Exploiting Structure and Semantics for Expressive Text Kernels

Stephan Bloehdorn
Institute AIFB
Knowledge Management Research Group
University of Karlsruhe
D-76128 Karlsruhe, Germany
bloehdorn@aifb.uni-karlsruhe.de

Alessandro Moschitti
Department of Information
and Communication Technology
University of Trento
I-38050 Povo di Trento, Italy
moschitti@dit.unitn.it

ABSTRACT

Several problems in text categorization are too hard to be solved by standard bag-of-words representations. Work in kernel-based learning has approached this problem by (i) considering information about the syntactic structure of the input or by (ii) incorporating knowledge about the semantic similarity of term features. In this paper, we propose a generalized framework consisting of a family of kernels that jointly incorporates syntax and semantics. We show that both components can be flexibly adapted and tuned towards the particular application domain. We demonstrate the power of this approach in a series of experiments on two diverse datasets, each of which presents a non-standard text categorization problem: one for the classification of natural language questions from a TREC question answering dataset and the other for the automated assignment of ICT-9 categories to short textual fragments of medical diagnoses.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Information Filtering

General Terms

Algorithms, Theory, Experimentation

Keywords

machine learning, text categorization, kernel methods, tree kernels, syntactic parsing, lexical semantics

1. INTRODUCTION

Text Categorization (TC) systems [17], which aim to automatically classify textual documents, are a major application domain of modern machine learning techniques. These are applied to documents represented by term vectors (i.e. *bag-of-words* representation), which is still the most popular choice in text mining tasks.

According to these models, the input documents are encoded as vectors whose dimensions correspond to the terms in the overall training corpus. The inner product (or the cosine) between two such vectors is used as kernel hence making the similarity of two documents dependant only on the amount of terms they share. This approach has an appealing simplicity and has produced good results in cases where sufficient training data is available.

However, one of the main shortcomings of this kind of representation is that it does not encode the syntactic structure of the input text and provides a too much approximated lexical semantics. As a consequence, the resulting models either cannot take advantage of complex patterns, e.g. verbal phrases with identical (or similar) verbs or are not sufficiently robust with respect to variations in the used terminology.

Several modifications to this rather flat representation have aimed at incorporating syntactic information (e.g. n-gram models, part-of-speech tags of terms, and so on.) or knowledge about semantic dependencies (e.g. latent semantic indexing). In particular, there has been increased interest in incorporating such a-priori knowledge within Kernel-based learning algorithms like Support Vector Machines (SVMs) [11] by means of a specific choice of the employed kernel function [6].

For the case of syntactic representations, *Tree Kernels* [5, 21, 16] have been proposed as a powerful framework to exploit parse trees of the input texts. For the case of lexical semantics, *Semantic Kernels* are methods that exploit background information from semantic networks such as WordNet [19, 14, 2] or from statistical models of term co-occurrence [7] to make different, though semantically similar, terms contribute to the overall similarity of the input tokens. While both approaches seem intuitive and powerful, natural language draws from both, syntax and semantics. Therefore, finding principled techniques for approaching both extensions at the same time within a unified framework appears to be a promising research line.

In this paper, we present a generalization of the original Tree Kernel function [5] (cf. section 2) which directly incorporates semantic background information. Based on our own preliminary results presented in [3], we introduce a family of kernels which we call *Semantic Syntactic Tree Kernels (SSTKs)* (cf. section 3). The new kernels build upon linguistic structure and background knowledge about the semantic dependencies of terms at the same time.

Technically, the proposed kernels introduce two new components, namely an embedded semantic term kernel and a leaf weighting component, to improve the matching of tree fragments containing terminal nodes (cf. section 3.1).

We show that both building blocks of the new kernel, namely syntax and semantics, can be flexibly adapted and tuned towards the particular application domain. Regarding the semantic components, we proposed two models for the design of semantic term kernels, one based on taxonomic background knowledge (cf. section 3.3) and one based on latent semantics (cf. 3.4).

We demonstrate the power of the new class of kernels in an extensive set of experiments on two diverse datasets, each of which presents a hard text categorization problem (cf. section 4). In the first experiment, we deal with the problem of classifying natural language questions from a TREC question answering dataset (cf. section 4.2) while in the second experiment, we aim at the automated assignment of ICT-9 categories to short textual fragments of medical diagnoses (CMC corpus, cf. section 4.3).

Our experiments on question classification show that the new models remarkably outperform both, bag-of-words kernels and the original tree kernels while on the CMC corpus, the plain tree kernel representation alone is capable of an impressive improvement of the accuracy. Results and related work are summarized in a concluding discussion (cf. section 5) including an overview on envisioned future work.

2. BACKGROUND

In this section, we shortly review the technical background for our contribution. We briefly review the basic concepts of SVMs and Kernel Methods and in particular tree kernels for syntactic structures.

2.1 SVMs and Kernel Methods

Support Vector Machines (SVMs) [20] are state-of-the-art learning methods based on the principle of linear classification. The good generalization capabilities of SVMs based on the maximum margin principle are theoretically well grounded in statistical learning theory and SVMs have empirically proved to produce highly effective classifiers.

