Structured Lexical Similarity via Convolution Kernels on Dependency Trees

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

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

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

Abstract

A central topic in natural language processing is the design of lexical and syntactic features suitable for the target application. In this paper, we study convolution dependency tree kernels for automatic engineering of syntactic and semantic patterns exploiting lexical similarities. We define efficient and powerful kernels for measuring the similarity between dependency structures, whose surface forms of the lexical nodes are in part or completely different. The experiments with such kernels for question classification show an unprecedented results, e.g. 41% of error reduction of the former state-of-the-art. Additionally, semantic role classification confirms the benefit of semantic smoothing for dependency kernels.

1 Introduction

A central topic in Natural Language Processing is the design of lexical and syntactic features suitable for the target application. The selection of effective patterns composed of syntactic dependencies and lexical constraints is typically a complex task.

Additionally, the availability of training data is usually scarce. This requires the development of generalized features or the definition of semantic similarities between them, e.g. as proposed in (Resnik, 1995; Jiang and Conrath, 1997; Schutze, 1998; Pedersen et al., 2004a; Bloehdorn and Moschitti, 2007b; Davis et al., 2007) or in semi-supervised settings, e.g. (Chapelle et al., 2006). A semantic similarity can be defined at structural level over a graph, e.g. (Freeman, 1977; Bunke and Shearer, 1998; Brandes, 2001; Zhao et al., 2009), as well as combining structural and lexical similarity over semantic networks, e.g. (Cowie et al., 1992; Wu and Palmer, 1994; Resnik, 1995; Jiang and Conrath, 1997; Schutze, 1998; Leacock and Chodorow, 1998; Pedersen et al., 2004a; Budanitsky and Hirst, 2006). More recent research also focuses on mechanisms to define if two structures, e.g. graphs, are enough similar, as explored in (Mihalcea, 2005; Zhao et al., 2009; Fürstenau and Lapata, 2009; Navigli and Lapata, 2010).

On one hand, previous work shows that there is a substantial lack of automatic methods for engineering lexical/syntactic features (or more in general syntactic/semantic similarity). On the other hand, automatic feature engineering of syntactic or shallow semantic structures has been carried out by means of structural kernels, e.g. (Collins and Duffy, 2002; Kudo and Matsumoto, 2003; Cumby and Roth, 2003; Cancedda et al., 2003; Daumé III and Marcu, 2004; Toutanova et al., 2004; Shen et al., 2003; Gliozzo et al., 2005; Kudo et al., 2005; Titov and Henderson, 2006; Zelenko et al., 2002; Bunescu and Mooney, 2005; Zhang et al., 2006). The main idea of structural kernels is to generate structures that in turn represent syntactic or shallow semantic features. Most notably, the work in (Bloehdorn and Moschitti, 2007b) encodes lexical similarity in such kernels. This is essentially the syntactic tree kernel (STK) proposed in (Collins and Duffy, 2002) in which syntactic fragments from constituency trees can be matched even if they only differ in the leaf nodes (i.e. they have different surface forms). This implies matching scores lower than 1, depending on the semantic similarity of the corresponding leaves in the syntactic fragments.

Although this kernel achieves state-of-the-art performance in NLP tasks, such as Question Classification (Bloehdorn and Moschitti, 2007b) and Textual Entailment (Mehdad et al., 2010), it offers clearly possibility of improvement: (i) better possibility to exploit semantic smoothing since, e.g., trivially STK only matches the syntactic structure *apple/orange* when comparing *the big beautiful apple* to *a nice large orange*; and (ii) STK cannot be effectively applied to dependency structures, e.g. see experiments and motivation in (Moschitti, 2006a). Additionally, to our knowledge, there is no previous study that clearly describes how dependency structures should be converted in trees to be fully and effectively exploitable by convolution kernels. Indeed, although the work in (Culotta and Sorensen, 2004) defines a dependency tree also using node similarity, it is not a convolution kernel: this results in a much poorer feature space.

In this paper, we propose a study of convolution kernels for dependency structures aiming at jointly modeling syntactic and lexical semantic similarity. More precisely, we define several dependency trees exploitable by the Partial Tree Kernel (PTK) (Moschitti, 2006a) and compared them with STK over constituency trees. Most importantly, we define an innovative and efficient class of kernels, i.e. the Smoothed Partial Tree Kernels (SPTKs), which can measure the similarity of structural similar trees whose nodes are associated with different but related lexicals. Given the convolution nature of such kernels any possible node path of lexicals provide a contribution smoothed by the similarity accounted by its nodes.

The extensive experimentation on two datasets of question classification (QC) and semantic role labeling (SRL), shows that: (i) PTK applied to our dependency trees outperforms STK, demonstrating that dependency parsers are fully exploitable for feature engineering based on structural kernels; (ii) SPTK outperforms any previous kernels achieving an unprecedented result of 41% of error reduction with respect to the former state-of-the-art on QC; and (iii) the experiments on SRL confirm that the approach can be applied to different tasks without any tuning and again achieving state-of-the-art accuracy.

In the reminder of this paper, Section 2 provides the background for structural and lexical similarity kernels. Section 3 introduces SPTK. Section 4 provides our representation models for dependency trees. Section 5 presents the experimental evaluation for QC and SRL. Section 6 derives the conclusions.

2 Kernel Background

In kernel-based machines, both learning and classification algorithms only depend on the inner product between instances. This in several cases can be efficiently and implicitly computed by kernel functions by exploiting the following dual formulation: $\sum_{i=1..l} y_i \alpha_i \phi(o_i)\phi(o) + b = 0$, where o_i and o are two objects, ϕ is a mapping from the objects to feature vectors $\vec{x_i}$ and $\phi(o_i)\phi(o) = K(o_i, o)$ is a kernel function implicitly defining such mapping. In case of structural kernels, K determines the shape of the substructures describing the objects above. The most general kind of kernels used in NLP are string kernels, e.g. (Shawe-Taylor and Cristianini, 2004), the Syntactic Tree Kernels (Collins and Duffy, 2002) and the Partial Tree Kernels (Moschitti, 2006a).

