of argument heads (e.g. *commission* vs. *delegation*, or *gift* vs. *present*). Automatic generation of such classes is an interesting direction for future research.

7 Conclusion

We have proposed new approaches to characterize verb classes in learning algorithms. The key idea is the use of structural representation of verbs based on syntactic dependencies and the use of structural kernels to measure similarity between such representations. The advantage of kernel methods is that they can be directly used in some learning algorithms, e.g., SVMs, to train verb classifiers. Very interestingly, we can encode distributional lexical similarity in the similarity function acting over syntactic structures and this allows for generalizing selection restrictions through a *sort of* (supervised) syntactic and semantic co-clustering.

The verb classification results show a large improvement over the state-of-the-art for both VerbNet and FrameNet, with a relative error reduction of about 13.5% and 16.0%, respectively. In the future, we plan to exploit the models learned from FrameNet and VerbNet to carry out automatic mapping of verbs from one theory to the other.

Acknowledgements This research is partially supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant numbers 247758 (ETERNALS), 288024 (LIMOSINE) and 231126 LIVINGKNOWLEDGE. Many thanks to the reviewers for their valuable suggestions.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *LRE*, 43(3):209–226.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In Gianni Amati, Claudio Carpineto, and Gianni Romano, editors, *Proceedings of ECIR*, volume 4425 of *Lecture Notes in Computer Science*, pages 307–318. Springer, APR.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *CIKM'07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 861–864, New York, NY, USA. ACM.

Susan Windisch Brown, Dmitriy Dligach, and Martha Palmer. 2011. Verbnet class assignment as a wsd task. In *Proceedings of the Ninth International Conference on Computational Semantics*, IWCS '11, pages 85–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT and EMNLP*, pages 724–731, Vancouver, British Columbia, Canada, October.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL'02*.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. 2001. Latent semantic kernels. In Carla Brodley and Andrea Danyluk, editors, *Proceedings of ICML-01, 18th International Conference on Machine Learning*, pages 66–73, Williams College, US. Morgan Kaufmann Publishers, San Francisco, US.

Danilo Croce, Cristina Giannone, Paolo Annesi, and Roberto Basili. 2010. Towards open-domain semantic role labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 237–246, Uppsala, Sweden, July. Association for Computational Linguistics.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured Lexical Similarity via Convolution Kernels on Dependency Trees. In *Proceedings of EMNLP 2011*.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and SVM reranking. In *Proceedings of EMNLP'04*.

Dmitriy Dligach and Martha Palmer. 2008. Novel semantic features for verb sense disambiguation. In *ACL (Short Papers)*, pages 29–32. The Association for Computer Linguistics.

J. Firth. 1957. A synopsis of linguistic theory 1930-1955. In *Studies in Linguistic Analysis*. Philological Society, Oxford. reprinted in Palmer, F. (ed. 1968) Selected Papers of J. R. Firth, Longman, Harlow.

G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. 1988. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proc. of SIGIR '88*, New York, USA.

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

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02), Philadelphia, PA*.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of ACL*, pages 929–936, Sydney, Australia, July.

Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of ACL'05*, pages 403–410.

G. Golub and W. Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*.

T. Joachims. 2000. Estimating the generalization performance of a SVM efficiently. In *Proceedings of ICML'00*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of CoNLL 2008*, pages 183–187.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Tom Landauer and Sue Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104.

Dekang Lin. 1998. Automatic retrieval and clustering of similar word. In *Proceedings of COLING-ACL*, Montreal, Canada.

Edward Loper, Szu ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between propbank and verbnet. In *In Proceedings of the 7th International Workshop on Computational Linguistics*.

Yashar Mehdad, Alessandro Moschitti, and Fabio Massimo Zanzotto. 2010. Syntactic/semantic structures for textual entailment recognition. In *HLT-NAACL*, pages 1020–1028.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of ECML'06*, pages 318–329.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. Isp: Learning inferential selectional preferences. In *Proceedings of HLT/NAACL 2007*.

Daniele Pighin and Alessandro Moschitti. 2009a. Efficient linearization of tree kernel functions. In *Proceedings of CoNLL'09*.

Daniele Pighin and Alessandro Moschitti. 2009b. Reverse engineering of tree kernel feature spaces. In *Proceedings of EMNLP*, pages 111–120, Singapore, August. Association for Computational Linguistics.

Daniele Pighin and Alessandro Moschitti. 2010. On reverse feature engineering of syntactic tree kernels. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 223–233, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Ryan Rifkin and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylyania.

Hinrich Schutze. 1998. Automatic word sense discrimination. *Journal of Computational Linguistics*, 24:97–123.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Libin Shen, Anoop Sarkar, and Aravind k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Empirical Methods for Natural Language Processing (EMNLP)*, pages 89–96, Sapporo, Japan.

Ivan Titov and James Henderson. 2006. Porting statistical parsers with data-defined kernels. In *Proceedings of CoNLL-X*.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Ludwig Wittgenstein. 1953. *Philosophical Investigations*. Blackwells, Oxford.

Alexander S. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING*, pages 947–953.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez, and Mihai Surdeanu. 2010. Improving semantic role classification with selectional preferences. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 373–376, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of EMNLP-ACL*, pages 181–201.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In *Proceedings of NAACL*.