Another distinguishing feature of SVMs is their capability of naturally incorporating domain-specific notions of item similarity by means of a corresponding kernel function.

Formally, any function κ that for all $x, z \in X$ satisfies $\kappa(x, z) = \langle\phi(x), \phi(z)\rangle$, is a valid kernel, whereby $\mathcal{X}$ is the input domain under consideration and ϕ is a suitable mapping from $\mathcal{X}$ to a feature (Hilbert-) space $\mathcal{F}$. The choice of a particular kernel function thus implies an implicit mapping to a feature space different from the input space $\mathcal{X}$ which is (hopefully) well suited to capture the geometry of the classification problem.

Kernels can be designed by either choosing an explicit mapping function ϕ and incorporating it into an inner product or by directly defining the kernel function κ while making sure that it complies with the requirement of being a positive semi-definite function. Several closure properties aid the construction of valid kernels such as closure under sum, product, multiplication by a positive scalar and combination with well-known kernel modifiers.

In particular, a given kernel κ can be normalized using the *cosine normalization modifier* given by $\kappa'(x, y) = (\kappa(x, y)) \,/\, (\sqrt{\kappa(x, x)}\sqrt{\kappa(y, y)})$ to produce kernel evaluations (i.e. similarity measures) normalized to absolute values between 0 and 1. The reader is referred to the rich literature for further information on SVMs and kernel methods, e.g. [18] for a comprehensive introduction.

2.2 Tree Kernels for Syntactic Structures

Let us introduce some notations. We define a tree as a connected directed graph with no cycles. *Trees* are denoted as $T_1, T_2, \ldots$; *tree nodes* are denoted as $n_1, n_2, \ldots$; and the set of nodes in tree T_i are denoted as N_{T_i}. We denote the set of all *substructures (fragments)* that occur in a given set of trees as $\{f_1, f_2, \ldots\} = \mathcal{F}$.

As the structures we will work with are parse trees, each node with its children is associated with a grammar production rule. The labels of the leaf nodes of the parse trees correspond to terms, i.e. *terminal symbols*, whereas the *preterminal symbols* are the parents of leaves. As an example Figure 1 shows a parse tree of the sentence fragment `‘‘Examination indicates bacterial pneumonia.’’` with some of its substructures.

Tree Kernels have been designed based on the idea of representing trees in terms of all their substructures (fragments). The job of the kernel function is then to efficiently count the number of tree substructures that are common to both argument trees. Note that the number of such fragments of a single tree can be obtained by evaluating the kernel function between the tree with itself.

Definition 1. Given two trees T_1 and T_2 we define the *(Subset-) Tree Kernel* [5] as:

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

where $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$, and where $I_i(n)$ is an indicator function which determines whether fragment f_i is rooted in node n.

Consequently, the value of Δ is equal to the number of common fragments rooted at nodes n_1 and n_2. Obviously, the naive enumeration over all tree fragments is computationally problematic as the number of substructures that need to be considered grows exponentially in the number of nodes of the input trees. However, as noted in previous work on tree kernels [5], we can efficiently compute Δ as follows:

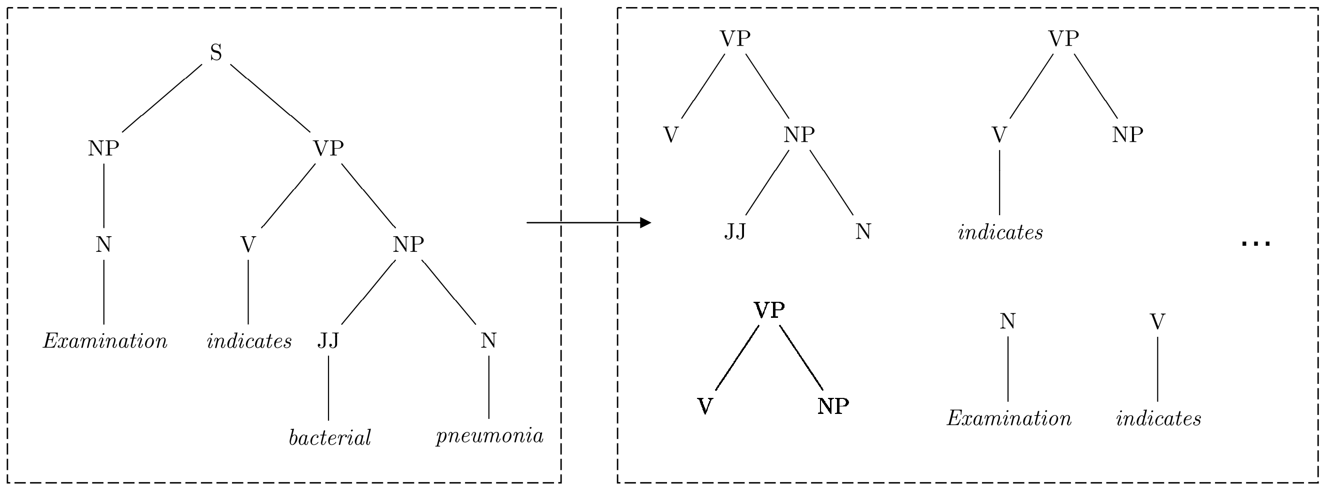

Figure 1: Simplified parse tree of the sentence "Examination indicates bacterial pneumonia." with examples for some of its fragments. For subset trees, fragments are not required to include all possible productions up to the leaves of the subtree.

