Kernel Methods, Syntax and Semantics for Relational Text Categorization

Alessandro Moschitti
Department of Information Engineering and Computer Science
University of Trento
38100 Povo di Trento, Italy
moschitti@disi.unitn.it

ABSTRACT

Previous work on Natural Language Processing for Information Retrieval has shown the inadequateness of semantic and syntactic structures for both document retrieval and categorization. The main reason is the high reliability and effectiveness of language models, which are sufficient to accurately solve such retrieval tasks. However, when the latter involve the computation of relational semantics between text fragments simple statistical models may result ineffective. In this paper, we show that syntactic and semantic structures can be used to greatly improve complex categorization tasks such as determining if an answer correctly responds to a question. Given the high complexity of representing semantic/syntactic structures in learning algorithms, we applied kernel methods along with Support Vector Machines to better exploit the needed relational information. Our experiments on answer classification on Web and TREC data show that our models greatly improve on bag-of-words.

Categories and Subject Descriptors

H. [**Information Systems**]: Information Storage and Retrieval—*Content Analysis and Indexing, Linguistic processing*

General Terms

Algorithms, Experimentation, Performance

Keywords

Support Vector Machines, Kernel Methods, Text Categorization, Question Answering, Natural Language Processing

1. INTRODUCTION

Previous work on Natural Language Processing (NLP) for Information Retrieval (IR) has shown that the increase of the computational complexity for the processing of advanced linguistic representations is not justified by the small gain in accuracy, which often turns out to be a decrease. For example, apparently promising linguistic structures like *subject-verb-object* have been shown in TREC to be inadequate for typical retrieval tasks [35]. Similar findings have been derived for Text Categorization using advanced linguistic processing, e.g. [22, 12, 1, 26]. Despite such failure, work on Question Answering (QA) suggests that syntactic and linguistic structures help in solving the task [41, 13].

From the above studies, it emerges that when the retrieval task is linguistically *complex*, syntax and semantics may play a relevant role. In this perspective, one of the most complex Text Categorization task relates to the detection of the relationships between two text fragments. One typical example of relation, useful for designing retrieval systems, is the one holding between question and answer, i.e. if the latter text fragment correctly responds to the former.

In QA, this task is mostly tackled by using different heuristics and classifiers, which aim at extracting the best answers [6, 8]. However, if the question is a definition, a more effective approach would be to test if there is a correct relationship between the answer and the query. This depends on the structures of the two text fragments. Designing language models to capture such a relation could be too complex since it requires expensive probabilistic models (in terms of design and computational resources) for the representation of structural information. More specifically, such models suffer from (i) computational complexity issues, e.g. the processing of large bayesian networks, (ii) a high complexity to estimate and smooth probabilities and (iii) high sensitiveness to irrelevant features and processing errors. These aspects make the use of linguistic processing very difficult since it inevitably introduces structures which contain noise and errors.

In contrast, discriminative models such as Support Vector Machines (SVMs) [38] have been proven to be robust to noise and irrelevant features. Thus, partially correct linguistic structures may still provide a relevant contribution since only the relevant information will 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. [7, 20, 11, 32, 10, 21, 37, 17, 44]. More specifically, in question classification, tree kernels [43, 25] have shown accuracy comparable to the best models, e.g. [23].

Moreover, [31, 28, 24] have shown that shallow semantic information in the form of predicate argument structures (PASs) [14, 16] improves the automatic detection of correct answers to a target question. In particular, in [28], we proposed kernels for processing PASs (in PropBank[1] format [19]) extracted from question/answer pairs. However, the relatively high kernel computational complexity and the limited improvement on the bag-of-words (BOW) do

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

not make the use of such technique practical for real world retrieval applications.

In this paper, we carry out a thorough 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 [28].

The extensive experiments carried out on two different corpora of questions and answers, which have been 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 60%, 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 answer re-ranking systems.

- Our best question/answer classifier, used as re-ranker, significantly improves the QA system accuracy, confirming its promising applicability.

In the remainder, Section 2 presents our use of kernel functions for structural information and Section 3 introduces our data representations. Section 4 reports on our experiments with the above models whereas Section 5 illustrates the related work. Finally, conclusions are drawn in Section 6.

2. KERNELS FOR STRUCTURED DATA

Kernel Methods refer to a large class of learning algorithms based on inner product vector spaces, among which Support Vector Machines (SVMs) are one of the most well-known algorithms. The main idea behind SVMs is to learn a hyperplane $H(\vec{x}) = \vec{w}\cdot\vec{x}+b = 0$, where $\vec{x}$ is the feature vector representation of a classifying object o whereas $\vec{w} \in \Re^n$ (a vector space) and $b \in \Re$ are parameters learnt from training examples by applying the *Structural Risk Minimization principle* [38]. The object o is mapped in $\vec{x}$ with a feature function $\phi : \mathcal{O} \rightarrow \Re^n$, where $\mathcal{O}$ is the set of the objects. o is categorized in the target class only if $H(\vec{x}) \geq 0$.

The kernel trick allows us to rewrite the decision hyperplane as:

$$H(\vec{x}) = \Big(\sum_{i=1..l} y_i\alpha_i\vec{x}_i \Big) \cdot \vec{x} + b =$$

$$\sum_{i=1..l} y_i\alpha_i\vec{x}_i \cdot \vec{x} + b = \sum_{i=1..l} y_i\alpha_i\phi(o_i)\cdot\phi(o) + b,$$

where y_i is equal to 1 for positive and -1 for negative examples, $\alpha_i \in \Re$ with $\alpha_i \geq 0$, o_i $\forall i \in \{1,..,l\}$ are the training instances and the product $K(o_i, o) = \langle\phi(o_i)\cdot\phi(o)\rangle$ is the kernel function associated with the mapping ϕ.