2.1 String Kernels

The String Kernels (SK) that we consider count the number of subsequences shared by two strings of symbols, s_1 and s_2. Some symbols during the matching process can be skipped. This modifies the weight associated with the target substrings as shown by the following SK equation:

$$SK(s_1, s_2) = \sum_{u \in \Sigma^*} \phi_u(s_1) \cdot \phi_u(s_2) =$$
$$\sum_{u \in \Sigma^*} \sum_{\vec{I_1}: u = s_1[\vec{I_1}]} \sum_{\vec{I_2}: u = s_2[\vec{I_2}]} \lambda^{d(\vec{I_1}) + d(\vec{I_2})}$$

where, $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$ is the set of all strings, $\vec{I_1}$ and $\vec{I_2}$ are two sequences of indexes $\vec{I} = (i_1, ..., i_{|u|})$, with $1 \leq i_1 < ... < i_{|u|} \leq |s|$, such that $u = s_{i_1}..s_{i_{|u|}}$, $d(\vec{I}) = i_{|u|} - i_1 + 1$ (distance between the first and last character) and $\lambda \in [0, 1]$ is a decay factor.

It is worth noting that: (a) longer subsequences receive lower weights; (b) some characters can be omitted, i.e. gaps; (c) gaps determine a weight since the exponent of λ is the number of characters and gaps between the first and last character; and (c) the complexity of the SK computation is $O(mnp)$ (Shawe-Taylor and Cristianini, 2004), where m and n are the lengths of the two strings, respectively and p is the length of the largest subsequence we want to consider.

2.2 Tree Kernels

Convolution Tree Kernels compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. For this purpose, let the set $\mathcal{F} = \{f_1, f_2, \dots, f_{|\mathcal{F}|}\}$ be a tree fragment space and $\chi_i(n)$ be an indicator function, 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 $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \chi_i(n_1)\chi_i(n_2)$. The latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes. The Δ function determines the richness of the kernel space and thus different tree kernels. Hereafter, we consider the equation to evaluate STK and PTK [1].

2.2.1 Syntactic Tree Kernels (STK)

To compute STK is enough to compute $\Delta_{STK}(n_1, n_2)$ as follows (recalling that since it is a syntactic tree kernels, each node can be associated with a production rule): (i) if the productions at n_1 and n_2 are different then $\Delta_{STK}(n_1, n_2) = 0$; (ii) if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children then $\Delta_{STK}(n_1, n_2) = \lambda$; and (iii) if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then $\Delta_{STK}(n_1, n_2) = \lambda \prod_{j=1}^{l(n_1)} (1 + \Delta_{STK}(c_{n_1}^j, c_{n_2}^j))$, where $l(n_1)$ is the number of children of n_1 and c_n^j is the j-th child of the node n. Note that, since the productions are the same, $l(n_1) = l(n_2)$ and the computational complexity of STK is $O(|N_{T_1}||N_{T_2}|)$ but the average running time tends to be linear, i.e. $O(|N_{T_1}| + |N_{T_2}|)$, for natural language syntactic trees (Moschitti, 2006a).

2.2.2 The Partial Tree Kernel (PTK)

The computation of PTK is carried out by the following Δ_{PTK} function: if the labels of n_1 and n_2 are different then $\Delta_{PTK}(n_1, n_2) = 0$; else $\Delta_{PTK}(n_1, n_2) =$

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

[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.

where $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11} + 1$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21} + 1$. This way, we penalize both larger trees and child subsequences with gaps. PTK is more general than the STK as if we only consider the contribution of shared subsequences containing all children of nodes, we implement the STK kernel. The computational complexity of PTK is $O(p\rho^2|N_{T_1}||N_{T_2}|)$ (Moschitti, 2006a), where p is the largest subsequence of children that we want consider and ρ is the maximal outdegree observed in the two trees. However the average running time again tends to be linear for natural language syntactic trees (Moschitti, 2006a).

2.3 Lexical Semantic Kernel

Given two text fragments d_1 and $d_2 \in D$ (the text fragment set), a general lexical kernel (Basili et al., 2005) defines their similarity as:

$$K(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} (\omega_1 \omega_2) \times \sigma(w_1, w_2) \qquad (1)$$

where ω_1 and ω_2 are the weights of the words (features) w_1 and w_2 in the documents d_1 and d_2, respectively, and σ is a term similarity function, e.g. (Pedersen et al., 2004b; Sahlgren, 2006; Corley and Mihalcea, 2005; Mihalcea et al., 2005). Technically, any σ can be used, provided that the resulting Gram matrix, $\boldsymbol{G} = K(d_1, d_2)\ \forall d_1, d_2 \in D$ is positive semi-definite (Shawe-Taylor and Cristianini, 2004) (D is typically the training text set).

We determine the term similarity function through distributional analysis (Pado and Lapata, 2007), according to the idea that the meaning of a word can be described by the set of textual contexts in which it appears (*Distributional Hypothesis*, (Harris, 1964)). The contexts are words appearing in a n-window with target words: such a space models a generic notion of semantic relatedness, i.e. two words close in the space are likely to be either in paradigmatic or syntagmatic relation as in (Sahlgren, 2006). The original word-by-word context matrix M is decomposed through Singular Value Decomposition (SVD) (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 approximated by $M_l = U_l S_l V_l^T$ in which only the first l columns of U and V are used, and only the first l greatest singular values are considered. This approximation supplies a way to project a generic term w_i into the l-

dimensional space using $W = U_l S_l^{1/2}$, where each row corresponds to the representation vectors $\vec{w_i}$. Therefore, given two words w_1 and w_2, the term similarity function σ is estimated as the cosine similarity between the corresponding projections $\vec{w_1}, \vec{w_2}$, i.e $\sigma(w_1, w_2) = \frac{\vec{w_1} \cdot \vec{w_2}}{\|\vec{w_1}\| \|\vec{w_2}\|}$. The latent semantic kernels (Siolas and d'Alch Buc, 2000; Cristianini et al., 2001) derive G by applying LSA, resulting in a valid kernel.