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

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch^j_{n_1}, ch^j_{n_2})).$$

where $nc(n_1)$ is the number of children of n_1 and ch^j_n is the j-th child of node n. Note that, since the productions are the same, $nc(n_1) = nc(n_2)$. Of course, the kernel can again be normalized using the cosine normalization modifier.

Additionally, a decay factor λ can be added by modifying steps (2) and (3) as follows:

2. $\Delta(n_1, n_2) = \lambda$,
3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch^j_{n_1}, ch^j_{n_2}))$.

The role of the decay factor is to penalize larger tree structures by giving them less weight in the overall summation.

3. DESIGNING SEMANTIC SYNTACTIC TREE KERNELS

The Tree Kernel introduced in the previous section relies on the intuition of counting all common substructures of two trees. However, if two trees have similar substructures that employ different though related terminology at the leaves, they will not be matched. From a semantic point of view, this is an evident drawback. For example, `''indicates bacterial pneumonia''` should be more related to `''indicates viral infection''` than to `''indicates cystic fibrosis''` due to the higher similarity of the terminal nodes `pneumonia` vs. `infection` and `bacterial` vs. `viral`. We will now look at some techniques to simulate such an effect.

3.1 Semantic Smoothing for Tree Kernels

Together with the standard bag-of-words representation for text documents, recent research work in the direction of *Semantic Kernels* has looked at ways to allow for *partial matches* between vector components [19, 18, 1, 2]. These kernels typically compute a smoothed function of the type $K' = \vec{d_1'} Q \vec{d_2}$ where $\vec{d_1'}$ and $\vec{d_2}$ are the feature vectors associated with the documents d_1 and d_2 whereas Q is a positive semi-definite *smoothing matrix* that encodes the similarity of terms.

In analogy with this kind of smoothing kernels, we are now interested in also counting *partial matches* between tree fragments within tree kernels. A partial match occurs when two fragments differ only by their terminal symbols, e.g. `[N [pneumonia]]` and `[N [infection]]`. In this case the match should give a contribution smaller than 1, depending on the semantic similarity of the respective terminal nodes. To ensure the validity of the overall kernel, this similarity needs to be a valid kernel on the terms itself and will be denoted by κ_S – we will look at ways to design such functions in sections 3.3 and 3.4. For this purpose, we first define the similarity of two such tree fragments.

Definition 2. For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the *Tree Fragment Similarity Kernel* as:

$$\kappa_{\mathcal{F}}(f_1, f_2) = comp(f_1, f_2) \prod_{t=1}^{nt(f_1)} \kappa_S(f_1(t), f_2(t))$$

where $comp(f_1, f_2)$ (compatible) is 1 if f_1 differs from f_2 only in the terminal nodes and is 0 otherwise, $nt(f_i)$ is the number of terminal nodes and $f_i(t)$ is the t-th terminal symbol of f_i (numbered from left to right).

Note that, as the tree fragments need to be compatible, they have the same number of terminal symbols at compatible positions. Conceptually, this means that the similarity of two tree fragments is above zero only if the tree fragments have an identical structure. The fragment similarity is evaluated as the product of all semantic similarities of corresponding terminal nodes (i.e. sitting at identical positions). It is maximal if all pairs have a similarity score of 1. We now define the overall tree kernel as the sum over the evaluations of $\kappa_{\mathcal{F}}$ over all pairs of tree fragments in the argument trees. Technically, this means changing the summation in the second formula of definition 1 as suggested by the following

Definition 3. Given two trees T_1 and T_2 we define the *Semantic Syntactic Tree Kernel* as:

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

where $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} I_i(n_1) I_j(n_2) \kappa_{\mathcal{F}}(f_i, f_j)$.

The naive evaluation of this kernel would require even more computation and memory than for the naive computation of the standard kernel as also all *compatible pairs of tree fragments* would need to be considered in the summation. Luckily, this enhanced kernel can again be evaluated in a far more efficient manner by exploiting the same recursion structure as the standard tree kernel by modifying the computation of Δ as follows:

1. if n_1 and n_2 are pre-terminals and $label(n_1) = label(n_2)$ then $\Delta(n_1, n_2) = \lambda\, \kappa_{\mathcal{S}}(ch_{n_1}^1, ch_{n_2}^1)$,

2. if n_1 and n_2 are not pre-terminals and the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;

3. if n_1 and n_2 are not pre-terminals and the productions at n_1 and n_2 are the same then:

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch_{n_1}^j, ch_{n_2}^j)).$$

Hereby, $label(n_i)$ is the label of node n_i and $\kappa_{\mathcal{S}}$ is a valid term similarity kernel. Note that since n_1 and n_2 are pre-terminals of a parse tree they can have only one child (i.e. $ch_{n_1}^1$ and $ch_{n_2}^1$) and such children are words.

