Syntactic and Semantic Kernels for Short Text Pair Categorization

Alessandro Moschitti
Department of Computer Science and Engineering
University of Trento
Via Sommarive 14
38100 POVO (TN) - Italy
moschitti@disi.unitn.it

Abstract

Automatic detection of general relations between short texts is a complex task that cannot be carried out only relying on language models and bag-of-words. Therefore, learning methods to exploit syntax and semantics are required. In this paper, we present a new kernel for the representation of shallow semantic information along with a comprehensive study on kernel methods for the exploitation of syntactic/semantic structures for short text pair categorization. Our experiments with Support Vector Machines on question/answer classification show that our kernels can be used to greatly improve system accuracy.

1 Introduction

Previous work on Text Categorization (TC) has shown that advanced linguistic processing for document representation is often ineffective for this task, e.g. (Lewis, 1992; Furnkranz et al., 1998; Allan, 2000; Moschitti and Basili, 2004). In contrast, work in question answering suggests that syntactic and semantic structures help in solving TC (Voorhees, 2004; Hickl et al., 2006). From these studies, it emerges that when the categorization task is linguistically *complex*, syntax and semantics may play a relevant role. In this perspective, the study of the automatic detection of relationships between short texts is particularly interesting. Typical examples of such relations are given in (Giampiccolo et al., 2007) or those holding between question and answer, e.g. (Hovy et al., 2002; Punyakanok et al., 2004; Lin and Katz, 2003), i.e. if a text fragment correctly responds to a question.

In Question Answering, the latter problem is mostly tackled by using different heuristics and classifiers, which aim at extracting the best answers (Chen et al., 2006; Collins-Thompson et al., 2004). However, for definitional questions, a more effective approach would be to test if a *correct relationship* between the answer and the query holds. This, in turns, depends on the structure of the two text fragments. Designing language models to capture such relation is too complex since probabilistic models suffer from (i) computational complexity issues, e.g. for the processing of large bayesian networks, (ii) problems in effectively estimating and smoothing probabilities and (iii) high sensitiveness to irrelevant features and processing errors. In contrast, discriminative models such as Support Vector Machines (SVMs) have theoretically been shown to be robust to noise and irrelevant features (Vapnik, 1995). Thus, partially correct linguistic structures may still provide a relevant contribution since only the relevant information would be taken into account. Moreover, such a learning approach supports the use of kernel methods which allow for an efficient and effective representation of structured data.

SVMs and Kernel Methods have recently been applied to natural language tasks with promising results, e.g. (Collins and Duffy, 2002; Kudo and Matsumoto, 2003; Cumby and Roth, 2003; Shen et al., 2003; Moschitti and Bejan, 2004; Culotta and Sorensen, 2004; Kudo et al., 2005; Toutanova et al., 2004; Kazama and Torisawa, 2005; Zhang et al., 2006; Moschitti et al., 2006). In particular, in question classification, tree kernels, e.g. (Zhang and Lee, 2003), have shown accuracy comparable to the best models, e.g. (Li and Roth, 2005).

Moreover, (Shen and Lapata, 2007; Moschitti et al., 2007; Surdeanu et al., 2008; Chali and Joty,

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 576–584,
Athens, Greece, 30 March – 3 April 2009.

2008) have shown that shallow semantic information in the form of Predicate Argument Structures (PASs) (Jackendoff, 1990; Johnson and Fillmore, 2000) improves the automatic detection of correct answers to a target question. In particular, in (Moschitti et al., 2007) kernels for the processing of PASs (in PropBank[1] format (Kingsbury and Palmer, 2002)) extracted from question/answer pairs were proposed. However, the relatively high kernel computational complexity and the limited improvement on bag-of-words (BOW) produced by this approach do not make the use of such technique practical for real world applications.

In this paper, we carry out a complete study on the use of syntactic/semantic structures for relational learning from questions and answers. We designed sequence kernels for words and Part of Speech Tags which capture basic lexical semantics and basic syntactic information. Then, we design a novel shallow semantic kernel which is far more efficient and also more accurate than the one proposed in (Moschitti et al., 2007).

The extensive experiments carried out on two different corpora of questions and answers, derived from Web documents and the TREC corpus, show that:

• Kernels based on PAS, POS-tag sequences and syntactic parse trees improve the BOW approach on both datasets. On the TREC data the improvement is interestingly high, e.g. about 61%, making its application worthwhile.

• The new kernel for processing PASs is more efficient and effective than previous models so that it can be practically used in systems for short text pair categorization, e.g. question/answer classification.

In the remainder of this paper, Section 2 presents well-known kernel functions for structural information whereas Section 3 describes our new shallow semantic kernel. Section 4 reports on our experiments with the above models and, finally, a conclusion is drawn in Section 5.