Another methods to design a valid kernel is to represent words as word vectors and compute σ as their scalar product between such vectors. For example, in (Bloehdorn et al., 2006), bag of hyponyms and hypernyms (up to a certain level of WordNet hierarchy) were used to build such vectors. We will refer to such similarity as WL (word list).

3 Smoothing Partial Tree Kernel (SPTK)

Combining lexical and structural kernels provides clear advantages on all-vs-all words similarity, which tends to semantically diverge. Indeed syntax provides the necessary restrictions to compute an effective semantic similarity. Following this idea, Bloedhorn & Moschitti (2007a) modified step (i) of Δ_{STK} computation as follows: (i) if n_1 and n_2 are pre-terminal nodes with the same number of children, $\Delta_{STK}(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} \sigma(lex(n_1), lex(n_2))$, where lex returns the node label. This allows to match fragments having same structure but different leaves by assigning a score proportional to the product of the lexical similarities of each leaf pair. Although it is an interesting kernel, the fact that lexicals must belong to the leaf nodes of exactly the same structures limits its applications. Trivially, it cannot work on dependency trees. Hereafter, we define a much more general smoothed tree kernel that can be applied to any tree and exploit any combination of lexical similarities, respecting the syntax enforced by the tree.

3.1 SPTK Definition

If n_1 and n_2 are leaves then $\Delta_\sigma(n_1, n_2) = \mu\lambda\sigma(n_1, n_2)$; else

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

where σ is any similarity between nodes, e.g. between their lexical labels, and the other variables are the same of PTK.

3.2 Soundness

A completely formal proof of the validity of the Eq. 2 is beyond the purpose of this paper (mainly due to space reason). Here we give a first sketch: let us consider σ as a string matching between node labels and $\lambda = \mu = 1$. Each recursive step of Eq. 2 can be seen as a summation of $(1 + \prod_{j=1}^{l(\vec{I}_1)} \Delta_{STK}(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j})))$, i.e. the Δ_{STK} recursive equation (see Sec. 2.2.1), for all subsequences of children $c_{n_1}(\vec{I}_{1j})$. In other words, PTK is a summation of an exponential number of STKs, which are valid kernels. It follows that PTK is a kernel. Note that the multiplication by λ and μ elevated to any power only depends on the target fragment. Thus, it just gives an additional weight to the fragment and does not violate the Mercer's conditions. In contrast, the multiplication by $\sigma(n_1, n_2)$ does depend on both comparing examples, i.e. on n_1 and n_2. However, if the matrix $\big[\sigma(n_1, n_2)\big] \forall n_1, n_2 \in f \in \mathcal{F}$ is positive semi-definite, a decomposition exists such that $\sigma(n_1, n_2) = \phi(n_1)\phi(n_2) \Rightarrow \Delta_\sigma(n_1, n_2)$ can be written as $\sum_{i=1}^{|\mathcal{F}|} \phi(n_1)\chi_i(n_1)\phi(n_2)\chi_i(n_2)$ = $\sum_{i=1}^{|\mathcal{F}|} \phi_\sigma(n_1)\phi_\sigma(n_2)$ (see Section 2.2), which proves SPTK to be a valid kernel.

3.3 Efficient Evaluation

We followed the idea in (Moschitti, 2006a) for efficiently computing SPTK. We consider Eq. 2 evaluated with respect to sequences of different length p; it follows that

$$\Delta(n_1, n_2) = \mu\sigma(n_1, n_2)\big(\lambda^2 + \sum_{p=1}^{m} \Delta_p(c_{n_1}, c_{n_2})\big),$$

where Δ_p evaluates the number of common subtrees rooted in subsequences of exactly p children (of n_1 and n_2) and $m = min\{l(c_{n1}), l(c_{n2})\}$. Given the two child sequences $s_1 a = c_{n_1}$ and $s_2 b = c_{n_2}$ (a and b are the last children)

$$\Delta_p(s_1 a, s_2 b) = \Delta(a, b) \times \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1| - i + |s_2| - r} \times \times \Delta_{p-1}(s_1[1:i], s_2[1:r])$$

where $s_1[1 : i]$ and $s_2[1 : r]$ are the child subsequences from 1 to i and from 1 to r of s_1 and s_2. If we name the double summation term as D_p, we can

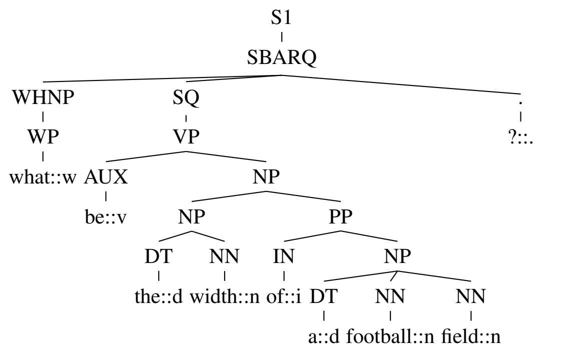

Figure 1: Constituent Tree (CT)

rewrite the relation as:

$$\Delta_p(s_1a, s_2b) = \begin{cases} \Delta(a,b)D_p(|s_1|,|s_2|) \text{ if } \sigma(a,b) > 0; \\ 0 \qquad\qquad\qquad\qquad otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

$$D_p(k,l) = \Delta_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k,l-1) \\ +\lambda D_p(k-1,l) - \lambda^2 D_p(k-1,l-1)$$

By means of the above relation, we can compute the child subsequences of two sequences s_1 and s_2 in $O(p|s_1||s_2|)$. Thus the worst case complexity of the SPTK is identical to PTK, i.e. $O(p\rho^2|N_{T_1}||N_{T_2}|)$, where ρ is the maximum branching factor of the two trees. The latter is very small in natural language parse trees and we also avoid the computation of node pairs with non similar labels.

