# Combined Syntactic and Semantic Kernels for Text Classification

Stephan Bloehdorn[1] and Alessandro Moschitti[2]

[1] Institute AIFB, University of Karlsruhe, Germany
bloehdorn@aifb.uni-karlsruhe.de
[2] University of Rome 'Tor Vergata', Italy
moschitti@info.uniroma2.it

**Abstract.** The exploitation of syntactic structures and semantic background knowledge has always been an appealing subject in the context of text retrieval and information management. The usefulness of this kind of information has been shown most prominently in highly specialized tasks, such as classification in Question Answering (QA) scenarios. So far, however, additional syntactic or semantic information has been used only individually. In this paper, we propose a principled approach for jointly exploiting both types of information. We propose a new type of kernel, the Semantic Syntactic Tree Kernel (SSTK), which incorporates linguistic structures, e.g. syntactic dependencies, and semantic background knowledge, e.g. term similarity based on WordNet, to automatically learn question categories in QA. We show the power of this approach in a series of experiments with a well known Question Classification dataset.

## 1 Introduction

Text Classification (TC) systems [1], which aim at automatically classifying textual input into categories according to some criterion of interest are a major application domain of modern machine learning techniques. Pioneered by [2], Support Vector Machines (SVMs) have been heavily used for text classification tasks, typically showing good results. In the simplest case, such systems use the standard *bag-of-words* feature representation which encodes the input 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, several modifications to this rather flat representation have been shown to improve the overall performance in selected scenarios. In particular, there has been interest in incorporating information about (i) the syntactic structure of the input texts and (ii) the semantic dependencies within the used terminology.

Kernel-based learning algorithms like Support Vector Machines have become a prominent framework for using such a-priori knowledge about the problem domain by means of a specific choice of the employed kernel function. On the one

hand, *Tree Kernels* [3,4] have been used as a powerful way to encode the syntactic structure of the textual input in the form of parse trees and have shown good results in many natural language applications. On the other hand, *Semantic Smoothing Kernels* [5,6,7] exploit background information from semantic reference structures such as WordNet to make different, though semantically similar, terms contribute to the overall similarity of the input instances. Intuitively, the power of this kind of kernels is most apparent when too little training data is available to build stable models by counting occurrences of identical terminology only. While both approaches seem intuitive and powerful, language draws from both syntax and lexical semantics, therefore, finding principled techniques for combining kernels on linguistic structures, e.g. Tree Kernels, with Semantic Kernels appears to be a promising research line.

In this paper, we propose a new type of kernel, the *Semantic Syntactic Tree Kernel (SSTK)*, which exploits linguistic structure and background knowledge about the semantic dependencies of terms at the same time. More technically, the proposed kernel uses semantic smoothing to improve the matching of tree fragments containing terminal nodes.

We show the power of this approach in a series of experiments from the Question Classification (QC) domain. Question Classification [8] 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. A major challenge of Question Classification compared to standard Text Classification settings is that questions typically contain only extremely few words which makes this setting a typical victim of data sparseness. Previous work has shown that Tree Kernels as well as Semantic Smoothing Kernels were individually capable of improving effectiveness in QC tasks. Our evaluation studies confirm these findings and indicate a consistent further improvement of the results when the proposed combined kernel is used. 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.

The remainder of this paper is structured as follows. Section 2 introduces kernel methods and some related work. Sections 3, 4 and 5 describe the design of the Semantic Smoothing Kernels, Tree Kernels and the combined Semantic Syntactic Tree Kernels, respectively. Section 6 gives an account on the performance of these in a series of evaluation experiments from the QC domain. We conclude in section 7 with a summary of the contributions, final remarks and a discussion of envisioned future work.

## 2 Kernel Methods and Related Work

Support Vector Machines [9] are state-of-the-art learning methods based on the older idea of linear classification. The distinguishing feature of SVMs is the theoretically well motivated and efficient training strategy for determining the separating hyperplane based on the margin maximization principle. In our context, however, the interesting property of SVMs is their capability of naturally

incorporating data-specific notions of item similarity by means of a corresponding kernel function. Formally, any function $\kappa$ 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 $\phi$ is a suitable mapping from $\mathcal{X}$ to a feature (Hilbert-) space $\mathcal{F}$. Kernels can be designed by either choosing an explicit mapping function $\phi$ and incorporating it into an inner product or by directly defining the kernel function $\kappa$ 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 from known valid kernels. In particular, kernels are closed under sum, product, multiplication by a positive scalar and combination with well-known kernel modifiers. In particular, a given kernel $\kappa$ 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. [10] for a comprehensive introduction.