2 String and Tree Kernels

Feature design, especially for modeling syntactic and semantic structures, is one of the most difficult aspects in defining a learning system as it requires efficient feature extraction from learning objects. Kernel methods are an interesting representation approach as they allow for the use of all object substructures as features. In this perspective, String Kernel (SK) proposed in (Shawe-Taylor and Cristianini, 2004) and the Syntactic Tree Kernel (STK) (Collins and Duffy, 2002) allow for modeling structured data in high dimensional spaces.

2.1 String Kernels

The String Kernels that we consider count the number of substrings containing gaps shared by two sequences, i.e. some of the symbols of the original string are skipped. Gaps modify the weight associated with the target substrings as shown in the following.

Let Σ be a finite alphabet, $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$ is the set of all strings. Given a string $s \in \Sigma^*$, $|s|$ denotes the length of the strings and s_i its compounding symbols, i.e $s = s_1..s_{|s|}$, whereas $s[i:j]$ selects the substring $s_i s_{i+1}..s_{j-1}s_j$ from the i-th to the j-th character. u is a subsequence of s if there is a sequence of indexes $\vec{I} = (i_1, ..., i_{|u|})$, with $1 \leq i_1 < ... < i_{|u|} \leq |s|$, such that $u = s_{i_1}..s_{i_{|u|}}$ or $u = s[\vec{I}]$ for short. $d(\vec{I})$ is the distance between the first and last character of the subsequence u in s, i.e. $d(\vec{I}) = i_{|u|} - i_1 + 1$. Finally, given s_1, s_2 $\in \Sigma^*$, $s_1 s_2$ indicates their concatenation.

The set of all substrings of a text corpus forms a feature space denoted by $\mathcal{F} = \{u_1, u_2, ..\} \subset \Sigma^*$. To map a string s in $\mathbb{R}^\infty$ space, we can use the following functions: $\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{d(\vec{I})}$ for some $\lambda \leq 1$. These functions count the number of occurrences of u in the string s and assign them a weight $\lambda^{d(\vec{I})}$ proportional to their lengths. Hence, the inner product of the feature vectors for two strings s_1 and s_2 returns the sum of all common subsequences weighted according to their frequency of occurrences and lengths, i.e.

$$SK(s_1, s_2) = \sum_{u \in \Sigma^*} \phi_u(s_1) \cdot \phi_u(s_2) = \sum_{u \in \Sigma^*} \sum_{\vec{I}_1 : u = s_1[\vec{I}_1]} \lambda^{d(\vec{I}_1)}$$

$$\sum_{\vec{I}_2 : u = t[\vec{I}_2]} \lambda^{d(\vec{I}_2)} = \sum_{u \in \Sigma^*} \sum_{\vec{I}_1 : u = s_1[\vec{I}_1]} \sum_{\vec{I}_2 : u = t[\vec{I}_2]} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)},$$

where $d(.)$ counts the number of characters in the substrings as well as the gaps that were skipped in the original string.

2.2 Syntactic Tree Kernel (STK)

Tree kernels compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. Let $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ be the set of tree

[1] www.cis.upenn.edu/~ace

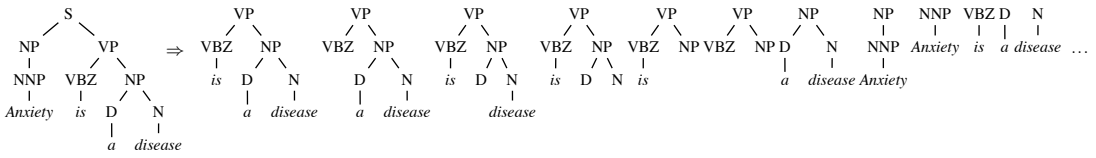

Figure 1: A tree for the sentence "Anxiety is a disease" with some of its syntactic tree fragments.

fragments 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 defined as $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \chi_i(n_1)\chi_i(n_2)$.

Δ function counts the number of subtrees rooted in n_1 and n_2 and can be evaluated as follows (Collins and Duffy, 2002):
1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (i.e. they are pre-terminal symbols) then $\Delta(n_1, n_2) = \lambda$;
3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{l(n_1)} (1 + \Delta(c_{n_1}(j), c_{n_2}(j)))$, where $l(n_1)$ is the number of children of n_1, $c_n(j)$ is the j-th child of node n and λ is a decay factor penalizing larger structures.

Figure 1 shows some fragments of the subtree on the left part. These satisfy the constraint that grammatical rules cannot be broken. For example, *[VP [VBZ NP]]* is a valid fragment which has two non-terminal symbols, *VBZ* and *NP*, as leaves whereas *[VP [VBZ]]* is not a valid feature.

3 Shallow Semantic Kernels

The extraction of semantic representations from text is a very complex task. For it, traditionally used models are based on lexical similarity and tends to neglect lexical dependencies. Recently, work such as (Shen and Lapata, 2007; Surdeanu et al., 2008; Moschitti et al., 2007; Moschitti and Quarteroni, 2008; Chali and Joty, 2008), uses PAS to consider such dependencies but only the latter three researches attempt to completely exploit PAS with Shallow Semantic Tree Kernels (SSTKs). Unfortunately, these kernels result computational expensive for real world applications. In the remainder of this section, we present our new kernel for PASs and compare it with the previous SSTK.