We note that PTK generalizes both (i) SK, allowing the similarity between sequences (node children) structured in a tree and (ii) STK, allowing the computation of STK over any possible pair of subtrees extracted from the original tree. For this reason, we do not dedicate additional space on the definition of the smoothed SK or smoothed STK, which are in any case important corollary findings of our research.

3.4 Innovative Features of SPTK

The most similar kernel to SPTK is the Syntactic Semantic Tree Kernel (SSTK) proposed in (Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b). However, the following aspects show the remarkable innovativeness of SPTK:

- SSTK can only work on constituency trees and not on dependency trees (see (Moschitti, 2006a)).
- The lexical similarity in SSTK is only applied to leaf nodes in exactly the same syntactic constituents. Only complete matching of the structure of subtrees is allowed: there is absolutely no flexibility, e.g. the NP structure "cable television system" has no match with the NP "video streaming system". SPTK provides matches between all possible relevant subparts, e.g. "television system" and "video system" (so also exploiting the meaningful similarity between "video" and "television").
- The similarity in the PTK equation is added such that SPTK still corresponds to a scalar product in the semantic/structure space[2].
- We have provided a fast evaluation of SPTK with dynamic programming (otherwise the computation would have required exponential time).

4 Dependency Tree Structures

The feature space generated by the structural kernels, presented in the previous section, obviously depends on the input structures. In case of PTK and SPTK different tree representations may lead to engineer more or less effective syntactic/semantic feature spaces. The next two sections provide our representation models for dependency trees and their discussion.

4.1 Proposed Computational Structures

Given the following sentence:

(s1) *What is the width of a football field?*

The representation tree for a phrase structure paradigm leaves little room for variations as shown by the constituency tree (CT) in Figure 1. We apply lemmatization to the lexicals to improve generalization and, at the same time, we add a generalized PoS-tag, i.e. noun (n::), verb (v::), adjective (::a), determiner (::d) and so on, to them. This is useful to measure similarity between lexicals belonging to the same grammatical category.

In contrast, the conversion of dependency structures in computationally effective trees (for the above kernels) is not straightforward. We need to decide the role of lexicals, their grammatical functions (GR), PoS-tags and dependencies. It is natural

[2]This is not trivial: for example if sigma is added in Eq. 2 by only multiplying the λ^{d1+d2} term, no valid space is generated.

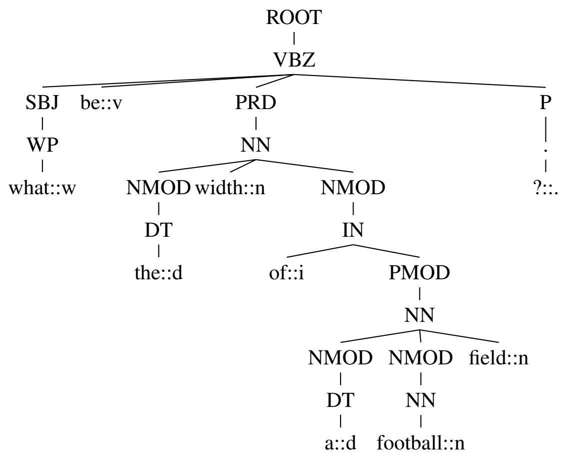

Figure 2: PoS-Tag Centered Tree (PCT)

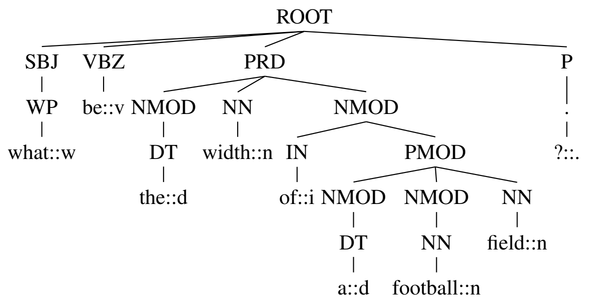

Figure 3: Grammatical Relation Centered Tree (GRCT)

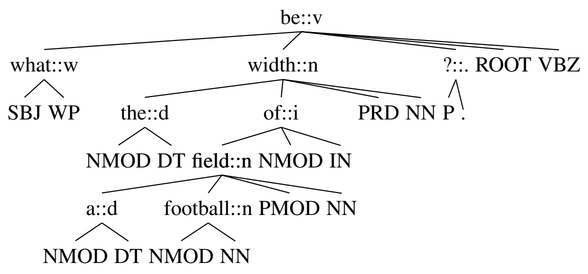

Figure 4: Lexical Centered Tree (LCT)

to associate edges with dependencies but, since our kernels cannot process labels on the arcs, they must be associated with tree nodes. The basic idea of our structures is to use (i) one of the three kinds of information above as central node, from which dependencies are drawn and (ii) all the other information as features (in terms of additional nodes) attached to the central nodes.