Lexical semantic kernels were initially introduced in [5] using inverted path length as a similarity measure and subsequently revisited in [11,12], each time based on different design principles. Semantic kernels based on superconcept representations were investigated in [6] and [7]. As an alternative, [11] have put Semantic Kernels into the context of Latent Semantic Indexing.

Tree Kernels were firstly introduced in [3] 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 [13], 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 [14] 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 [15], 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 [16] 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.

To our knowledge, no other work has so far combined the syntactic and semantic properties of natural language in a principled way as proposed in our approach.

## 3 Semantic Similarity Kernels

In this section we describe the first component of our new kernel, the Semantic Smoothing Kernel, which combines semantic similarity of terms with the standard bag-of-words representation.

Inverted Path Length:

$$sim_{IPL}(c_1, c_2) = \frac{1}{(1 + d(c_1, c_2))^{\alpha}}$$

Wu & Palmer:

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

Resnik:

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

Lin:

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

**Table 1.** Measures of semantic similarity.

### 3.1 Semantic Networks and Similarity

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. To compute useful notions of semantic similarity among the input terms, we employ semantic reference structures which we call, for simplicity, *Semantic Networks*. These can be seen 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.

The measures relevant in the context of this paper are summarized in table 1. These measures 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 $c_2$. (ii) The *depth* (*dep*) of a concept refers to the distance of the concept to the unique root node[3]. (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 [17] for a detailed and recent survey of the field.

[3] If the structure is not a perfect tree structure, we use the minimal depth.

| | |
|---|---|
| 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. $SC(\bar{c})_j = sim_{RES}(\bar{c}, c_j)$. |
| path-1 | Weighting based on inverted path length, i.e. $SC(\bar{c})_j = sim_{IPL}(\bar{c}, c_j)$ for all superconcepts $c_j$ of $\bar{c}$ and $SC(\bar{c})_j = 0$ otherwise using the parameter $\alpha = 1$. |
| path-2 | The same but using the parameter $\alpha = 2$. |
| 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. $SC(\bar{c})_j = sim_{WUP}(\bar{c}, c_j)$. |

**Table 2.** Weighting Schemes for the Superconcept Kernel $\kappa_{\mathcal{S}}$

### 3.2 Semantic Similarity Kernels based on Superconcepts

In this section, we introduce a class of kernel functions defined on terms that can be embedded in other kernels that make (in whatever way) use of term matching[4]

**Definition 1 (Superconcept Kernel).** *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 (super-)concepts present in the employed semantic network and the respective entries are determined by a particular weighting scheme.*

This idea, recently investigated in [7], is based on the observation that the more two concepts are similar the more they share common superconcepts. A similar approach has been proposed in [6], however focusing on the simple case of giving the superconcepts in the mapping full and equal weight while varying the number of superconcepts that are considered.

Obviously, $\kappa_{\mathcal{S}}$ is a valid kernel as 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. In the same way as in earlier work [7], we have investigated the use of different weighting schemes for the representation of the superconcepts motivated by the following considerations:

1. The weight a superconcept $SC(\bar{c})_j$ receives in the vectorial description of concept $\bar{c}$ should be influenced by its distance from $\bar{c}$.
2. 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.

[4] For simplicity, we will restrict our attention on defining kernel functions for *concepts* and leave the details of the consideration of lexical ambiguity to the next section.

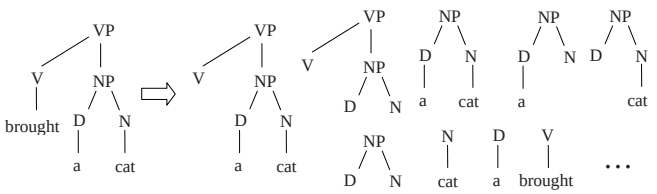


**Fig. 1.** A tree with some of its fragments.

We have used the measures of semantic similarity introduced in table 1 as weighting schemes, summarized in table 2. The different weighting schemes behave differently wrt the above motivations. While full does not implement any of them, full-ic considers rationale 2 while path-1 and path-2 consider rationale 1. The schemes lin and wup reflect combinations of both rationales. The superconcept kernel $\kappa_{\mathcal{S}}$ can normalized to $[0, 1]$ in the usual way using the cosine normalization modifier.