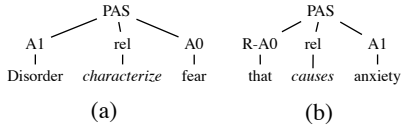

Figure 2: Predicate Argument Structure trees associated with the sentence: "Panic disorder is characterized by unexpected and intense fear that causes anxiety.".

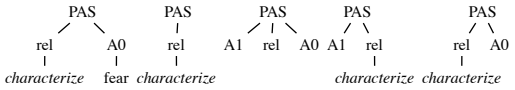

Figure 3: Some of the tree substructures useful to capture shallow semantic properties.

3.1 Shallow Semantic Structures

Shallow approaches to semantic processing are making large strides in the direction of efficiently and effectively deriving tacit semantic information from text. Large data resources annotated with levels of semantic information, such as in the FrameNet (Johnson and Fillmore, 2000) and PropBank (PB) (Kingsbury and Palmer, 2002) projects, make it possible to design systems for the automatic extraction of predicate argument structures (PASs) (Carreras and Màrquez, 2005). PB-based systems produce sentence annotations like:

[$_{A1}$ Panic disorder] is [$_{rel}$ characterized] [$_{A0}$ by unexpected and intense fear] [$_{R-A0}$ that] [$_{rel}$causes] [$_{A1}$ anxiety].

A tree representation of the above semantic information is given by the two PAS trees in Figure 2, where the argument words are replaced by the head word to reduce data sparseness. Hence, the semantic similarity between sentences can be measured in terms of the number of substructures between the two trees. The required substructures violate the STK constraint (about breaking production rules), i.e. since we need any set of nodes linked by edges of the initial tree. For example, interesting semantic fragments of Figure 2.a are shown in Figure 3.

Unfortunately, STK applied to PAS trees cannot generate such fragments. To overcome this problem, a Shallow Semantic Tree Kernel (SSTK) was designed in (Moschitti et al., 2007).

3.2 Shallow Semantic Tree Kernel (SSTK)

SSTK is obtained by applying two different steps: first, the PAS tree is transformed by adding a layer

of SLOT nodes as many as the number of possible argument types, where each slot is assigned to an argument following a fixed ordering (e.g. rel, $A0$, $A1$, $A2$, ...). For example, if an A1 is found in the sentence annotation it will be always positioned under the third slot. This is needed to "artificially" allow SSTK to generate structures containing subsets of arguments. For example, the tree in Figure 2.a is transformed into the first tree of Fig. 4, where "null" just states that there is no corresponding argument type.

Second, to discard fragments only containing slot nodes, in the STK algorithm, a new step 0 is added and the step 3 is modified (see Sec. 2.2):
0. if n_1 (or n_2) is a pre-terminal node and its child label is *null*, $\Delta(n_1, n_2) = 0$;
3. $\Delta(n_1, n_2) = \prod_{j=1}^{l(n_1)} (1 + \Delta(c_{n_1}(j), c_{n_2}(j))) - 1$.

For example, Fig. 4 shows the fragments generated by SSTK. The comparison with the ideal fragments in Fig. 3 shows that SSTK well approximates the semantic features needed for the PAS representation. The computational complexity of SSTK is $O(n^2)$, where n is the number of the PAS nodes (leaves excluded). Considering that the tree including all the PB arguments contains 52 slot nodes, the computation becomes very expensive. To overcome this drawback, in the next section, we propose a new kernel to efficiently process PAS trees with no addition of slot nodes.

3.3 Semantic Role Kernel (SRK)

The idea of SRK is to produce all child subsequences of a PAS tree, which correspond to sequences of predicate arguments. For this purpose, we can use a string kernel (SK) (see Section 2.1) for which efficient algorithms have been developed. Once a sequence of arguments is output by SK, for each argument, we account for the potential matches of its children, i.e. the head of the argument (or more in general the argument word sequence).

More formally, given two sequences of argument nodes, s_1 and s_2, in two PAS trees and considering the string kernel in Sec 2.1, the SRK(s_1, s_2) is defined as:

$$\sum_{\substack{\vec{I}_1 : u = s_1[\vec{I}_1] \\ \vec{I}_2 : u = s_2[\vec{I}_2]}} \prod_{l=1..|u|} (1 + \sigma(s_1[\vec{I}_{1l}], s_2[\vec{I}_{2l}])) \lambda^{d(\vec{I}_1)+d(\vec{I}_2)}, \quad (1)$$

where u is any subsequence of argument nodes, $\vec{I}_l$ is the index of the l-th argument node, $s[\vec{I}_l]$ is the corresponding argument node in the sequence s and $\sigma(s_1[\vec{I}_{1l}], s_2[\vec{I}_{2l}])$ is 1 if the heads of the arguments are identical, otherwise is 0.