Beside the novelty of taking into account tree fragments that are not identical it should be noted that in contrast to the application of lexical semantic similarities to simple term vectors in semantic smoothing kernels, the lexical semantic similarity is now constrained in the syntactic structures, which limit errors/noise due to incorrect (or not provided) word sense disambiguation.

3.2 Tuning Leaf Contribution

We have remarked that the parameter λ is well suited to downweight the contribution of larger tree structures. On the other hand, we want to allow fragments with many matching/similar leaves to contribute more to the overall kernel than fragments without leaves. For this purpose, we introduce an additional parameter, α in the computation of the Δ function by modifying step (1) of the above computation (i.e. leaf matching) as follows:

1. if n_1 and n_2 are pre-terminals and $label(n_1) = label(n_2)$ then $\Delta(n_1, n_2) = \alpha\, \lambda\, \kappa_{\mathcal{S}}(ch_{n_1}^1, ch_{n_2}^1)$,

While α could of course be used to further downweight the contribution of the leaves, we will typically select a value $\alpha > 1$ to allow for a stronger contribution.

3.3 Taxonomic Term Kernels

Up to now, we have regarded the term similarity kernel as a kind of "black-box" that hopefully produces accurate kernel values that correlate with the perceived similarity of the participating terms. In this section, we look at a way, initially proposed in [2] to encode such kernels by means of taxonomic background knowledge structures, e.g. WordNet.

The formal description of semantic kernels requires the introduction of some definitions. We denote terms as $t_1, t_2, \ldots \in \mathcal{T}$ and concepts as $c_1, c_2, \ldots \in \mathcal{C}$; we also sometimes use the somewhat informal disambiguation operator $c(\cdot)$ to map terms to concept representations.

Semantic Networks can be regarded as directed graphs semantically linking concepts by means of taxonomic relations (e.g. *[cat] is-a [mammal]*). Research in Computational Linguistics has led to a variety of well-known measures of semantic similarity in semantic networks. Such measures typically make use of several notions:

(i) The *distance* (d) of two concepts c_1 and c_2, is the number of superconcept edges between c_1 and c_2, where superconcepts are a subset of more important concepts, e.g. higher level concepts. In our study we consider all possible concepts, i.e. all WordNet synsets.

(ii) The *depth* (dep) of a concept refers to the distance of the concept to the unique root node (if the structure is not a perfect tree structure, we use the minimal depth).

(iii) The *lowest super ordinate (lso)* of two concepts refers to the concept with maximal depth that subsumes them both.

(iv) The probability $P(c)$ of encountering a concept c which can be estimated from corpus statistics. When probabilities are used, the measures follow the trail of information theory in quantifying the *information concept (IC)* of an observation as the negative log likelihood. We point the interested reader to [4] for a detailed and recent survey of the field.

In our work, we have looked at the following measures of term similarity:

Definition 4. Wu & Palmer:

$$sim_{WUP}(c_1, c_2) =$$

$$\frac{2\,dep(lso(c_1, c_2))}{d(c_1, lso(c_1, c_2)) + d(c_2, lso(c_1, c_2)) + 2\,dep(lso(c_1, c_2))}$$

Definition 5. Resnik (full-ic):

$$sim_{RES}(c_1, c_2) = -\log\, P(lso(c_1, c_2))$$

Definition 6. Lin:

$$sim_{LIN}(c_1, c_2) = \frac{2\,\log\, P(lso(c_1, c_2))}{\log\, P(c_1) + \log P\,(c_2)}$$

While some of these measures can be regarded as valid kernels themselves [14], we investigate a different approach. This idea, recently investigated in [2], is based on the observation that the more two concepts are similar the more they share common superconcepts. We thus model the similarity of two terms as a dot product of their respective superconcept vectors (i.e. vectors of all WordNet synsets).

Definition 7. The *Superconcept Kernel* $\kappa_{\mathcal{S}}$ for two concepts $c_i, c_j \in \mathcal{C}$ is given by $\kappa_{\mathcal{S}}(c_i, c_j) = \langle SC(c_i), SC(c_j)\rangle$, whereby $SC(\cdot)$ is a function $\mathcal{C} \rightarrow \mathrm{R}^{|\mathcal{C}|}$ that maps each concept to a real vector whose dimensions correspond to superconcepts present in the employed semantic network and the respective entries are determined by a particular weighting scheme.