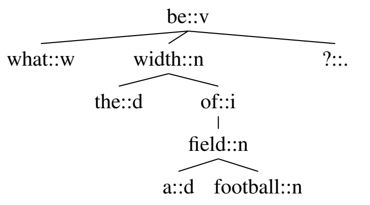

Figure 5: Lexical Only Centered Tree (LOCT)

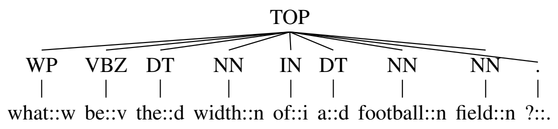

Figure 6: Lexical and PoS-Tag Sequences Tree (LPST)

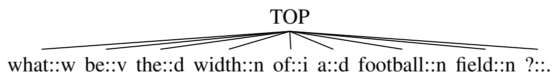

Figure 7: Lexical Sequences Tree (LST)

We define three main trees: the PoS-Tag Centered Tree (PCT), e.g. see Figure 2, where the GR is added as father and the lexical as a child; the GR Centered Tree (GRCT), e.g. see Figure 3, where the PoS-Tags are children of GR nodes and fathers of their associated lexicals; and the Lexical Centered Tree (LCT), e.g. see Figure 4, in which both GR and PoS-Tag are added as the rightmost children.

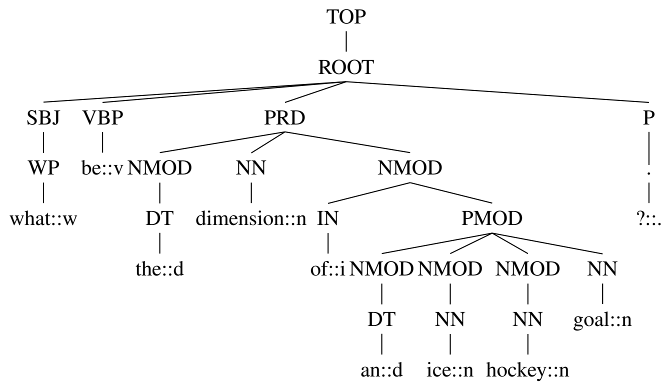

Figure 8: Grammatical Relation Centered Tree of (s2)

4.2 Comparative Structures

To better study the role of the above dependency structures, especially from a performance perspective, we define additional structures: the Lexical Only Centered Tree (LOCT), e.g. see Figure 5, which is an LCT only containing lexical nodes; the Lexical and PoS-Tag Sequences Tree (LPST), e.g. see Figure 6, which ignores the syntactic structure of the sentence being a simple sequence of PoS-Tag nodes, where lexicals are simply added as children; and the Lexical Sequence Tree (LST), where only lexical items are leaves of a single root node. PTK and PSTK applied to it simulates a standard SK and an SK with smoothing, respectively.

4.3 Structural Features

Section 2 has already described the kind of features generated by SK, STK and PTK. However, it is interesting to analyze what happens when SPTK is applied. For example, given the following sentence syntactically and semantically similar to s1:

(s2) *What is the dimension of an ice hockey goal?*

Figure 8 shows the corresponding GRCT, whose largest PTK fragment shared with the GRTC of s1 (Fig. 3) is: *(ROOT (SBJ (WP (what::w))) (PRD (NMOD (DT (the::d))) (NN) (NMOD (IN (of::i)) (PMOD (NMOD (DT)) (NMOD (NN)) (NN)))) (P (. (?::.))))*. If smoothing is applied the matching is almost total, i.e. also the children: *width::n/dimension::n*, *football::n/hockey::n* and *field::n/goal::n* will be matched (with a smoothing equal to the product of their similarities).

The matching using LCT is very interesting: without smoothing, the largest subtree is: *(be::v (what::w (SBJ) (WP)) (ROOT))*; when smoothing is used only the fragment *(NMOD (NN (ice::n))* will not be part of the match.

This suggests that LCT will probably receive the major benefit from smoothing. Additionally, with respect to all the above structures, LCT is the only one that can produce only lexical fragments, i.e. paths only composed by similar lexical nodes constrained by syntactic dependencies. All the other trees produce fragments in which lexicals play the role of features of GR or PoS-Tag nodes.

5 Experiments

The aim of the experiments is to analyze different levels of representation, i.e. structure, for syntactic dependency parses. At the same time, we compare with the constituency trees and different kernels to derive the best syntactic paradigm for convolution kernels. Most importantly, the role of lexical similarity embedded in syntactic structures will be investigated. For this purpose, we first carry out extensive experiments on coarse and fine grained QC and then we verify our findings on a completely different task, i.e. Argument Classification in SRL.

5.1 General experimental setup

Tools: for SVM learning, we extended the SVM-LightTK software[3] (Moschitti, 2006a) (which includes structural kernels in SVMLight (Joachims, 2000)) with the smooth match between tree nodes. For generating constituency trees, we used the Charniak parser (Charniak, 2000) whereas we applied LTH syntactic parser (described in (Johansson and Nugues, 2008a)) to generate dependency trees.

Lexical Similarity: we used the Eq. 1 with $\omega_1 = \omega_2 = 1$ and σ is derived with both approaches described in Sec. 2.3. The first approach is LSA-based: LSA was applied to ukWak (Baroni et al., 2009), which is a large scale document collection made by 2 billion tokens. More specifically, to build the matrix M, POS tagging is first applied to build rows with pairs $\langle$lemma, ::POS$\rangle$, or 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. The SVD reduction is then applied to M, with a dimensionality cut of $l = 250$. The second approach uses the similarity based on word list (WL) as provided in (Li and Roth, 2002).

Models: SVM-LightTK is applied to the different tree representations discussed in Section 4. Since PTK and SPTK are typically used in our experiments, to have a more compact acronym for each model, we associate the latter with the name of the structure, i.e. this indicates that PTK is applied to it. Then the presence of the subscript $_{WL}$ and $_{LSA}$ indicates that SPTK is applied along with the corresponding similarity, e.g. LCT$_{WL}$ is the SPTK kernel applied to LCT structure, using WL similarity. We experiment with multi-classification, which we model through *one-vs-all* scheme by selecting the category associated with the maximum SVM margin. The quality of such classification is measured with accuracy. We determine the statistical significance by using the model described in (Yeh, 2000) and implemented in (Padó, 2006).