Proposition 1 *SRK computes the number of all possible tree substructures shared by the two evaluating PAS trees, where the considered substructures of a tree T are constituted by any set of nodes (at least two) linked by edges of T.*

Proof The PAS trees only contain three node levels and, according to the proposition's thesis, substructures contain at least two nodes. The number of substructures shared by two trees, T_1 and T_2, constituted by the root node (`PAS`) and the subsequences of argument nodes is evaluated by $\sum_{\vec{I}_1 : u = s_1[\vec{I}_1], \vec{I}_2 : u = s_2[\vec{I}_2]} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)}$ (when $\lambda = 1$). Given a node in a shared subsequence u, its child (i.e. the head word) can be both in T_1 and T_2, originating two different shared structures (with or without such head node). The matches on the heads (for each shared node of u) are combined together generating different substructures. Thus the number of substructures originating from u is the product, $\prod_{l=1..|u|}(1+\sigma(s_1[\vec{I}_{1l}], s_2[\vec{I}_{2l}]))$. This number multiplied by all shared subsequences leads to Eq. 1. □

We can efficiently compute SRK by following a similar approach to the string kernel evaluation in (Shawe-Taylor and Cristianini, 2004) by defining the following dynamic matrix:

$$D_p(k, l) = \sum_{i=1}^{k} \sum_{r=1}^{l} \lambda^{k-i+l-r} \times \gamma_{p-1}(s_1[1:i], s_2[1:r]), \quad (2)$$

where $\gamma_p(s_1, s_2)$ counts the number of shared substructures of exactly p argument nodes between s_1 and s_2 and again, $s[1:i]$ indicates the sequence portion from argument 1 to i. The above matrix is then used to evaluate $\gamma_p(s_1 a, s_2 b) =$

$$\begin{cases} \lambda^2(1 + \sigma(h(a), h(b)))D_p(|s_1|, |s_2|) \text{ if } a = b; \\ 0 \qquad\qquad otherwise. \end{cases} \quad (3)$$

where $s_1 a$ and $s_2 b$ indicate the concatenation of the sequences s and t with the argument nodes, a and b, respectively and $\sigma(h(a), h(b))$ is 1 if the children of a and b are identical (e.g. same head). The interesting property is that:

$$\begin{aligned} D_p(k, l) = \gamma_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k, l-1) \\ + \lambda D_p(k-1, l) - \lambda^2 D_p(k-1, l-1). \end{aligned} \quad (4)$$

To obtain the final kernel, we need to consider all possible subsequence lengths. Let m be the minimum between $|s_1|$ and $|s_2|$, $SRK(s_1, s_2) = \sum_{p=1}^{m} \gamma_p(s_1, s_2)$.

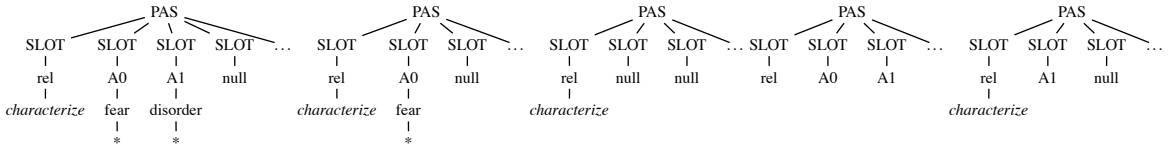

Figure 4: Fragments of Fig. 2.a produced by the SSTK (similar to those of Fig. 3).

Regarding the processing time, if ρ is the maximum number of arguments in a predicate structure, the worst case computational complexity of SRK is $O(\rho^3)$.

3.4 SRK vs. SSTK

A comparison between SSTK and SRK suggests the following points: first, although the computational complexity of SRK is larger than the one of SSTK, we will show in the experiment section that the running time (for both training and testing) is much lower. The worse case is not really informative since as shown in (Moschitti, 2006), we can design fast algorithm with a linear average running time (we use such algorithm for SSTK).

Second, although SRK uses trees with only three levels, in Eq.1, the function σ (defined to give 1 or 0 if the heads match or not) can be substituted by any kernel function. Thus, σ can recursively be an SRK (and evaluate Nested PASs (Moschitti et al., 2007)) or any other potential kernel (over the arguments). The very interesting aspect is that the efficient algorithm that we provide (Eqs 2, 3 and 4) can be accordingly modified to efficiently evaluate new kernels obtained with the σ substitution[2].

Third, the interesting difference between SRK and SSTK (in addition to efficiency) is that SSTK requires an ordered sequence of arguments to evaluate the number of argument subgroups (arguments are sorted before running the kernel). This means that the natural order is lost. SRK instead is based on subsequence kernels so it naturally takes into account the order which is very important: without it, syntactic/semantic properties of predicates cannot be captured, e.g. passive and active forms have the same argument order for SSTK.