Note that it is not necessary to apply the mapping ϕ, we can use $K(o_i, o)$ directly. This allows, under the Mercer's conditions [30], for defining abstract kernel functions which generate implicit feature spaces. In turn, this alleviates the feature extraction/design step and allows for the use of huge feature space (possibly infinite) since the scalar product (i.e. $K(\cdot,\cdot)$) is implicitly evaluated.

In the remainder of this section, we present the String Kernel (SK) proposed in [30] to evaluate the number of subsequences between two sequences, the Syntactic Tree Kernel (STK) [7], which computes the number of syntactic tree fragments, the Shallow Semantic Tree Kernel (SSTK) [28], which considers fragments from PASs, and the Partial Tree Kernel (PTK) [25], which provides a very general representation of trees in terms of tree fragments.

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 characters 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 $\sigma \in \Sigma^*$, $|\sigma|$ denotes the length of the string, σ can be written as $s_1..s_{|s|}$ with $s_i \in \Sigma$ and $\sigma[i:j]$ selects the substring $s_is_{i+1}..s_{j-1}s_j$ from the i-th to the j-th character. u is a subsequence of σ if there is a sequence of indices $\vec{I} = (i_1, ..., i_{|u|})$, with $1 \leq i_1 < ... < i_{|u|} \leq |\sigma|$, such that $u = s_{i_1}..s_{i_{|u|}}$ or $u = \sigma[\vec{I}]$ for short. $d(\vec{I})$ is the distance between the first and last character of the subsequence u in σ, i.e. $d(\vec{I}) = i_{|u|} - i_1 + 1$. Finally, given $\sigma_1, \sigma_2 \in \Sigma^*$, $\sigma_1\sigma_2$ indicates their concatenation.

The set of all substrings of a text corpus forms a feature space denoted by $\mathcal{F} \subset \Sigma^*$. To map a string σ into $\mathbb{R}^\infty$ space, we can use the following functions: $\phi_u(\sigma) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{d(\vec{I})}$ for some $\lambda \leq 1$. These functions count the number of occurrences of u in the string σ and assign them a weight $\lambda^{d(\vec{I})}$ proportional to their lengths. Hence, the inner product of the feature vectors for two strings σ_1 and σ_2 returns the sum of all common subsequences weighted according to their frequency of occurrences and lengths, i.e.

$$SK(\sigma_1,\sigma_2) = \sum_{u\in\Sigma^*} \phi_u(\sigma_1)\cdot\phi_u(\sigma_2) = \sum_{u\in\Sigma^*} \sum_{\vec{I}_1:u=\sigma_1[\vec{I}_1]} \lambda^{d(\vec{I}_1)}$$

$$\sum_{\vec{I}_2:u=\sigma_2[\vec{I}_2]} \lambda^{d(\vec{I}_2)} = \sum_{u\in\Sigma^*} \sum_{\vec{I}_1:u=\sigma_1[\vec{I}_1]} \sum_{\vec{I}_2:u=\sigma_2[\vec{I}_2]} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)}$$

It is worth to note that: (a) longer subsequences receive lower weights; (b) valid substrings are sequences of the original string with some characters omitted, i.e. gaps; (c) gaps determine the weighting function since $d(.)$ counts the number of characters in the substrings as well as the gaps that were skipped in the original string, and (d) symbols of a string can also be whole words, i.e. the Word Sequence Kernel [4].

2.2 Tree Kernels

The main underlying idea of tree kernels is to 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 fragments and $\chi_i(n)$ 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 $\Delta(n_1,n_2) = \sum_{i=1}^{|\mathcal{F}|} \chi_i(n_1)\chi_i(n_2)$.

The Δ function is equal to the number of common fragments rooted in nodes n_1 and n_2, and thus, depends on the fragment type. We report its algorithm for the evaluation of the number of syntactic tree fragments (STFs) [7], the number of shallow semantic tree fragments (SSTFs) [28], and the number of partial tree fragment (PTFs) [25].

2.2.1 *Syntactic Tree Kernel (STK)*

A syntactic tree fragment (STF) is a set of nodes and edges from the original tree which is still a tree and with the constraint that

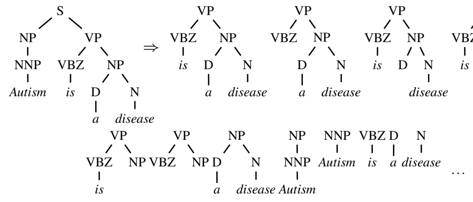


Figure 1: A tree for the sentence "Autism is a disease" with some of its syntactic tree fragments (STFs).

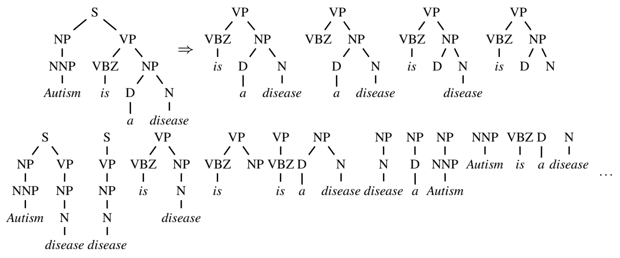


Figure 2: A tree for the sentence "Autism is a disease" with some of its partial tree fragments (PTFs).

any node must have all or none of its children. This is equivalent to stating that the production rules contained in the STF cannot be partial.

To compute the number of common STFs rooted in n_1 and n_2, the STK uses the following Δ function [7]:

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 10 STFs (out of 17) of the subtree on the left. STFs satisfy the constraint that grammatical rules cannot be broken. For example, *[VP [VBZ NP]]* is a STF which has two non-terminal symbols, *VBZ* and *NP*, as leaves whereas *[VP [VBZ]]* is not a STF.