The parameterization of each classifier is carried on a held-out set (30% of the training) and concerns

[3]http://disi.unitn.it/moschitti/Tree-Kernel.htm

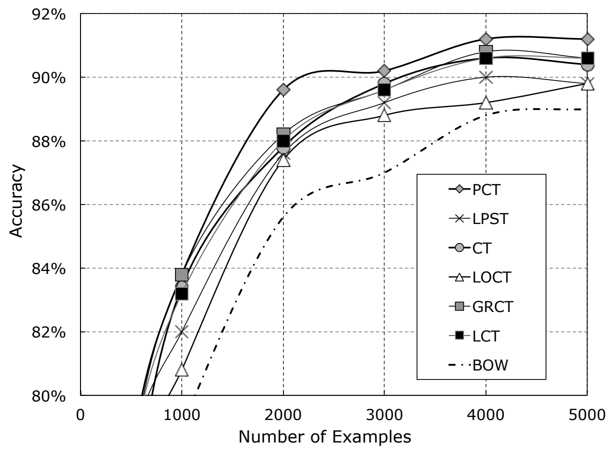

Figure 9: Learning curves: comparison with no similarity

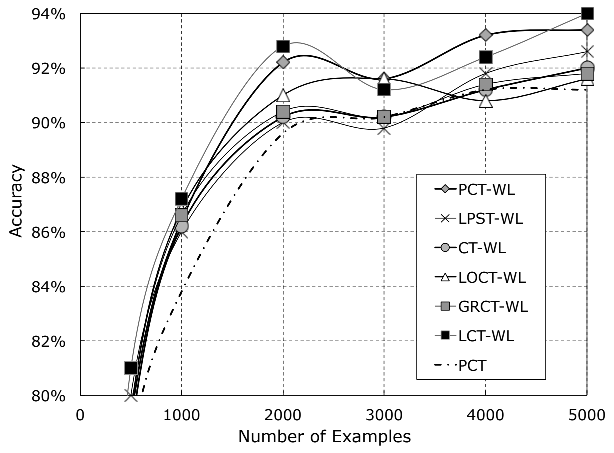

Figure 10: Learning curves: comparison with similarity

with the setting of the trade-off parameter (option -c) and the Leaf Weight (LeW) (see Sec. 5.2), which is used to linearly scale the contribution of the leaf nodes. In contrast, the cost-factor parameter of the SVM-LightTK is set as the ratio between the number of negative and positive examples for attempting to have a balanced Precision/Recall.

5.2 QC experiments

For these experiments, we used the UIUC dataset (Li and Roth, 2002). It is composed by a training set of 5,452 questions and a test set of 500 questions[4]. Question classes are organized in two levels: 6 coarse-grained classes (like `ENTITY` or `HUMAN`) and 50 fine-grained sub-classes (e.g. `Plant`, `Food` as subclasses of `ENTITY`).

The outcome of the several kernels applied to several structures for the coarse and fine grained QC is reported in Table 1. The first column shows the experimented models, obtained by applying PTK/SPTK to the structures described in Sec. 4. The last two rows are: CT-STK, i.e. STK applied to a constituency tree and BOW, which is a linear kernel applied to lexical vectors. Column 2, 3 and 4 report the accuracy using no, LSA and WL similarity, where LeW is the amplifying parameter, i.e. a weight associated with the leaves in the tree. The last three columns refer to the fine grained task.

It is worth nothing that when no similarity is applied: (i) BOW produces high accuracy, i.e. 88.8% but it is improved by STK (the current state-of-the-art[5] in QC (Zhang and Lee, 2003; Moschitti et al., 2007)); (ii) PTK applied to the same tree of STK produces a slightly lower value (non-statistically significant difference); (iii) interestingly, when PTK is instead applied to dependency structures, it improves STK, i.e. 91.60% vs 91.40% (although not significantly); and (iv) LCT, strongly based on lexical nodes, is the least accurate, i.e 90.80% since it is obviously subject to data sparseness (fragments only composed by lexicals are very sparse).

The very important results can be noted when lexical similarity is used, i.e. SPTK is applied: (a) all the syntactic-base structures using both LSA or WL improve the classification accuracy. (b) CT gets the lowest improvement whereas LCT achieves an impressive result of 94.80%, i.e more than 41% of relative error reduction. It seems that the lexical similar paths when driven by syntax produces accurate features. Indeed, when syntax is missing such as for the unstructured lexical path of LST$_{LSA}$, the accuracy does not highly improve or may also decrease. Additionally, the result of our best model is so high that its errors only refer to questions like *What did Jesse Jackson organize ?*, where the classifier selected *Entity* instead of $Human$ category. These refer to clear cases where a huge amount of background knowledge is needed for deriving the exact solution.

Finally, on the fine grained experiments LCT still produces the most accurate outcome again exceeding the state-of-the-art (Zhang and Lee, 2003), where WL significantly improves on all models (CT included).

[4]http://cogcomp.cs.illinois.edu/Data/QA/QC/

[5]Note that in (Bloehdorn and Moschitti, 2007b), higher accuracy values for smoothed STK are shown for different parameters but the best according to a validation set is not highlighted.