$\kappa_{\mathcal{S}}$ is a valid kernel since it is defined explicitly in terms of a dot product computation. So far, however, we have left the details of the function $SC(\cdot)$ that maps concepts to its superconcepts unspecified. Based on the term similarity measures introduced above, we have investigated the use of different weighting schemes for the representation of the superconcepts motivated by the following considerations:

(i) the weight a superconcept $SC(\bar{c})_j$ receives in the vectorial description of concept $\bar{c}$ should be influenced by its distance from $\bar{c}$ and

(ii) the weight a superconcept $SC(\bar{c})_j$ receives in the vectorial description of concept $\bar{c}$ should be influenced by its overall depth in the semantic network.

In summary, we have investigated four different weighting schemes:

full No weighting, i.e. $SC(\bar{c})_j = 1$ for all superconcepts c_j of $\bar{c}$ and $SC(\bar{c})_j = 0$ otherwise.

full-ic Weighting using information content of $SC(\bar{c})_j$, i.e. $SC(\bar{c})_j = sim_{RES}(\bar{c}, c_j)$.

lin Weighting using the Lin similarity measure, i.e. $SC(\bar{c})_j = sim_{LIN}(\bar{c}, c_j)$.

wup Weighting using the Wu&Palmer similarity measure, i.e. $SC(\bar{c})_j = sim_{WUP}(\bar{c}, c_j)$.

The superconcept kernel $\kappa_{\mathcal{S}}$ can normalized to $[0, 1]$ in the usual way using the cosine normalization modifier.

3.4 Latent Semantic Term Kernels

The taxonomy kernel that has been described in the previous section appears to be an accurate indicator of similarity of terms. However, it requires the prior existence of a taxonomic background knowledge structure and the computation of the respective measures. As an alternative, work in the direction of latent semantic indexing (LSI) [10] investigated means for calculating term similarities by means of co-occurrence analysis of terms in documents and vice versa. We now shortly review the main technique of LSI.

Given a term-by-document matrix M, the singular value decomposition of M is given by $M = U\Sigma V'$ where Σ is a diagonal matrix with the same dimensionality as D containing the singular values in decreasing arrangement and U,V are orthogonal matrices. The columns of U are the singular vectors of the feature space corresponding to the respective singular value. A projection onto the first k dimensions is given by $U_k = I_k U$, where I_k is the identity matrix with all but the first k diagonal elements zero.

Definition 8. We can thus define the *latent semantic similarity kernel* [7] of terms t_i and t_j as $\kappa_{\mathcal{S}}^{LSI} = \langle U_k^i (U_k^j)' \rangle$ whereby U_k^i is the i-th (row) vector of the truncated matrix U_k.

Again, the kernel $\kappa_{\mathcal{S}}$ can be normalized to $[0, 1]$ in the usual way using the cosine normalization modifier.

4. EXPERIMENTS

In an extensive set of experiments we aimed at showing that our approach is effective for IR and text mining applications and at investigating how particular design choices affect performance. For this purpose, we ran experiments on two diverse datasets, each of which constitutes a comparatively "hard" text categorization problem, i.e. a categorization problem where the simple bag-of-words paradigm does not perform sufficiently well. The first experiment (cf. section 4.2) was conducted on a TREC question classification corpus while the second experiment (cf. section 4.3) was conducted on a corpus of short clinical free texts.

4.1 Experimental Setup

We have implemented the kernels introduced in section 3.1 within the SVM-light-TK software[1] which encodes tree kernel functions in SVM-light [12]. In all experiments, we either

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

		$\lambda = 0.05$		$\lambda = 0.01$		$\lambda = 0.005$		$\lambda = 0.001$	
		μF_1	A	μF_1	A	μF_1	A	μF_1	A
	linear (BOW)	81.5	**89.2**	81.5	**89.2**	81.5	**89.2**	81.5	**89.2**
$\alpha = 1$	default (string match)	89.8	**90.8**	89.7	**91.2**	89.7	**91.2**	89.7	**91.4**
	taxo-full	91.0	**92.6**	90.6	**92.6**	90.6	**92.6**	90.7	**92.6**
	taxo-full-ic	91.0	**92.6**	90.9	92.2	90.9	92.2	90.9	92.2
	taxo-lin	90.9	92.2	91.2	92.4	91.2	92.4	91.3	92.4
	taxo-wupalmer	91.0	**92.6**	91.2	**92.6**	91.2	**92.6**	91.3	**92.6**
	gvsm	89.5	90.8	89.5	91.2	89.5	91.0	89.5	90.8
	lsi-100	88.9	90.8	89.5	90.4	89.6	90.4	89.5	90.4
	lsi-50	89.3	**91.4**	89.8	**91.8**	89.9	**91.8**	89.9	**91.8**
$\alpha = 2$	default (string match)	89.7	**91.4**	89.8	**91.6**	89.9	**91.4**	90.0	**91.4**
	taxo-full	91.2	93.2	91.5	**93.0**	91.6	**93.0**	91.6	**93.0**
	taxo-full-ic	91.1	92.8	91.3	92.6	91.2	92.6	91.2	92.6
	taxo-lin	91.1	92.4	91.4	92.6	91.2	92.6	91.5	92.6
	taxo-wupalmer	91.6	**93.6**	91.5	**93.0**	91.4	92.8	91.4	92.8
	gvsm	89.7	91.2	89.5	91.4	89.5	91.4	89.6	91.4
	lsi-100	88.9	91.2	90.0	91.0	90.0	90.8	90.0	91.0
	lsi-50	89.8	**91.8**	89.9	**92.0**	90.0	**92.2**	90.1	**92.2**