2.2.2 Shallow Semantic Tree Kernel (SSTK)

A shallow semantic tree fragment (SSTF) is almost identical to a STF, the difference being that the contribution of special nodes labeled with *null* should be zero. This is necessary as the SSTK is applied to special trees containing SLOT nodes, which, when empty, have children labeled with *null*. Two steps are modified in the algorithm:

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. $\Delta(n_1, n_2) = \prod_{j=1}^{l(n_1)} (1 + \Delta(c_{n_1}(j), c_{n_2}(j))) - 1$,

2.2.3 Partial Tree Kernel (PTK)

If we relax the production rule constraint over the STFs, we obtain a more general substructure type called partial tree fragment (PTF), generated by the application of partial production rules, e.g. *[VP [VBZ [is]]]* in Figure 2. The Δ function for PTK is relatively simple. Given two nodes n_1 and n_2, STK is applied to all possible child subsequences of the two nodes, i.e. the String Kernel is applied to determine the subsequences and the STK is applied on each of such child substrings. More formally:

1. if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. else $\Delta(n_1, n_2) =$
$$1 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))$$

where $\vec{I}_1 = \langle h_1, h_2, h_3, .. \rangle$ and $\vec{I}_2 = \langle k_1, k_2, k_3, .. \rangle$ are index sequences associated with the ordered child sequences c_{n_1} of n_1 and c_{n_2} of n_2, respectively, $\vec{I}_{1j}$ and $\vec{I}_{2j}$ point to the j-th child in the corresponding sequence, and, again, $l(\cdot)$ returns the sequence length, i.e. the number of children.

Furthermore, we add two decay factors: μ for the depth of the tree and λ for the length of the child subsequences with respect to the original sequence, i.e. we account for gaps. It follows that

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

where $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11}$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21}$. This way, we penalize both larger trees and child subsequences with gaps[2].

2.3 Kernel Engineering

Kernel engineering can be carried out by combining basic kernels with additive or multiplicative operators or by designing specific data objects (vectors, sequences and tree structures) for the target tasks.

It is worth noting that kernels applied to new structures produce new kernels as shown hereafter. Let $\mathrm{K}(t_1, t_2) = \phi(t_1) \cdot \phi(t_2)$ be a basic kernel, where t_1 and t_2 are two trees. If we map t_1 and t_2 into two new structures s_1 and s_2 with a mapping $\phi_M(\cdot)$, we obtain: $\mathrm{K}(s_1, s_2) = \phi(s_1) \cdot \phi(s_2) = \phi(\phi_M(t_1)) \cdot \phi(\phi_M(t_2)) = \phi'(t_1) \cdot \phi'(t_2) = \mathrm{K}'(t_1, t_2)$, which is a noticeably different kernel induced by the mapping $\phi' = \phi \circ \phi_M$.

For instance, in the next section, we will define the novel PAS$_{PTK}$ and POS$_{SK}$ kernels by applying PTK and SK to innovative structures, i.e. predicate argument structures and sequences of Part of Speech Tags, respectively.

3. RELATIONAL REPRESENTATIONS FOR QUESTION AND ANSWER PAIRS

Capturing the relationships between two text fragments is a complex task. Some work regarding the relation between question and answer has been carried out in [29, 13, 42]. In such work, the answer extraction step is implemented by means of unsupervised approaches which measure the relevance between questions and answers. However, learning to classify if an answer is correct for a question (the problem will be formally defined in the next section) is conceptually different from extraction in that not only the relatedness between the target question and answer is taken into account but also the other Q/A training pairs are used. The similarity between pairs clearly depends on syntactic and semantic properties; thus, in addition to the usual bag-of-words (BOW), we study methods to capture Q/A structures using String Kernels over word and POS-tag sequences and tree kernels over full syntactic parse trees (PTs) and shallow semantic trees.

[2]An efficient algorithm for its computation is given in [25].

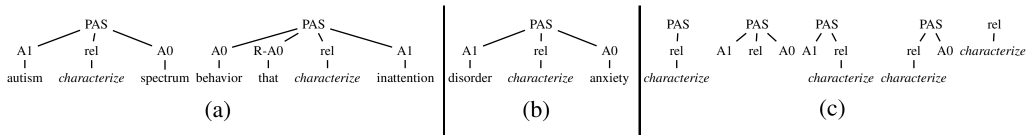


Figure 3: Compact PAS$_{PTK}$ structures of s_1 (a) and s_2 (b) and some fragments they have in common as produced by the PTK (c). Arguments are replaced with their most important word (or semantic head) to reduce data sparseness.

3.1 The Classification of Paired Texts

A Q/A classifier receives pairs $\langle q, a\rangle$ as input and judges if the answer a correctly responds to the query q. For its design, a set of examples of correct and incorrect pairs is needed. The learning algorithm operates by comparing the content of questions and the content of answers in a separate fashion rather than just comparing a question with its corresponding candidate answers. In a learning framework where kernel functions are deployed, 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 $K(p_1, p_2) = K_\tau(q_1, q_2) \oplus K_\alpha(a_1, a_2)$, where K_τ and K_α are kernel functions defined over questions and over answers, respectively, and $\oplus$ is a valid operation between kernels, e.g. sum or multiplication.

In Section 2, we described sequence and tree kernels, which can be applied to the sequential and tree representations of questions and answers, respectively. In the following section we describe several of such linguistically motivated representations.

3.2 Representation with Word and POS-tag sequences and Trees

For a basic syntactic and semantic representation of both questions and answers, we propose two different kernels: the Part of Speech Sequence Kernel (POS$_{SK}$) and Word Sequence Kernel (WSK). The former is obtained by applying the String Kernel on the sequence of POS-tags of a question or answer. For example, given the sentence s_0: *What is autism?*, the associated POS sequence is *WP AUX NN ?* and some of subsequences extracted by POS$_{SK}$ are *WP NN* or *WP AUX*. WSK is applied to word sequences of both questions and answers; given s_0, sample substrings are: *What is autism*, *What is*, *What autism*, *is autism*, etc.

A more complete structure is the full parse tree (PT) of the sentence, that constitutes the input of the STK. For instance, the STK accepts the syntactic parse, *(SBARQ (WHNP (WP What))(SQ (VP (AUX is)(NP (NN autism))))(. ?))*, and extracts from it all possible tree fragments (STFs).