	COARSE						FINE					
	NO		LSA		WL		NO		LSA		WL	
	LeW	Acc.	*LeW*	Acc.	*LeW*	Acc.	*LeW*	Acc.	*LeW*	Acc.	*LeW*	Acc.
CT	4	90.80%	2	91.00%	5	92.20%	4	84.00%	5	83.00%	7	86.60%
GRCT	3	**91.60%**	4	92.60%	2	**94.20%**	3	83.80%	4	83.20%	2	85.00%
LCT	1	90.80%	1	**94.80%**	1	**94.20%**	0.33	**85.40%**	1	86.20%	0.33	**87.40%**
LOCT	1	89.20%	1	93.20%	1	91.80%	1	**85.40%**	1	**86.80%**	1	87.00%
LST	1	88.20%	1	85.80%	1	89.60%	1	84.00%	1	80.00%	1	85.00%
LPST	3	89.40%	1	89.60%	1	92.40%	3	84.20%	4	82.20%	1	84.60%
PCT	4	91.20%	4	92.20%	5	93.40%	4	84.80%	5	84.00%	5	85.20%
CT-STK	-	91.20%	-	-	-	-	-	82.20%	-	-	-	-
BOW	-	88.80%	-	-	-	-	-	83.20%	-	-	-	-

Table 1: Accuracy of structural several kernels on different structures for coarse and fine grained QC

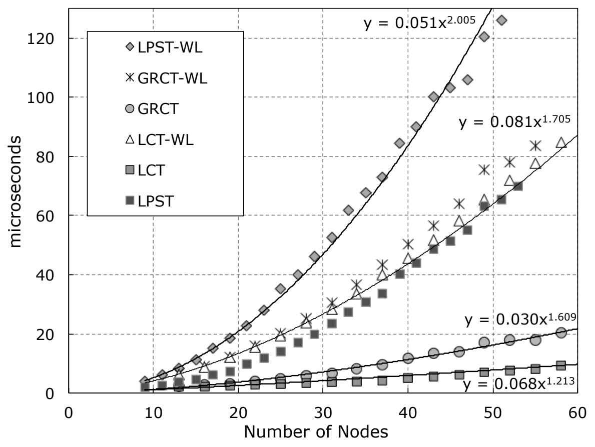

Figure 11: Micro-seconds for each kernel computation

5.3 Learning curves

It is interesting to study the impact of syntactic/semantic kernels on the learning generalization. For this purpose, Fig. 9 reports the learning curve of the previous models without lexical similarity whereas Fig. 10 shows the complete SPTK behavior through the different structures. We note that when no similarity is used the dependency trees better generalize than constituency trees or non-syntactic structures like LPST or BOW. When WL is activated, all models outperform the best kernel of the previous pool, i.e. PCT (see dashed line of Fig. 10 or the top curve in Fig. 9).

5.4 Kernel Efficiency

We plotted the average running time of each computation of PTK/SPTK applied to the different structures. We divided the examples from QC based on the number of nodes in each example. Figure 11 shows the elapsed time in function of the number of nodes for different tree representations. We note that: (i) when the WL is not active, LCT and GRCT are very fast as they impose hierarchical matching of subtrees; (ii) when the similarity is activated, LCT$_{WL}$ and GRCT$_{WL}$ tend to match many more tree fragments thus their complexity increases. However, the equations of the curve fit, shown in the figure, suggests that the trend is sub-quadratic ($x^{1.7}$). Only LPST$_{WL}$, which has no structure, matches a very large number of sequences of nodes, when the similarity is active. This increases the complexity, which results in an order higher than 2.

5.5 FrameNet Role Classification Experiments

To verify that our findings are general and that our syntactic/semantic dependency kernels can be effectively exploited for diverse NLP tasks, we experimented with a completely different application, i.e. FrameNet SRL classification (gold standard boundaries). We used the FrameNet version 1.3 with the 90/10% split between training and test set (i.e 271,560 and 30,173 examples respectively), as defined in (Johansson and Nugues, 2008b), one of the best system for FrameNet parsing. We used the LTH dependency parser. LSA was applied to the BNC corpus, the source of the FrameNet annotations.

For each of 648 frames, we applied SVM along with the best models for QC, i.e. GRCT and LCT, to learn its associated binary role classifiers (RC) for a total of 4,254 classifiers. For example, Figure 12 shows the LCT representation of the first two roles of the following sentence:

$[Bootleggers]_{\text{CREATOR}}, then$ **copy** $[the\ film]_{\text{ORIGINAL}}$
$[onto\ hundreds\ of\ VHS\ tapes]_{\text{GOAL}}$

Table 2 shows the results of the different multi-classifiers. GRCT and LCT show a large accuracy, i.e. 87.60. This improves up to 88.74 by activating the LSA similarity. The combination GRCT$_{LSA}$+LCT$_{LSA}$ significantly improves the

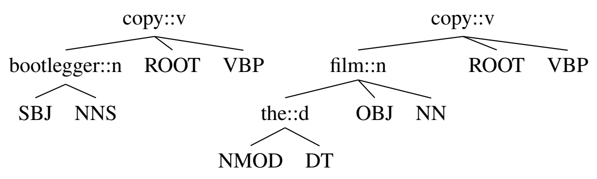

Figure 12: LCT Examples for argument roles

Kernel	Accuracy
GRCT	87.60%
GRCT$_{LSA}$	88,61%
LCT	87.61%
LCT$_{LSA}$	88.74%
GRCT + LCT	87.99%
GRCT$_{LSA}$ + LCT$_{LSA}$	**88.91%**

Table 2: Argument Classification Accuracy

above model, achieving 88.91%. This is very close to the state-of-the-art of SRL for classification (using a single classifier, i.e. no joint model), i.e. 89.6%, achieved in (Johansson and Nugues, 2008b). Finally, it should be noted that, to learn and test the SELF_MOTION multi-classifier, containing 14,584 examples, distributed on 22 roles, SVM-SPTK employed 1.5 h and 10 minutes, respectively[6].

6 Final Remarks and Conclusion

In this paper, we have proposed a study on representation of dependency structures for the design of effective structural kernels. Most importantly, we have defined a new class of kernel functions, i.e. SPTKs, that carry out syntactic and lexical similarities on the above structures. SPTK exploits the latter by providing generalization trough lexical similarities constrained in them. This allows for automatically generating feature spaces of generalized syntactic/semantic dependency substructures.

To test our models, we carried out experiments on QC and SRL. These show that by exploiting the similarity between two sets of words carried out according to their dependency structure leads to an unprecedented result for QC, i.e. 94.8% of accuracy. In contrast, when no structure is used the accuracy does not significantly improves. We have also provided a fast algorithm for the computation of SPTK and empirically shown that it can easily scale.