Table 1: Evaluation of different kernels on the question classification dataset for different values of α, different values of λ and different semantic smoothing kernels. Evaluation results for the linear kernel based on the bag-of-words representation are included as baseline (obviously, the settings of λ don't apply here). For each combination, we report micro-averaged F-measure (μF_1) and accuracy (A). All numbers are percentages. The best accuracy for each class of kernels is highlighted.

used the noun hierarchy of WordNet[2] as the underlying semantic network for calculating topological term similarity kernels[3] or standard LSI for LSI-based term similarity kernels. Kernel similarities that were undefined because of a missing mapping to a noun synset or because they were not included in the term-document matrix for LSI (e.g. stopwords) were implicitly assumed to take the default values (i.e. zero for distinct and one identical terms respectively).

4.2 Question Classification

In the first set of experiment, we evaluate different types of kernels on a dataset from the Question Classification (QC) domain. The long tradition of Question Answering in TREC has produced a large question set used by several researchers which can be exploited for experiments on Question Classification.

Question Classification [13] aims at detecting the type of a question, e.g. whether it asks for a person or for an organization which is critical to locate and extract the right answers in question answering systems. According to [13], we can define question classification *"to be the task that, given a question, maps it to one of k classes, which provide a semantic constraint on the sought-after answer"*.

A major challenge of Question Classification compared to standard Text Classification settings is that questions typically contain extremely few words which make this setting a typical victim of data sparseness. Previous work has thus shown that Semantic Smoothing Kernels are capable of improving the effectiveness in QC tasks. On the other hand, questions have a specific syntactic structure and plain tree kernels have shown to be effective as well.

We consider the same dataset and classification problem as introduced in [13, 23]. The dataset consists of free text questions and is freely available[4]. It is divided into 5,500 questions[5] for training and 500 questions from TREC 10 for testing. Each of these questions is labeled with exactly one class of the *coarse grained* classification scheme (see [23]) consisting of the following 6 classes: Abbreviations, Descriptions (e.g. *definition* and *manner*), Entity (e.g. *animal*, *body* and *color*), Human (e.g. *group* and *individual*), Location (e.g. *city* and *country*) and Numeric (e.g. *code* and *date*). In these experiments, we used the same experimental setup as used in [23] as it contains the most comprehensive comparison of experiments on the QC corpus introduced above.

We compared the linear kernel based on bag-of-words[6], the original tree kernel and a set of semantic syntactic tree kernel configurations as introduced in Section 3 with different term similarities (i.e. normalized taxonomic and latent semantic kernels, where the LSA matrix is obtained on the question dataset).

The question parse trees were obtained by running Char-

[2]`http://wordnet.princeton.edu/`

[3]For word sense disambiguation, we used a simplifying assumption in mapping each term to its most frequent noun sense (if it exists).
Note that this approach implies an inherent word sense disambiguation side effect, likely to have a negative impact on the results. The results can also be seen as a pessimistic estimate.

[4]`http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`

[5]Selected from the 4500 English questions published by USC (Hovy et al., 2001), 500 questions annotated for rare classes and the 894 questions from TREC 8 and TREC 9.

[6]To allow for a fair comparison, we used a modified preprocessing which does *not* remove any stopwords as function words like "What" or "Who" are highly informative for the question classification task.

	linear kernel (BOW)			tree kernel, $\lambda = 0.01$			tree kernel, LSI50 ,$\lambda = 0.01$		
class	P	R	F_1	P	R	F_1	P	R	F_1
786-2	91.82	92.86	92.34	96.73	100.00	98.34	96.38	100.00	98.16
780-6	84.06	84.67	84.36	97.84	99.27	98.55	97.84	99.27	98.55
599-0	81.03	83.19	82.10	91.67	97.35	94.42	93.28	98.23	95.69
593-70	92.31	93.20	92.75	98.10	100.00	99.04	98.10	100.00	99.04
591	87.01	84.81	85.90	100.00	97.47	98.72	100.00	97.47	98.72
486	81.94	84.29	83.10	98.55	97.14	97.84	98.55	97.14	97.84
596-54	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
786-07	89.58	100.00	94.50	100.00	100.00	100.00	100.00	100.00	100.00
593-89	77.50	72.09	74.70	100.00	97.67	98.82	100.00	97.67	98.82
599-7	91.67	94.29	92.96	100.00	100.00	100.00	100.00	100.00	100.00
788-30	93.55	85.29	89.23	100.00	97.06	98.51	100.00	94.12	96.97
786-50	86.49	94.12	90.14	97.14	100.00	98.55	97.14	100.00	98.55
493-90	82.35	60.87	70.00	100.00	91.30	95.45	100.00	91.30	95.45
V13-02	27.78	22.73	25.00	100.00	63.64	77.78	100.00	59.09	74.28
795-5	100.00	88.24	93.75	100.00	100.00	100.00	100.00	100.00	100.00
518-0	45.45	31.25	37.04	100.00	100.00	100.00	100.00	100.00	100.00
753-3	66.67	37.50	48.00	100.00	93.75	96.77	100.00	93.75	96.77
277-00	93.33	93.33	93.33	100.00	100.00	100.00	100.00	100.00	100.00
741-90	45.45	33.33	38.46	100.00	80.00	88.89	100.00	86.67	92.86
759-89	90.91	90.91	90.91	100.00	100.00	100.00	100.00	100.00	100.00
Macro-av.	80.44	76.35	78.34	99.00	95.73	97.34	99.06	95.74	97.37