3.3 Shallow Semantic Representation

Our semantic representation takes into account that definitions are characterized by a latent semantic structure, thanks to which *similar concepts* result in *structurally similar* formulations. Precisely, understanding whether a candidate answer is correct for a definition question would imply knowing the correct definition and comparing the current candidate to the former. When such information is unavailable (as in open domain QA) the learning algorithm must mimic the behavior of a human (who does not know the exact definition) and check whether such answer is formulated as a "typical" definition and whether answers defining similar concepts are expressed in a similar way. A method to capture sentence structure [2] is the use of predicate argument structures described hereafter.

3.3.1 Predicate Argument 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 as in the FrameNet [16] and ProbBank [19] projects, make it possible to design systems for the automatic extraction of predicate argument structures (PASs)[5].

Such systems identify predicates (e.g. verbs) and their arguments in a sentence. For example, in the English sentence, 'John likes apples.', the predicate is 'likes' whereas 'John' and 'apples', bear the semantic role labels *agent* (ARG0) and *theme* (ARG1). The crucial fact about semantic roles is that regardless of the overt syntactic structure variation, the underlying predicates remain the same. Hence, for the sentence 'John opened the door' and 'the door opened', although 'the door' is the object of the first sentence and the subject of the second, it is the 'theme' in both sentences. Same idea applies to passive constructions.

To represent PASs in the learning algorithm, we consider two trees: Shallow Semantic Trees for SSTK and Shallow Semantic Trees for PTK, both according to PropBank definition, indicated as PAS$_{SSTK}$ and PAS$_{PTK}$, respectively. These are automatically generated by our system. As an example, let us consider the sentence (from our QA TREC corpus)

s_1: *Autism is characterized by a broad spectrum of behavior that includes extreme inattention to surroundings and hypersensitivity to sound and other stimuli.*,

which results in the PB annotation:

[$_{A1}$ Autism] is [$_{rel}$ characterized] [$_{A0}$ by a broad spectrum of behavior] [$_{R-A0}$ that] [$_{rel}$includes] [$_{A1}$ extreme inattention to surroundings and hypersensitivity to sound and other stimuli].

Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.

s_2: *Panic disorder is characterized by unrealistic or excessive anxiety.*

This results in the PB annotation:

[$_{A1}$ Panic disorder] is [$_{rel}$ characterized] [$_{A0}$ by unrealistic or excessive anxiety].

It can be observed here that, although autism is a different disease from panic disorder, the structure of both definitions and the latent semantics they contain (inherent to behavior, disorder, anxiety) are similar. So for instance, s_2 appears as a definition even to someone who only knows what the definition of autism looks like.

The above annotation can be compactly represented by predicate argument structure trees (PASs) such as those in Figure 3. Here, we notice that the semantic similarity between sentences is explicitly visible in terms of common fragments extracted by PTK from their respective PASs.

An equivalent PAS representation (PAS$_{SSTK}$) compatible with SSTK (see Section 2.2.2) was introduced in [28] (see Figure 4). Here, arguments follow a fixed ordering (i.e. rel, $A0$, $A1$, $A2$, ...) and a layer of SLOT nodes "artificially" allows SSTK to generate structures containing subsets of arguments. PAS$_{PTK}$ is semantically equivalent to PAS$_{SSTK}$ but PTK is able to extract a richer set of features which take gaps into account, (compare the first two fragments of Figures 3.(c) and 4). Moreover, PAS$_{PTK}$ does not need SLOT nodes to extract fragments containing argument subsets. This results in a visibly more compact representation (compare Figures 3.(b) and 4). Moreover, the accuracy of computing the

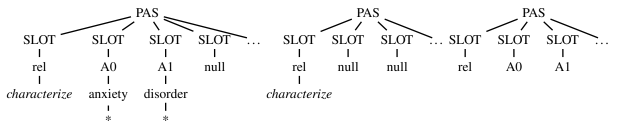


Figure 4: PAS$_{SSTK}$ of s_2 and some of its fragments produced by the SSTK.

matches between two PASs can only improve as only nodes that are actually useful are represented.

4. EXPERIMENTS

Our experiments aim at studying the impact of kernel methods applied to linguistic representations for the detection of QA relations. In particular, we first show that our new shallow semantic tree kernel is far more efficient and effective than previously proposed kernels for shallow semantic processing. Then, we study the impact of the new Part of Speech Tagging sequence kernel, the Word Sequence Kernel and novel kernel combinations for QA classification. Finally, since the obvious use of the above classifier is to improve the answer extraction phase, we experimented with it and a basic QA system.

4.1 Experimental Setup

We implemented the String Kernel (SK), the Syntactic Tree Kernel (STK), the Shallow Semantic Tree Kernel (SSTK) and the Partial Tree Kernel (PTK) described in Section 2 in our SVM-Light-TK toolkit, available at `disi.unitn.it/moschitti` (which is based on SVM-Light [15] software). Each kernel is associated with diverse input objects:

- the linear kernel is used with the bag-of-words (BOW) or bag-of-POS-tags (POS) features;
- SK is used with word sequences (WSK) and POS sequences (POS$_{SK}$);
- STK is used with syntactic parse trees (PTs) automatically derived with Charniak's parser;
- SSTK and PTK are applied to two different PASs (see Section 3.3), i.e. PAS$_{SSTK}$ and PAS$_{PTK}$, automatically derived with our SRL system [27].

Kernel combinations are obtained by simply summing the target kernels. In particular, since answers often contain more than one PAS (see Figure 3), we sum PTK (or SSTK) applied to all pairs $P_1 \times P_2$, 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 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 designed two datasets containing answers for definitional questions only (available at `disi.unitn.it/~silviaq/resources.html`). These are among the most complex and interesting in the literature [18, 9] as they require deeper linguistic processing than factoid answers. We chose them since the models for the solution of factoid questions may be too easy, e.g. simple language model approaches along with the use of a named entity recognizer may be sufficient.