Finally, SRK gives a weight to the predicate substructures by considering their length, which also includes gaps, e.g. the sequence (A0, A1) is more similar to (A0, A1) than (A0, A-LOC, A1), in turn, the latter produces a heavier match than (A0, A-LOC, A2, A1) (please see Section 2.1). This is another important property for modeling shallow semantics similarity.

[2]For space reasons we cannot discuss it here.

4 Experiments

Our experiments aim at studying the impact of our kernels applied to syntactic/semantic structures for the detection of relations between short texts. In particular, we first show that our SRK is far more efficient and effective than SSTK. Then, we study the impact of the above kernels as well as sequence kernels based on words and Part of Speech Tags and tree kernels for the classification of question/answer text pairs.

4.1 Experimental Setup

The task used to test our kernels is the classification of the correctness of $\langle q, a\rangle$ pairs, where a is an answer for the query q. The text pair kernel operates by comparing the content of questions and the content of answers in a separate fashion. Thus, given two pairs $p_1 = \langle q_1, a_1\rangle$ and $p_2 = \langle q_2, a_2\rangle$, a kernel function is defined as $K(p_1, p_2) = \sum_\tau K_\tau(q_1, q_2) + \sum_\tau K_\tau(a_1, a_2)$, where τ varies across different kernel functions described hereafter.

As a basic kernel machine, we used our SVM-Light-TK toolkit, available at `disi.unitn.it/moschitti` (which is based on SVM-Light (Joachims, 1999) software). In it, we implemented: the String Kernel (SK), the Syntactic Tree Kernel (STK), the Shallow Semantic Tree Kernel (SSTK) and the Semantic Role Kernel (SRK) described in sections 2 and 3. Each kernel is associated with the above linguistic objects: (i) the linear kernel is used with the bag-of-words (BOW) or the bag-of-POS-tags (POS) features. (ii) SK is used with word sequences (i.e. the Word Sequence Kernel, WSK) and POS sequences (i.e. the POS Sequence Kernel, PSK). (iii) STK is used with syntactic parse trees automatically derived with Charniak's parser; (iv) SSTK and SRK are applied to two different PAS trees (see Section 3.1), automatically derived with our SRL system.

It is worth noting that, since answers often con-

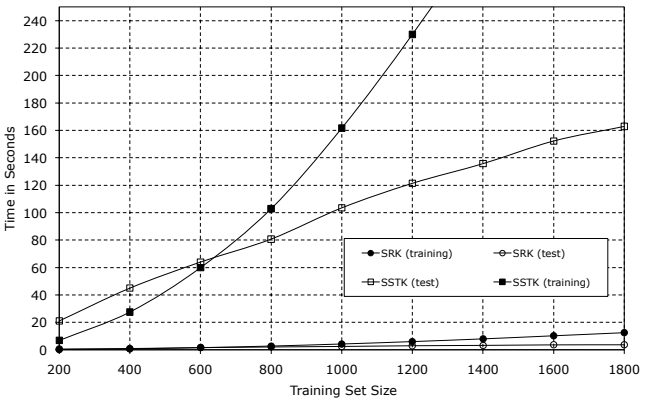

Figure 5: Efficiency of SRK and SSTK

tain more than one PAS, we applied SRK or SSTK to all pairs $P_1 \times P_2$ and sum the obtained contribution, where P_1 and P_2 are the set of PASs of the first and second answer[3]. Although different kernels can be used for questions and for answers, we used (and summed together) the same kernels except for those based on PASs, which are only used on answers.

4.1.1 Datasets

To train and test our text QA classifiers, we adopted the two datasets of question/answer pairs available at `disi.unitn.it/~silviaq`, containing answers to only definitional questions. The datasets are based on the 138 TREC 2001 test questions labeled as "description" in (Li and Roth, 2005). Each question is paired with all the top 20 answer paragraphs extracted by two basic QA systems: one trained with the web documents and the other trained with the AQUAINT data used in TREC'07.

The WEB corpus (Moschitti et al., 2007) of QA pairs contains 1,309 sentences, 416 of which are positive[4] answers whereas the TREC corpus contains 2,256 sentences, 261 of which are positive.

4.1.2 Measures and Parameterization

The accuracy of the classifiers is provided by the average F1 over 5 different samples using 5-fold cross-validation whereas each plot refers to a single fold. We carried out some preliminary experiments of the basic kernels on a validation set and we noted that the F1 was maximized by using the default cost parameters (option -c of SVM-Light), $\lambda = 0.04$ (see Section 2). The trade-off parameter varied according to different kernels on WEB data (so it needed an ad-hoc estimation) whereas a value of 10 was optimal for any kernel on the TREC corpus.