Table 2: Evaluation of different kernels on 20 classes of the CMC dataset. For each class we report precision (P), recall (R) and F-measure (F_1). All numbers are percentages.

niak's parser [7]. We trained binary classifiers on each of the 6 classes. We preliminary selected the best cost-factor (parameter j) on the validation set and then experimented with different λ values. In all experiments, we used $c = 1$ as soft-margin parameter. The binary classifiers are then combined in a multiclassification scheme, which always selects the single class for which the test instance produces the highest margin score of the binary SVMs.

For all cases, we report the *micro-averaged* F_1, i.e. the harmonic mean of micro-averaged precision and recall, and the multi-classifier accuracy (since only one class should be assigned to a question).

Table 1 reports the results of the experiments[8]. The first column indicates the value of α (see section 3.2). The second column shows the type of term similarity kernel used together with the tree kernel function, where *string matching* means that the original tree kernel is used. The remaining columns report the micro-averaged F_1 and the multiclassification accuracy.

While the variation of the λ parameter seems to have a minor importance, the improvement of the tree kernels seems to be largely consistent. We can note that default tree kernels can achieve up to 91.6% of multiclassification accuracy (for $\alpha = 2$).

The above values can be improved considerably when we employ term similarity kernels. For the case of LSI-50, i.e. by truncating the U matrix at $k = 50$ components (see section 3.4), we can note a slight improvement of up to 92.2%.

If we use the generalized vector space model (GVSM) defined in [7], i.e. when all the U components are selected, or LSI-100, i.e. $k = 100$, the results are similar or slightly worse than the default tree kernels.

For the case of taxonomy-driven term similarity kernels, the addition of semantic information always improves performance whereby the improvement appears to be the highest for the "Full" and "Wu-Palmer" schemes. In particular, the tree kernel based on the "Wu-Palmer" similarity, $\alpha = 2$ and $\lambda = 0.05$ achieves the highest multiclassification accuracy, i.e. 93.6%. This is a great result as it improves (1) previous work on question classification using tree kernels, i.e. 90% in [23] and (2) the results obtained in [13], i.e. 92.5%, using many features and semantic resources manually annotated for such question dataset.

Regarding the α parameter, in general, we have noted that other values do not improve the results any further.

4.3 Classifying Clinical Free Text

In the second experiment, we worked on a dataset released by the Center of Computational Medicine in 2007[9]. The dataset consists of 978 data items composed of two text fields in which medical doctors report on (i) the clinical history of a patient and (ii) the impression gained after radiological examination of the patient. In this context, the categorization task is to automate the assignment of diseases according to the ICD-9-CM classification scheme to the data instances.

[7]`ftp://ftp.cs.brown.edu/pub/nlparser/`

[8]Note that our results for the bag-of-words kernel differ slightly from literature results, e.g. 90.0% of [23] which we suspect to be due to a different preprocessing

[9]`http://www.computationalmedicine.org/challenge`

Each of the data items is assigned to one up to three codes out of a pool of 45 ICD-9-CM codes. We performed binary classification experiments on the 20 largest categories.

As there is no official test data set with target values released yet, we computed leave-one-out estimates of the performance of three different kernels:

(i) the simple bag of words kernel (with terms indexed separately on the history and impression parts),

(ii) the combination of bow and (normalized) tree forest kernels for each of the history and impression text snippets and

(iii) the same but with the contribution of an term similarity kernel based on LSI with $k = 50$[10].

The forest kernel is defined over multiple parse trees coming from different sentences describing the history of a patient and the impression from the radiological examination. More formally, it is defined as follows:

$$K_{\texttt{all}}(S_1, S_2) = \sum_{T_1 \in P_1} \sum_{T_2 \in P_2} K_T(T_1, T_2), \qquad (1)$$

where S_1 and S_2 are the set of sentences of two instances (each instance both includes history and impression), P_1 and P_2 are the corresponding sets of parse trees and K_T is the semantic syntactic tree kernel defined by Eq. 1.