[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 $K_{\texttt{all}}(P_t, P_{t'}) = \sum_{p \in P_t} \sum_{p' \in P_{t'}} SSTK(p, p')$.

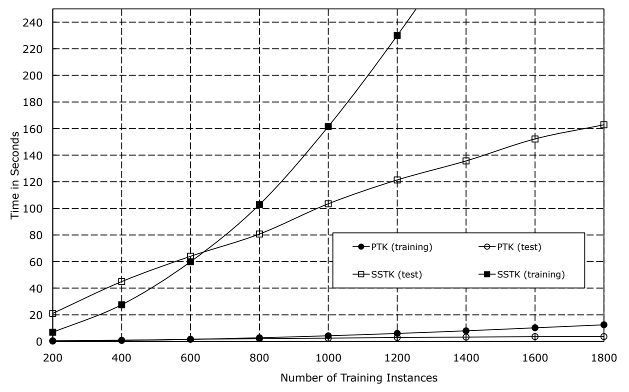


Figure 5: Efficiency of PTK and SSTK

The datasets were created by first collecting the 138 TREC 2001 test questions labeled as "description" in [23] (one of the largest available corpus of description questions) and, for each question, by gathering the top 20 answer paragraphs extracted by our basic QA system (BQAS).

BQAS was run on two sources: Web documents by utilizing Google (`code.google.com/apis/`) and the AQUAINT data used for TREC'07 (`trec.nist.gov/data/qa`) by exploiting Lucene (`lucene.apache.org`), yielding two QA classification corpora, namely WEB and TREC. Each paragraph was manually evaluated based on whether it contained an answer to the corresponding question. To simplify the classification problem, for each paragraph, we isolated the sentence with the maximal judgment[4] and labeled it as positive if it answered the question either concisely or with noise or as negative otherwise. The WEB corpus contains 1309 sentences, 416 of which are positive[5] answers and the TREC-QA corpus contains 2256 sentences, 261 of which are positive.

4.1.2 Measures and Parameterization

The accuracy of the classifiers is provided by F1 whereas the QA system performance is measured in terms of the Mean Reciprocal Rank (MRR)[6]. All values reported on tables refer to the average of 5 different samples using 5-fold cross-validation whereas typically 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), $\lambda = 0.04$ and $\mu = 0.4$ (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 15 was optimal for any kernel on TREC corpus.

4.2 Shallow Semantic Kernel Efficiency

To make the use of kernels for the processing of semantic information practical, we designed PAS$_{PTK}$, which is specifically designed for PTK. This is more compact than PAS$_{SSTK}$ and can be efficiently processed by PTK. In contrast, SSTK runs on large

[4]In case more than one sentence in the paragraph had the same judgment, we chose the first one.

[5]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".

[6]$MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{rk_i}$, where n is the number of questions and rk_i is the rank of the first correct answer to question i.

WEB Question/Answer Classification Corpus												
BOW	POS	POS$_{SK}$	WSK	PT	PAS$_{SSTK}$	PAS$_{PTK}$	BOW+POS	BOW+PT	POS$_{SK}$+PT	WSK+PT	PT+PAS$_{SSTK}$ +WSK	PT+PAS$_{PTK}$ +WSK
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	68.0±2.7	**68.2**±4.3
TREC Question/Answer Classification Corpus											+POS$_{SK}$	+POS$_{SK}$
24.2±3.1	29.8±5.8	32.8±6.9	25.3±3.8	33.8±4.5	21.8±3.7	23.6±4.7	28.3±4.2	30.2±5.3	36.4±9.3	32.9±7.8	36.2±7.1	**39.1**±6.9

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

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. PTK is able to process the same information with much smaller structures.

To test the above mentioned characteristics, we divided the training (TREC) data in 9 bins of increasing size (200 instances between two contiguous bins) and we measured the learning and testing time[7] for each bin. Figure 5 shows that in both the classification and learning phases, PTK is much faster than SSTK. With all training data, SSTK employs 487.15 seconds whereas PTK only uses 12.46 seconds, i.e. it is about 40 times faster, making the experimentation of SVMs with large datasets feasible.

To completely assess the benefit of PAS$_{PTK}$ with respect to PAS$_{SSTK}$, we also need to compare them in terms of classification accuracy. In the next section, we extensively test several kernels and their combinations.

4.3 Results for Question/Answer Classification

In these experiments, we tested different kernels and some of their most promising combinations. Since the nature of the type of the applied kernels strongly depends on the data they operate on, we simplify our notation by using only the name of the representation instead of using the more appropriate name combination (representation and kernel). In other words, we used BOW, POS and PT to indicate that a linear kernel is applied to bag-of-words and POS vectors and the syntactic tree kernel is applied to parse tree (PT).

The other kernel names show a subscription indicating the applied kernel, i.e. POS$_{SK}$, PAS$_{SSTK}$ and PAS$_{PTK}$. This suggests that SK is applied to POS sequences and that SSTK and PTK are applied to the PAS structures. The only exception is WSK indicating the Word Sequence Kernel, i.e. a string kernel applied to word sequences.

As kernel combinations, we used the sum between kernels[8] since it yields the joint feature space of the individual kernels [30].

Table 1 shows the average F1 $\pm$ the standard deviation over 5-folds on Web (and TREC) data of SVMs using different kernels. We note that:

- BOW achieves very high accuracy, comparable to the one produced by PT, i.e. 65.3 vs 65.1;

- the BOW+PT combination achieves 66.2, improving both BOW and PT but BOW+POS produces a lower F1, i.e. 63.7, than PT and BOW, indicating that POS does not provide useful information for this dataset;

- WSK improves (65.7) BOW and their sum is enhanced by WSK+PT (66.6), demonstrating that word sequences and PTs are very relevant for this task;

- both PAS$_{SSTK}$ and PAS$_{PTK}$ improve BOW yielding the highest results, i.e. about 68.