4.2 Shallow Semantic Kernel Efficiency

Section 2 has illustrated that SRK is applied to more compact PAS trees than SSTK, which runs on large structures containing as many slots as the number of possible predicate argument types. This impacts on the memory occupancy as well as on the kernel computation speed. To empirically verify our analytical findings (Section 3.3), we divided the training (TREC) data in 9 bins of increasing size (200 instances between two contiguous bins) and we measured the learning and test time[5] for each bin. Figure 5 shows that in both the classification and learning phases, SRK is much faster than SSTK. With all training data, SSTK employs 487.15 seconds whereas SRK only uses 12.46 seconds, i.e. it is about 40 times faster, making the experimentation of SVMs with large datasets feasible. It is worth noting that to implement SSTK we used the fast version of STK and that, although the PAS trees are smaller than syntactic trees, they may still contain more than half million of substructures (when they are formed by seven arguments).

4.3 Results for Question/Answer Classification

In these experiments, we tested different kernels and some of their most promising combinations, which are simply obtained by adding the different kernel contributions[6] (this yields the joint feature space of the individual kernels).

Table 1 shows the average F1 $\pm$ the standard deviation[7] over 5-folds on Web (and TREC) data of SVMs using different kernels. We note that: (a) BOW achieves very high accuracy, comparable to the one produced by STK, i.e. 65.3 vs 65.1; (b) the BOW+STK combination achieves 66.0, im-

[3] More formally, let P_t and $P_{t'}$ be the sets of PASs extracted from text fragments t and t'; the resulting kernel will be $K_{\mathtt{all}}(P_t, P_{t'}) = \sum_{p \in P_t} \sum_{p' \in P_{t'}} SRK(p, p')$.

[4] For instance, given the question "What are invertebrates?", the sentence "At least 99% of all animal species are invertebrates, comprising ..." was labeled "-1" , while "Invertebrates are animals without backbones." was labeled "+1".

[5] Processing time in seconds of a Mac-Book Pro 2.4 Ghz.

[6] All adding kernels are normalized to have a similarity score between 0 and 1, i.e. $K'(X_1, X_2) = \frac{K(X_1, X_2)}{\sqrt{K(X_1, X_1) \times K(X_2, X_2)}}$.

[7] The Std. Dev. of the difference between two classifier F1s is much lower making statistically significant almost all our system ranking in terms of performance.

WEB Corpus												
BOW	POS	PSK	WSK	STK	SSTK	SRK	BOW+POS	BOW+STK	PSK+STK	WSK+STK	STK+SSTK	STK+SRK
65.3±2.9	56.8±0.8	62.5±2.3	65.7±6.0	65.1±3.9	52.9±1.7	50.8±1.2	63.7±1.6	66.0±2.7	65.3±2.4	66.6±3.0	(+WSK) 68.0±2.7	(+WSK) **68.2**±4.3
TREC Corpus												
24.2±5.0	26.5±7.9	31.6±6.8	14.0±4.2	33.1±3.8	21.8±3.7	23.6±4.7	31.9±7.1	30.3±4.1	36.4±7.0	23.7±3.9	(+PSK) 37.2±6.9	(+PSK) **39.1**±7.3

Table 1: F1 ± Std. Dev. of the question/answer classifier according to several kernels on the WEB and TREC corpora.

proving both BOW and STK; (c) WSK (65.7) improves BOW and it is enhanced by WSK+STK (66.6), demonstrating that word sequences and STKs are very relevant for this task; and finally, WSK+STK+SSTK is slightly improved by WSK+STK+SRK, 68.0% vs 68.2% (not significantly) and both improve on WSK+STK.

The above findings are interesting as the syntactic information provided by STK and the semantic information brought by WSK and SRK improve on BOW. The high accuracy of BOW is surprising if we consider that at classification time, instances of the training models (e.g. support vectors) are compared with different test examples since questions cannot be shared between training and test set[8]. Therefore the answer words should be different and useless to generalize rules for answer classification. However, error analysis reveals that although questions are not shared between training and test set, there are common words in the answers due to typical Web page patterns which indicate if a retrieved passage is an incorrect answer, e.g. `Learn more about X`.

Although the ability to detect these patterns is beneficial for a QA system as it improves its overall accuracy, it is slightly misleading for the study that we are carrying out. Thus, we experimented with the TREC corpus which does not contain Web extra-linguistic texts and it is more complex from a QA task viewpoint (it is more difficult to find a correct answer).

Table 1 also shows the classification results on the TREC dataset. A comparative analysis suggests that: (a) the F1 of all models is much lower than for the Web dataset; (b) BOW shows the lowest accuracy (24.2) and also the accuracy of its combination with STK (30.3) is lower than the one of STK alone (33.1); (c) PSK (31.6) improves POS (26.5) information and PSK+STK (36.4) improves on PSK and STK; and (d) PAS adds further information as the best model is PSK+STK+SRK, which improves BOW from 24.2 to 39.1, i.e. 61%. Finally, it is worth noting that SRK provides a higher improvement (39.1-36.4) than SSTK (37.2-36.4).