Table 2 gives a detailed account of the results for all categories. We note that the plain tree kernels give an impressive improvement over the bag-of-words results, both in terms of precision and recall. In several cases it is possible to achieve precision and recall values of 100.00%. Consequently, the additional improvement achieved by the semantic smoothing is marginal.

5. DISCUSSION AND CONCLUSION

Lexical semantic kernels were initially introduced in [19] using inverted path length as a similarity measure and subsequently revisited in [7, 1], each time based on different design principles. Semantic kernels based on superconcept representations were investigated in [14] and [2]. As an alternative, [7] have put Semantic Kernels into the context of Latent Semantic Indexing.

Tree Kernels were firstly introduced in [5] and experimented with the Voted Perceptron for the parse-tree re-ranking task. The combination with the original PCFG model improved the syntactic parsing. In [22], two kernels over syntactic shallow parser structures were devised for the extraction of linguistic relations, e.g. *person-affiliation*. To measure the similarity between two nodes, the Contiguous String Kernel and the Sparse String Kernel were used. In [8] such kernels were slightly generalized by providing a matching function for the node pairs. The time complexity for their computation limited the experiments on a data set of just 200 news items. In [9], a feature description language was used to extract structural features from the syntactic shallow parse trees associated with named entities. The experiments on named entity categorization showed that too many irrelevant tree fragments may cause overfitting. In [15] Tree Kernels were firstly proposed for semantic role classification. The combination between such kernel and a polynomial kernel of standard features improved the state-of-the-art.

In this paper, we have investigated how the syntactic structures of natural language texts can be exploited simultaneously with semantic background knowledge on term similarity. For this purpose, we have proposed a new family of kernels called Semantic Syntactic Tree Kernels (SSTKs) that is based on Tree and Semantic Smoothing Kernels. We have motivated this class of kernels by counting all compatible tree fragments of two parse trees weighted by their joint terminology. To our knowledge, no other work has so far combined the syntactic and semantic properties of natural language in such a principled way.

We conducted a series of experiments on the TREC question classification data. Our new Syntactic Semantic Tree Kernel improves the state-of-the-art in Question Classification, which makes it a prototype of a possible future full-fledged natural language kernel. Our results indicate that the newly proposed Semantic Syntactic Tree Kernels outperform the conventional linear/semantic kernels as well as tree kernels improving the state of the art in Question Classification. For the case of CMC experiments, tree kernels appear to be highly successful compared to BOW, such that an additional improvement based on an LSI term kernel is hardly noticeable. For future work, we envision to use semantic smoothing in combination with different tree kernel paradigms such the variant proposed in [21, 16].

[10] As the corpus is to small for a meaningful computation of the LSI term similarities, we relied on a different medical corpus, namely the 1987 portion of the Ohsumed dataset for this experiment.

6. REFERENCES

[1] R. Basili, M. Cammisa, and A. Moschitti. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 1–8, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[2] S. Bloehdorn, R. Basili, M. Cammisa, and A. Moschitti. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 06), Hong Kong, 18-22 December 2006*, DEC 2006.

[3] S. Bloehdorn and A. Moschitti. Combined syntactic and semantic kernels for text classification. In G. Amati, C. Carpineto, and G. Romano, editors, *Advances in Information Retrieval - Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007), Rome, Italy*, volume 4425 of *Lecture Notes in Computer Science*, pages 307–318. Springer, APR 2007.

[4] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47,

March 2006.

[5] M. Collins and N. Duffy. Convolution kernels for natural language. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14: Neural*, pages 625–632. MIT Press, 2001.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.

[7] N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent Semantic Kernels. *Journal of Intelligent Information Systems*, 18(2-3):127–152, 2002.

[8] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of ACL'04*, 2004.

[9] C. Cumby and D. Roth. Kernel methods for relational learning. In *Proceedings of the Twentieth International Conference (ICML 2003)*, 2003.

[10] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.

[11] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[12] T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[13] X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, 2002.

[14] D. Mavroeidis, G. Tsatsaronis, M. Vazirgiannis, M. Theobald, and G. Weikum. Word sense disambiguation for exploiting hierarchical thesauri in text classification. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *Knowledge Discovery in Databases: Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2005), Porto, Portugal, October 3-7, 2005*, pages 181–192. Springer, 2005.

[15] A. Moschitti. A study on convolution kernels for shallow semantic parsing. In *proceedings of ACL*, 2004.

[16] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006*, 2006.

[17] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[18] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, June 2004.

[19] G. Siolas and F. d'Alche Buc. Support vector machines based on a semantic kernel for text categorization. In *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, volume 5, pages 205–209, 2000.

[20] V. Vapnik, S. E. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation and signal processing. In *NIPS*, pages 281–287, 1996.

[21] S. V. N. Vishwanathan and A. J. Smola. Fast kernels for string and tree matching. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (Neural*, pages 569–576. MIT Press, 2003.

[22] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 2003.

[23] D. Zhang and W. S. Lee. Question classification using support vector machines. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32, New York, NY, USA, 2003. ACM Press.