The above findings are interesting as the syntactic information provided by STK and the semantic information brought by WSK and PAS$_{PTK}$ 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[9]. Therefore the answer words should be different and useless to generalize rules for answer classification. However, error analysis revealed that although questions are not shared between training and test set, there are common patterns 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 our study. 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:

- the F1 of all models is much lower than for the Web dataset;

- BOW shows the lowest accuracy (24.2) and also the accuracy of its combination with PT (30.2) is lower than the one of PT alone (32.8);

- additionally, when BOW is summed to POS (29.8) produces a lower result (28.3);

- SK is beneficial for exploiting POS information as POS$_{SK}$+PT (36.4) improves on POS and PT.

Finally, PAS adds further information as the best model is POS$_{SK}$+ PT+PAS$_{PTK}$, which improves BOW from 24.2 to 39.1, i.e. 61%.

4.3.1 Precision/Recall Curves

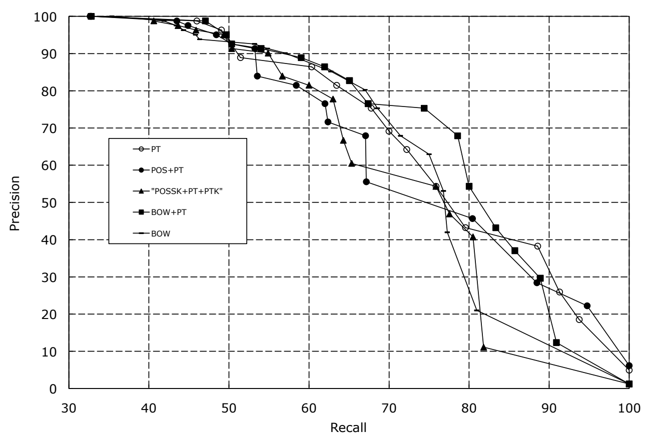
To better study the benefit of the proposed linguistic structures, we also plotted the Precision/Recall curves. Figure 6 shows the curves of some interesting kernels for four (out of five) folds of the Web dataset. As expected, BOW almost always shows the lowest curves (on 3 folds). Anyway, its relevant contribution is evident: when summed to PT, it produces the highest curves on folds 1 and 2. Moreover, WSK, which is able to exploits n-grams (with gaps), summed to PT produces very high curves[10] (see folds 3 and 4). In summary, all the kernel combinations tend to achieve slightly higher result than BOW. Again, the cause is the high contribution of BOW, which prevents the other models to clearly emerge.

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

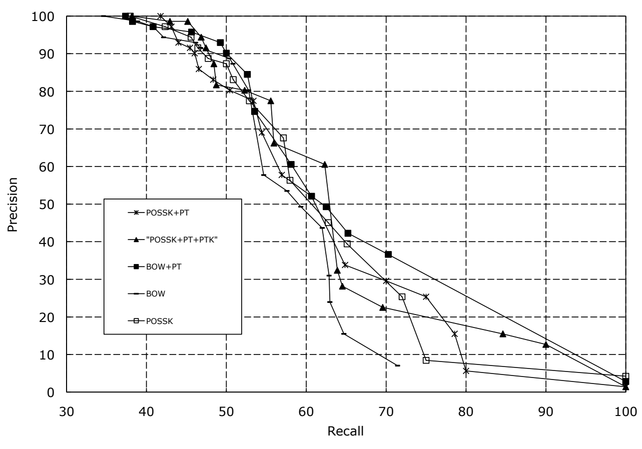
[8]All additive kernels are normalized to have a similarity score between 0 and 1, i.e. $K'(X_1, X_2) = \frac{K(X_1,X_2)}{\sqrt{K(X_1,X_1) \times K(X_2,X_2)}}$.

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

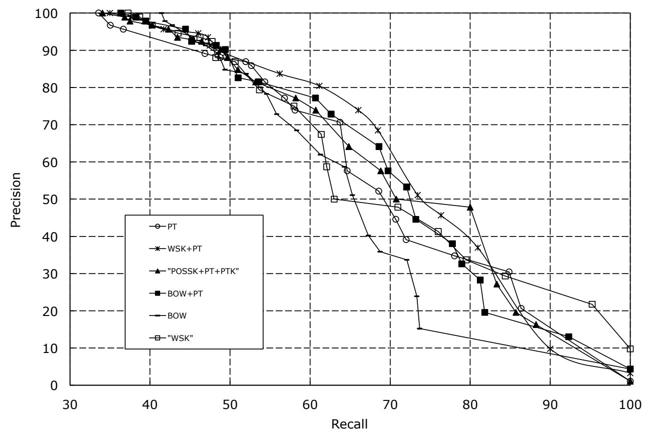
[10]Some of the kernels have been removed from the figures so that the plots result more visible.



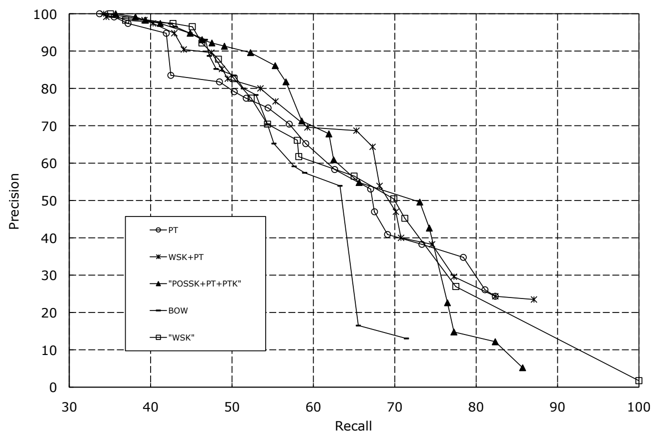
(a) Fold1



(b) Fold2



(c) Fold3



(d) Fold4

Figure 6: Precision/Recall curves of some kernel combinations over 4-folds of the WEB dataset.