The concept kernel $\kappa_{\mathcal{S}}(\cdot, \cdot)$ can be used directly in conjunction with the standard linear kernel by means of a simple *Semantic Smoothing Kernel*.

**Definition 2 (Semantic Smoothing Kernel).** *The Semantic Smoothing Kernel $\kappa_{\mathcal{S}}$ for two term vectors (input texts) $x, z \in X$ is given by $\kappa_{\hat{\mathcal{S}}}(x, z) = x'C'K_{\mathcal{S}}Cz$ where $K_{\mathcal{S}}$ is a square symmetric matrix whose entries represent the kernel evaluations between the concepts and $C$ denotes the matrix that encodes the evaluations of the disambiguation function $C$ that maps concept dimensions to term dimensions that constitute the input space $X$.*

## 4 Tree Kernels for Syntactic Structures

The main rationale behind Tree Kernels is to represent trees in terms of their substructures (fragments). The kernel function then counts the number of tree subparts common to both argument trees. 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 the execution of 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 in section 4 shows a parse tree of the sentence fragment `‘‘brought a cat’’` with some of its substructures.

**Definition 3 (Tree Kernel (Collins & Dufy, 2001)).** *Given two trees $T_1$ and $T_2$ we define the (Subset-) 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* $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$, *and where* $I_i(n)$ *is an indicator function which determines whether fragment* $f_i$ *is rooted in node* $n$.

$\Delta$ is equal to the number of common fragments rooted at nodes $n_1$ and $n_2$. We can compute it more efficiently as follows:

1. if the productions at $n_1$ and $n_2$ are different then $\Delta(n_1, n_2) = 0$;
2. if the productions at $n_1$ and $n_2$ are the same, and $n_1$ and $n_2$ 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

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

where $nc(n_1)$ is the number of children of $n_1$ and $ch_n^j$ 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 $\lambda$ can be added by modifying steps (2) and (3) as follows:

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

As an example, Figure 1 shows a parse tree of the sentence (fragment) ``bought a cat'' with some of the substructures that the tree kernel uses to represent it[5].

## 5 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 structures but employ different though related terminology at the leaves, they will not be matched. From a semantic point of view, this is an evident drawback as ``brought a cat'' should be more related to ``brought a tomcat'' than to ``brought a note''.

In analogy with the semantic smoothing kernels for the bag-of-words kernel as described in section 3.2, we are now interested in also counting *partial matches* between tree fragments. A partial match occurs when two fragments differ only by their terminal symbols, e.g. [N [cat]] and [N [tomcat]]. In this case the match should give a contribution smaller than 1, depending on the semantic similarity of the respective terminal nodes. For this purpose, we first define the similarity of two such tree fragments.

[5] The number of such fragments can be obtained by evaluating the kernel function between the tree with itself.

**Definition 4 (Tree Fragment Similarity Kernel).** *For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the Tree Fragment Similarity Kernel as*[6]*:*

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

*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).*

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 3 as in the following definition.

**Definition 5 (Semantic Syntactic Tree Kernel).** *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 $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} I_i(n_1) I_j(n_2) \kappa_{\mathcal{F}}(f_i, f_j)$.*

Obviously, 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 be evaluated in the same way as the standard tree kernel by adding the following step

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

as the first condition of the $\Delta$ function definition (Section 4), where $label(n_i)$ is the label of node $n_i$ and $\kappa_{\mathcal{S}}$ is a term similarity kernel, e.g. based on the superconcept kernel defined in section 3.2. Note that: (a) since $n_1$ and $n_2$ are pre-terminals of a parse tree they can have only one child (i.e. $ch^1_{n_1}$ and $ch^1_{n_2}$) and such children are words and (b) Step 2 is no longer necessary.

Beside the novelty of taking into account tree fragments that are not identical it should be noted that the lexical semantic similarity is constrained in syntactic structures, which limit errors/noise due to incorrect (or, as in our case, not provided) word sense disambiguation.

[6] Note that, as the tree fragments need to be compatible, they have the same number of terminal symbols at compatible positions.

## 6 Experimental Evaluation

In a series of experiments we aimed at showing that our approach is effective for IR and text mining applications. For this purpose, we experimented with the TREC question classification corpus for advanced retrieval based on the Question Answering paradigm.

### 6.1 Experimental setup

The long tradition of QA in TREC has produced a large question set used by several researchers which can be exploited for experiments on Question Classification. According to [8], 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"*. Such questions are categorized according to different taxonomies of different *grains*. We consider the same dataset and classification problem as described in [18,8]. The dataset consists of free text questions and is freely available[7]. It is divided into 5,500 questions[8] for training and the 500 TREC 10 questions for testing. Each of these questions is labeled with exactly one class of the *coarse grained* classification scheme (see [18]) 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*).