It should be noted that our models are not absolutely restricted to QC and SRL. Indeed, since most of the NLP applications are based on syntactic and lexical representations, SPTK will have a major impact in most of them, e.g.:

- Question Answering, the high results for QC will positively impact on the overall task.
- SRL, SPTK alone reaches the state-of-the-art (SOA) (only 0.7% less) in FrameNet role classification. This is very valuable as previous work showed that tree kernels (TK) alone perform lower than models based on manually engineered features for SRL tasks, e.g., (Moschitti, 2004; Giuglea and Moschitti, 2004; Giuglea and Moschitti, 2006; Moschitti, 2006b; Che et al., 2006; Moschitti et al., 2008). Thus for the first time in an SRL task, a general tree kernel reaches the same accuracy of heavy manual feature design. This also suggests an improvement when used in combinations with manual feature vectors.
- Relation Extraction and Pronominal Coreference, whose state-of-the-art for some tasks is achieved with the simple STK-CT (see (Zhang et al., 2006) and (Yang et al., 2006; Versley et al., 2008), respectively).
- In word sense disambiguation tasks, SPTK can generalize context according to syntactic and semantic constraints (selectional restrictions) making very effective distributional semantic approaches.
- In Opinion Mining SPTK will allow to match sentiment words within their corresponding syntactic counterparts and improve the state-of-the-art (Johansson and Moschitti, 2010b; Johansson and Moschitti, 2010a).
- Experiments on Recognizing Textual Entailment (RTE) tasks, the use of SSTK (instead of STK-CT) improved the state-of-the-art (Mehdad et al., 2010). SPTK may provide further enhancement and innovative and effective dependency models.

The above points also suggest many promising future research directions, which we would like to explore.

[6]Using one of the 8 processors of an Intel(R) Xeon(R) CPU E5430 @ 2.66GHz machine, 32Gb Ram.

Acknowledgements

This work has been partially supported by the EC project FP247758: Trustworthy Eternal Systems via Evolving Software, Data and Knowledge (EternalS).

References

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

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of CoNLL-2005*, pages 1–8, Ann Arbor, Michigan. Association for Computational Linguistics.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Proceedings of ECIR 2007, Rome, Italy*.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *In Proceedings of CIKM '07*.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of ICDM 06, Hong Kong, 2006*.

Ulrik Brandes. 2001. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25:163–177.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47.

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.

Horst Bunke and Kim Shearer. 1998. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19(3-4):255–259, March.

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

O. Chapelle, B. Schlkopf, and A. Zien. 2006. *Semi-Supervised Learning*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA, 09.

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

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 73–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

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*.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jim Cowie, Joe Guthrie, and Louise Guthrie. 1992. Lexical disambiguation using simulated annealing. In *in COLING*, pages 359–365.

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.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, pages 423–429, Barcelona, Spain, July.

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*.

Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. 2007. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 209–216, New York, NY, USA. ACM.

Linton C. Freeman. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41.

Hagen Fürstenau and Mirella Lapata. 2009. Graph alignment for semi-supervised semantic role labeling. In *In Proceedings of EMNLP '09*, pages 11–20, Morristown, NJ, USA.

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

A.-M. Giuglea and A. Moschitti. 2006. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of ACL*, Sydney, Australia.

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*, 2(2):pp. 205–224.

Zellig Harris. 1964. Distributional structure. In Jerrold J. Katz and Jerry A. Fodor, editors, *The Philosophy of Linguistics*. Oxford University Press.

J. J. Jiang and D. W. Conrath. 1997. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*.

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

Richard Johansson and Alessandro Moschitti. 2010a. Reranking models in fine-grained opinion analysis. In *Proceedings of the 23rd International Conference of Computational Linguistics (Coling 2010)*, pages 519–527, Beijing, China.

Richard Johansson and Alessandro Moschitti. 2010b. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 67–76, Uppsala, Sweden.

Richard Johansson and Pierre Nugues. 2008a. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 183–187, Manchester, United Kingdom.

Richard Johansson and Pierre Nugues. 2008b. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*, Manchester, UK, August 18-22.

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*.

Claudia Leacock and Martin Chodorow, 1998. *Combining Local Context and WordNet Similarity for Word Sense Identification*, chapter 11, pages 265–283. The MIT Press.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of ACL'02*.

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

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2005. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the American Association for Artificial Intelligence (AAAI 2006)*, Boston, July.

Rada Mihalcea. 2005. unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *In HLT/EMNLP 2005*, pages 411–418.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of ACL*, Barcelona, Spain.

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

Alessandro Moschitti. 2006b. Making tree kernels practical for natural language learning. In *Proccedings of EACL'06*.

Roberto Navigli and Mirella Lapata. 2010. An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Sebastian Pado and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2).

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

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004a. WordNet::Similarity - Measuring the Relatedness of Concept. In *Proc. of 5th NAACL*, Boston, MA.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004b. Wordnet::similarity - measuring the relatedness of concepts. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Demonstration Papers*, pages 38–41, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453.

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

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.

Georges Siolas and Florence d'Alch Buc. 2000. Support vector machines based on a semantic kernel for text categorization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 5*, page 5205. IEEE Computer Society.

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*.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008. Coreference systems based on kernels methods. In *The 22nd International Conference on Computational Linguistics (Coling'08)*, Manchester, England.

Zhibiao Wu and Martha Palmer. 1994. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138, New Mexico State University, Las Cruces, New Mexico.

Xiaofeng Yang, Jian Su, and Chewlim Tan. 2006. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. COLING-ACL 06*.

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

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

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM Press.

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

Peixiang Zhao, Jiawei Han, and Yizhou Sun. 2009. P-Rank: a comprehensive structural similarity measure over information networks. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 553–562, New York, NY, USA. ACM.