4.4 Precision/Recall Curves

To better study the benefit of the proposed linguistic structures, we also plotted the Precision/Recall curves (one fold for each corpus). Figure 6 shows the curve of some interesting kernels applied to the Web dataset. As expected, BOW shows the lowest curves, although, its relevant contribution is evident. STK improves BOW since it provides a better model generalization by exploiting syntactic structures. Also, WSK can generate a more accurate model than BOW since it uses n-grams (with gaps) and when it is summed to STK, a very accurate model is obtained[9]. Finally, WSK+STK+SRK improves all the models showing the potentiality of PASs.

Such curves show that there is no superior model. This is caused by the high contribution of BOW, which de-emphasize all the other models's result. In this perspective, the results on TREC are more interesting as shown by Figure 7 since the contribution of BOW is very low making the difference in accuracy with the other linguistic models more evident. PSK+STK+SRK, which encodes the most advanced syntactic and semantic information, shows a very high curve which outperforms all the others.

The analysis of the above results suggests that: first as expected, BOW does not prove very relevant to capture the relations between short texts from examples. In the QA classification, while BOW is useful to establish the initial ranking by measuring the similarity between question and answer, it is almost irrelevant to capture typical rules suggesting if a description is valid or not. Indeed, since test questions are not in the training set, their words as well as those of candidate answers will be different, penalizing BOW models. In these

[8] Sharing questions between test and training sets would be an error from a machine learning viewpoint as we cannot expect new questions to be identical to those in the training set.

[9] Some of the kernels have been removed from the figures so that the plots result more visible.

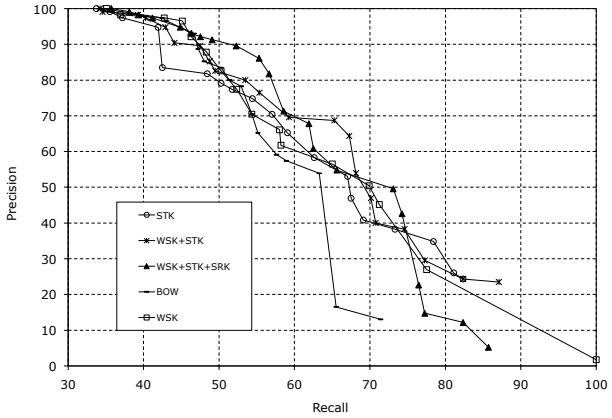

Figure 6: Precision/Recall curves of some kernel combinations on the WEB dataset.

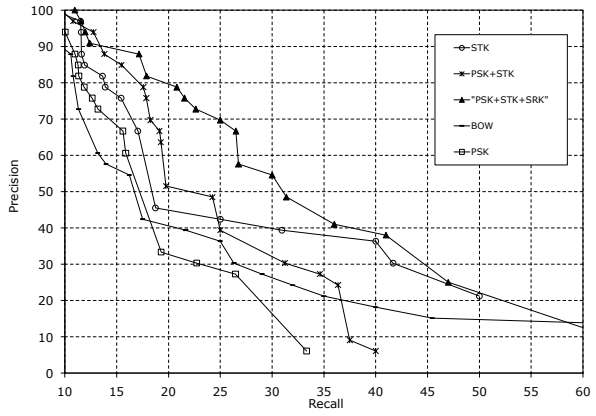

Figure 7: Precision/Recall curves of some kernel combinations on the TREC dataset.

conditions, we need to rely on syntactic structures which at least allow for detecting well formed descriptions.

Second, the results show that STK is important to detect typical description patterns but also POS sequences provide additional information since they are less sparse than tree fragments. Such patterns improve on the bag of POS-tags by about 6% (see POS vs PSK). This is a relevant result considering that in standard text classification bigrams or trigrams are usually ineffective.

Third, although PSK+STK generates a very rich feature set, SRK significantly improves the classification F1 by about 3%, suggesting that shallow semantics can be very useful to detect if an answer is well formed and is related to a question. Error analysis revealed that PAS can provide patterns like:

- `A0(X) R-A0(that) rel(result) A1(Y)`
- `A1(X) rel(characterize) A0(Y),`

where X and Y need not necessarily be matched.

Finally, the best model, PSK+STK+SRK, improves on BOW by 61%. This is strong evidence that complex natural language tasks require advanced linguistic information that should be exploited by powerful algorithms such as SVMs and using effective feature engineering techniques such as kernel methods.

5 Conclusion

In this paper, we have studied several types of syntactic/semantic information: bag-of-words (BOW), bag-of-POS tags, syntactic parse trees and predicate argument structures (PASs), for the design of short text pair classifiers. Our learning framework is constituted by Support Vector Machines (SVMs) and kernel methods applied to automatically generated syntactic and semantic structures.

In particular, we designed (i) a new Semantic Role Kernel (SRK) based on a very fast algorithm; (ii) a new sequence kernel over POS tags to encode shallow syntactic information; (iii) many kernel combinations (to our knowledge no previous work uses so many different kernels) which allow for the study of the role of several linguistic levels in a well defined statistical framework.