We have implemented the kernels introduced in sections 3–5 within the SVM-light-TK software available at `ai-nlp.info.uniroma2.it/moschitti/` which encodes tree kernel functions in SVM-light [19]. In all experiments, we used the noun hierarchy of WordNet[9] as the underlying semantic network. 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. Kernel similarities that were undefined because of a missing mapping to a noun synset were implicitly assumed to take the default values (i.e. zero for distinct and one identical terms respectively).

### 6.2 Evaluation of Superconcept Smoothing Kernels

In a first experiment we investigated the effect of simply smoothing term features based on the idea of the Semantic Smoothing Kernel. Questions were preprocessed using the usual steps, namely tokenization, lemmatization and stoopword-removal leading to a total number of 8,075 distinct TFIDF-weighted features.

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

[8] These are 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.

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

| macro-averaging | | | | | | |
|---|---|---|---|---|---|---|
| | soft margin parameter $c$ | | | | | |
| **kernel** | **0.1** | **0.2** | **0.3** | **1.0** | **2.0** | **3.0** |
| **linear** | 0.21 | 0.38 | 0.47 | 0.62 | 0.63 | 0.64 |
| **full** | 0.38 | 0.49 | 0.55 | 0.61 | 0.61 | 0.68 |
| **full-ic** | **0.53** | **0.53** | 0.53 | 0.62 | 0.55 | 0.55 |
| **path-1** | 0.25 | 0.42 | 0.51 | **0.64** | 0.64 | 0.64 |
| **path-2** | 0.22 | 0.39 | 0.47 | 0.63 | **0.65** | 0.64 |
| **lin** | 0.36 | 0.49 | **0.56** | **0.64** | 0.62 | **0.70** |
| **wup** | 0.34 | 0.49 | 0.54 | 0.62 | 0.61 | 0.69 |

| micro-averaging | | | | | | |
|---|---|---|---|---|---|---|
| | soft margin parameter $c$ | | | | | |
| **kernel** | **0.1** | **0.2** | **0.3** | **1.0** | **2.0** | **3.0** |
| **linear** | 0.09 | 0.25 | 0.34 | 0.55 | 0.57 | 0.58 |
| **full** | 0.27 | 0.38 | 0.45 | 0.55 | 0.56 | 0.68 |
| **full-ic** | **0.47** | **0.46** | **0.47** | **0.60** | 0.49 | 0.48 |
| **path-1** | 0.14 | 0.32 | 0.40 | 0.57 | 0.58 | 0.59 |
| **path-2** | 0.08 | 0.28 | 0.37 | 0.57 | **0.59** | 0.58 |
| **lin** | 0.27 | 0.37 | **0.47** | 0.57 | 0.57 | **0.69** |
| **wup** | 0.23 | 0.37 | 0.45 | 0.56 | 0.56 | 0.68 |

**Table 3.** Absolute macro and micro F1 results for QC, for different values of $c$ and different semantic smoothing kernels. The best results per setting of $c$ are highlighted

We performed *binary classification experiments* on each of the 6 question types for different settings of the 'soft margin' parameter $c$. Table 3 summarizes the absolute macro $F_1$ as well as the micro $F_1$ values obtained in the question classification setting. The best values per setting of $c$ are highlighted.

The results indicate that the smoothing of the term features considerably affects the performance compared to the simple linear kernel baseline. According to the results, the lin scheme seems to achieve the best overall performance with a relative improvement of 9.32% for the macro $F_1$ value in the case of $c = 3$ (i.e. the setting for which the linear kernel achieves its maximum). For a more detailed description of these experiments, refer to [7]. For QC, however, the traditional bag-of-words (bow) appears to be somewhat insufficient as it ignores structure and function words that are likely to influence the results.

### 6.3 Evaluation of Syntactic Semantic Tree Kernels

In these experiments, we used the same experimental setup as used in [18] as it contains the most comprehensive comparison of experiments on the QC corpus introduced above. As literature results are given in terms of the accuracy of the *multi-classification* of the TREC questions, to compare with them, we designed a multiclassifier based on the scores of the SVMs. For each questions, we selected the class associated with the maximum score.