The results on TREC in Figure 7 are more interesting. The contribution of BOW is always very low and thus the difference in accuracy with the other linguistic models is more evident. POS$_{SK}$ +PT+PAS$_{PTK}$, which encodes the most advanced syntactic and semantic information, shows a very high curve outperforming all the others (see folds 1, 3 and 4). Only, in Fold 2, the plots are close and POS$_{SK}$+PT shows a competitive curve.

The analysis of the above results suggests that: first as expected, BOW does not prove very relevant to learn re-ranking functions from examples; while it is useful to establish the initial ranking by measuring the similarity between question and answer, it is almost irrelevant to capture typical rules that suggest 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 conditions, we need to rely on syntactic structures which at least allow for detecting well formed descriptions.

Second, the results show that PT is important to detect typical description patterns but POS sequences also provide additional information since they are less sparse than tree fragments. Such patterns improve on the bag of POS-tags by about 4% (see POS vs POS$_{SK}$ on TREC data). This is a relevant result considering that in standard text classification bigrams or trigrams are usually ineffective.

Third, although POS$_{SK}$+PT generates a very rich feature set, i.e. POS patterns provided by SK and tree fragments generated by STK, PAS$_{PTK}$ is still able to significantly improve 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:

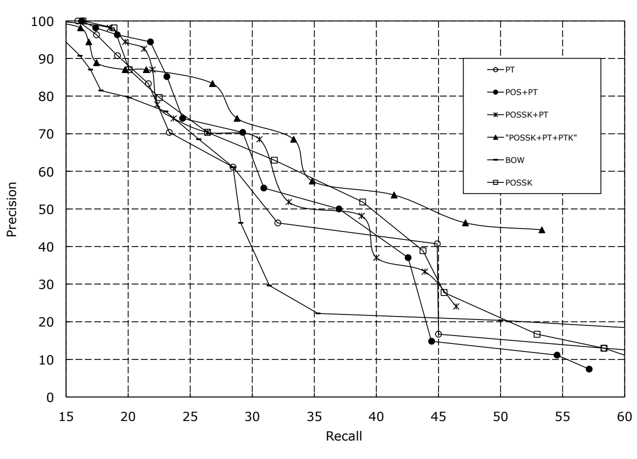
- `A1(X) R-A1(that) rel(result) A1(Y)`
- `A1(X) rel(characterize) A0(Y)`,

where X and Y need not necessarily be matched.

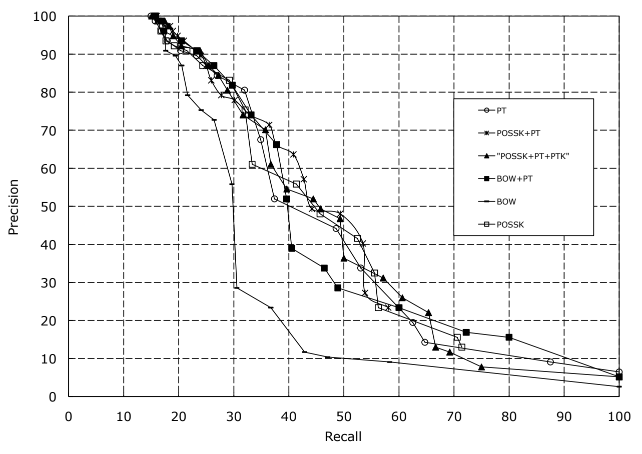
Finally, the best model, POS$_{SK}$+PT+PAS$_{PTK}$, improves on BOW by 61%. This is strong evidence that complex natural language tasks require advanced linguistic information that should be exploited by powerful algorithms such as SVMs and by effective feature engineering techniques such as kernel methods.

4.4 Re-ranking Question Answering Output

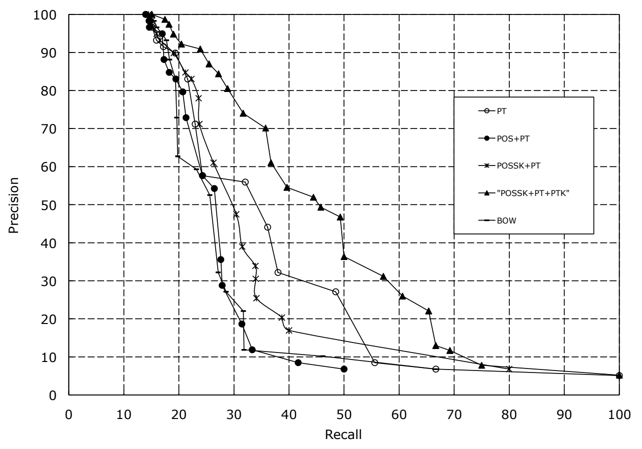
In these experiments, we assess the impact of the binary QA classifiers learned during the previous experiments when used as re-rankers. Our classifier-based re-ranking algorithm proceeds as follows. Starting from the top answer in the ranked list elaborated by BQAS, if the answer is classified as correct by the learned classifier, its rank is unchanged; otherwise the answer is pushed down, until the next answer labeled as incorrect according to the classifier is found.



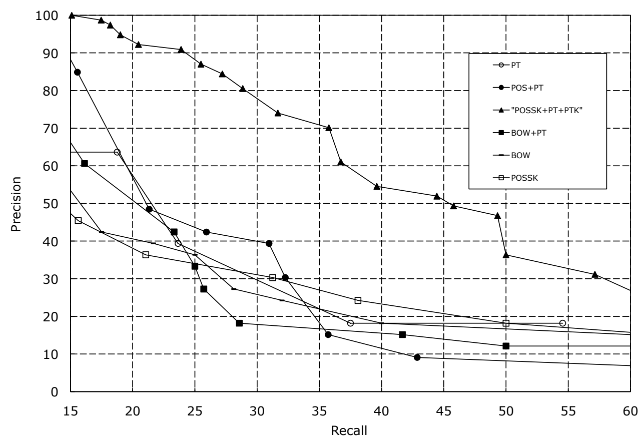
(a) Fold1



(b) Fold2



(c) Fold3



(d) Fold4

Figure 7: Precision/Recall curves of some kernel combinations over 4-folds of the TREC dataset.