The results on two different question/answer classification corpora suggest that (a) SRK for processing PASs is more efficient and effective than previous models, (b) kernels based on PAS, POS-tag sequences and syntactic parse trees improve on BOW on both datasets and (c) on the TREC data the improvement is remarkably high, e.g. about 61%.

Promising future work concerns the definition of a kernel on the entire argument information (e.g. by means of lexical similarity between all the words of two arguments) and the design of a *discourse kernel* to exploit the relational information gathered from different sentence pairs. A closer relationship between questions and answers can be exploited with models presented in (Moschitti and Zanzotto, 2007; Zanzotto and Moschitti, 2006). Also the use of PAS derived from FrameNet and PropBank (Giuglea and Moschitti, 2006) appears to be an interesting research line.

Acknowledgments

I would like to thank Silvia Quarteroni for her work on extracting linguistic structures. This work has been partially supported by the European Commission - LUNA project, contract n. 33549.

References

J. Allan. 2000. Natural Language Processing for Information Retrieval. In *NAACL/ANLP (tutorial notes)*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: SRL. In *CoNLL-2005*.

Y. Chali and S. Joty. 2008. Improving the performance of the random walk model for answering complex questions. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio.

Y. Chen, M. Zhou, and S. Wang. 2006. Reranking answers from definitional QA using language models. In *ACL'06*.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL'02*.

K. Collins-Thompson, J. Callan, E. Terra, and C. L.A. Clarke. 2004. The effect of document retrieval quality on factoid QA performance. In *SIGIR'04*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *ACL04*, Barcelona, Spain.

C. Cumby and D. Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*, Washington, DC, USA.

J. Furnkranz, T. Mitchell, and E. Rilof. 1998. A case study in using linguistic phrases for text categorization on the www. In *Working Notes of the AAAI/ICML, Workshop on Learning for Text Categorization*.

D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop*, Prague.

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

A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. 2006. Question answering with lccs chaucer at trec 2006. In *Proceedings of TREC'06*.

E. Hovy, U. Hermjakob, C.-Y. Lin, and D. Ravichandran. 2002. Using knowledge to facilitate factoid answer pinpointing. In *Proceedings of Coling*, Morristown, NJ, USA.

R. Jackendoff. 1990. *Semantic Structures*. MIT Press.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*.

C. R. Johnson and C. J. Fillmore. 2000. The framenet tagset for frame-semantic and syntactic coding of predicate-argument structures. In *ANLP-NAACL'00*, pages 56–62.

J. Kazama and K. Torisawa. 2005. Speeding up Training with Tree Kernels for Node Relation Labeling. In *Proceedings of EMNLP 2005*, pages 137–144, Toronto, Canada.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *LREC'02*.

T. Kudo and Y. Matsumoto. 2003. Fast Methods for Kernel-Based Text Analysis. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of ACL*.

T. Kudo, J. Suzuki, and H .Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*, US.

D. D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92*.

X. Li and D. Roth. 2005. Learning question classifiers: the role of semantic information. *JNLE*.

J. Lin and B. Katz. 2003. Question answering from the web using knowledge annotation and knowledge mining techniques. In *CIKM '03*.

A. Moschitti and R. Basili. 2004. Complex linguistic features for text classification: A comprehensive study. In *ECIR, Sunderland, UK*.

A. Moschitti and C. Bejan. 2004. A semantic kernel for predicate argument classification. In *CoNLL-2004*, Boston, MA, USA.

A. Moschitti and S. Quarteroni. 2008. Kernels on linguistic structures for answer extraction. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio.

A. Moschitti and F. Zanzotto. 2007. Fast and effective kernels for relational learning from texts. In Zoubin Ghahramani, editor, *Proceedings of ICML 2007*.

A. Moschitti, D. Pighin, and R. Basili. 2006. Semantic role labeling via tree kernel joint inference. In *Proceedings of CoNLL-X*, New York City.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *ACL'07*, Prague, Czech Republic.

A. Moschitti. 2006. Making Tree Kernels Practical for Natural Language Learning. In *Proceedings of EACL2006*.

V. Punyakanok, D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004*.

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

D. Shen and M. Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*.

L. Shen, A. Sarkar, and A. k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *EMNLP*, Sapporo, Japan.

M. Surdeanu, M. Ciaramita, and H. Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of ACL-08: HLT*, Columbus, Ohio.

K. Toutanova, P. Markova, and C. Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

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

E. M. Voorhees. 2004. Overview of the trec 2001 question answering track. In *Proceedings of the Thirteenth Text REtreival Conference (TREC 2004)*.

F. M. Zanzotto and A. Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st Coling and 44th ACL*, Sydney, Australia.

D. Zhang and W. Lee. 2003. Question classification using support vector machines. In *SIGIR'03*, Toronto, Canada. ACM.

M. Zhang, J. Zhang, and J. Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In *Proceedings of NAACL*, New York City, USA.