In this experiment, we compared the linear kernel based on bag-of-words (this time including function words), the original STK and the new SSTK as introduced in Section 5 with different term similarities[10]. The question parse trees were obtained by running the Charniak's parser. Table 4 reports the results of the multiclassifier based on SSTK. Column 1 shows the type of similarity used in the semantic syntactic tree kernel function, where *string matching* means that

[10] We again used the superconcept kernels as term similarities, however, in contrast to the previous section we used normalized versions.

the original tree kernel is used. Columns from 2 to 6 report the accuracy of the multiclassifier according to several values of the $\lambda$ parameter[11] (see Section 5).

| | **Accuracy** | | | | |
|---|---|---|---|---|---|
| **$\lambda$ parameter** | **0.4** | **0.05** | **0.01** | **0.005** | **0.001** |
| **linear (bow)** | 0.905 | | | | |
| **string matching** | 0.890 | 0.910 | **0.914** | **0.914** | 0.912 |
| **full** | 0.904 | **0.924** | 0.918 | 0.922 | 0.920 |
| **full-ic** | 0.908 | **0.922** | 0.916 | 0.918 | 0.918 |
| **path-1** | 0.906 | **0.918** | 0.912 | **0.918** | 0.916 |
| **path-2** | 0.896 | 0.914 | 0.914 | **0.916** | **0.916** |
| **lin** | 0.908 | **0.924** | 0.918 | 0.922 | 0.922 |
| **wup** | 0.908 | **0.926** | 0.918 | 0.922 | 0.922 |

**Table 4.** Accuracy of SSTK on QC, for different values of $\lambda$ parameter and different semantic smoothing kernels. The best $\lambda$ settings are highlighted

We note that (a) our basic Syntactic Tree Kernel improves the state-of-the-art accuracy, i.e. 91.4% vs. 90% of [18], (b) this is further improved when we use one of the semantic smoothing kernel and (c) the Wu-Palmer similarity achieves the highest accuracy, i.e. 92.6%.

## 7 Conclusion

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 (SSTK) 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 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. In the future, it would be interesting to study different models of lexical semantics, e.g. latent semantic kernels, with kernels based on different syntactic/semantic structures.

[11] We preliminary selected the best cost-factor (parameter j) on the validation set and then experimented with different $\lambda$ values. We also noted that the parameter $c$ is not critical when tree kernels are used. This means that the accuracy does not vary too much and the highest value seems to be achieved with the default SVM-light setting.

## References


1. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34**(1) (2002) 1–47
2. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Proceedings of ECML, Chemnitz, DE (1998)
3. Collins, M., Duffy, N.: Convolution kernels for natural language. In: NIPS, MIT Press (2001)
4. Moschitti, A.: Efficient convolution kernels for dependency and constituent syntactic trees. In: Proceedings of ECML, Berlin, Germany. (2006)
5. Siolas, G., d'Alche Buc, F.: Support Vector Machines based on a semantic kernel for text categorization. In: IJCNN. Volume 5. (2000)
6. Mavroeidis, D., Tsatsaronis, G., Vazirgiannis, M., Theobald, M., Weikum, G.: Word sense disambiguation for exploiting hierarchical thesauri in text classification. In: PKDD. (2005)
7. Bloehdorn, S., Basili, R., Cammisa, M., Moschitti, A.: Semantic kernels for text classification based on topological measures of feature similarity. In: Proceedings of ICDM. (2006)
8. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th International Conference on Computational Linguistics (COLING). (2002)
9. Vapnik, V., Golowich, S.E., Smola, A.J.: Support vector method for function approximation, regression estimation and signal processing. In: NIPS. (1996)
10. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
11. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent Semantic Kernels. Journal of Intelligent Information Systems **18**(2-3) (2002) 127–152
12. Basili, R., Cammisa, M., Moschitti, A.: A semantic kernel to exploit linguistic knowledge. In: AI*IA: Advances in Artificial Intelligence. (2005)
13. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. Journal of Machine Learning Research (2003)
14. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. In: Proceedings of ACL. (2004)
15. Cumby, C., Roth, D.: Kernel methods for relational learning. In: Proceedings of the Twentieth International Conference (ICML 2003). (2003)
16. Moschitti, A.: A study on convolution kernels for shallow semantic parsing. In: proceedings of ACL. (2004)
17. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics **32**(1) (2006) 13–47
18. Zhang, D., Lee, W.S.: Question classification using Support Vector Machines. In: Proceedings of SIGIR. (2003)
19. Joachims, T.: Making large-scale SVM learning practical. In: Advances in Kernel Methods. (1999)