Tables 2 and 3 report the classification $F1$ of the answer classifier (Column $Class.\ F1$) and the MRR of the QA systems. In case of IR and BQAS, $Class.\ F1$ is equal to the accuracy, i.e. the number of retrieved correct answers divided by the number of retrieved answers, whereas in case of the Re-ranker rows, $Class.\ F1$ refers to the question/answer classifier F1 (using BOW and the best model, respectively). Moreover, for the re-rankers, the MRR column refers to BQAS to which the BOW or the best re-rankers are applied.

From the $Class.\ F1$ column, it is visible that BQAS improves on the baseline IR engines, but the learned Q/A classifiers reach an almost double F1, i.e. 68.2 vs 36.8 on the WEB dataset (Table 2) and 39.1 vs 22.9 on the TREC dataset (Table 3). This suggests that re-rankers based on the classifiers draw more information from the available data with respect to BQAS. However, the real test of the usefulness of the binary classifiers learned in the previous sections must be measured in terms of the improvement of MRR of BQAS after answer re-ranking.

In the MRR column of Table 2, we first report our baseline ranking performance for the WEB dataset. We see that the IR engine (Google) ranking is outperformed by BQAS since Google ranks is based on the entire documents rather than the more accurate single passages. In the last 2 rows, we show the MRR of the re-rankers obtained by using the BOW and the best models (see Section 4.3) for re-ranking, respectively. With the BOW model, the re-ranker achieves 77.4%, i.e. an improvement of 21% over the ranking of BQAS; however, when the best re-ranking algorithm is applied, we achieve 81.1%, i.e. an impressive 25% of improvement over the baseline (Table 2). This shows that the additional structural information provided by the best model improves the performance of the re-ranker, which, in turn, improves the QA system.

In the TREC dataset (Table 3), the IR engine (Lucene) is also remarkably outperformed by BQAS; however, the BOW re-ranker yields a further improvement despite the fact that the corresponding F1 measure is not extraordinary. Furthermore, the best re-ranker model produces an improvement of about 4% over BQAS, i.e. a further 2% improvement over the BOW re-ranker (Table 3). The lower improvement on MRR of the best TREC re-ranker can be explained by its lower absolute classification accuracy than the Web re-ranker. This is due to the higher complexity of the TREC dataset.

5. RELATED WORK

Early work on using syntax and semantics in Information Retrieval was carried out in [39, 40, 34] and in [36, 35]. The results showed that the use of advanced linguistic information is not effective for document retrieval. In contrast, Question Answering work shows that semantic and syntax are essential to retrieve punctual answers, e.g [13, 41, 33]. However, successful approaches in

	$Class.\ F1$	MRR
IR (Google)	35.9±4.0	49.0±3.8
BQAS	36.8±3.6	56.2±3.2
Re-ranker (BOW)	65.3±2.9	77.4±2.7
Re-ranker (best)	68.2±4.3	81.1±2.1

Table 2: Classifier F1 and MRR (± Std. Dev.) of IR engine, BQAS, BOW resp. best re-ranker in WEB

	$Class.\ F1$	MRR
IR (Lucene)	21.3±1.0	16.2±3.4
BQAS	22.9±1.5	30.3±8.9
Re-ranker (BOW)	24.2±3.1	32.8±7.7
Re-ranker (best)	39.1±6.9	34.2±10.6

Table 3: Classifier F1 and MRR (± Std. Dev.) of IR engine, BQAS BOW resp. best re-rankers in TREC

TREC are based on many interconnected modules exploiting complex heuristics and fine tuning. The effective combination of such modules strongly depends on a manual setting, which is not often discussed or published. This prevents us from deriving general and useful findings on the use of natural language processors to model syntactic and semantic structures for retrieval tasks.

In our study, we avoid this problem by focusing on only one module of Question Answering, which can actually be seen as a typical Text Categorization task, i.e. the classification of pairs of text fragments constituted by question and answer. Since some kinds of questions can be solved with relatively simple representations, i.e. without the use of syntactic and semantic structures, we focus on the more complex task of processing definitional questions [3, 6, 31, 2, 28, 24]. In particular, the last four articles use predicate argument structures for re-ranking the answer lists, reporting significant improvement.

To our knowledge, our work in [28] is the only one using kernel methods for answer re-ranking. We used a syntactic tree kernel and a shallow semantic tree kernel based on predicate argument structures for re-ranking design. However, we only experimented with a Question Answering corpus derived from Web documents and the reported improvement, although significant, did not justify the adoption of relatively computationally expensive approaches like SVMs and kernel methods. In this paper, we have experimented with many more kernel types and with both Web and TREC documents and we could show that the potential improvement reachable by our approach is much higher (about 61% over BOW). Moreover, we have designed a faster kernel for the processing of semantic information.

In summary, the main property of our approach with respect to previous work on the use of syntactic and semantic structures is that we can define them without requiring a thorough linguistic analysis. We do not carry out feature engineering since we simply let kernel functions generate a large feature set (tree fragments or substrings) which effectively represent the semantic/syntactic information. The feasibility of this approach is due to the SVM theory which makes the learning algorithm robust to many irrelevant features (often produced by the NLP errors).

6. CONCLUSION

In this paper, we study 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 automatic question/answer pair classifiers. In addition, we investigate the role that syntax and semantics can play in modern Information Retrieval systems. These binary classifiers can select the correct answers from those provided by a basic QA system, thus improving the final system accuracy.

Our learning framework is constituted by Support Vector Machines (SVMs) and kernel methods applied to automatically generated syntactic and semantic structures. On the one hand, SVMs are robust to irrelevant/noisy features, which are inevitably contained in automatically generated linguistic structures. On the other hand, kernel methods allow for the extraction of many features from structured data, thus alleviating the manual design. In particular, we designed (i) a new shallow semantic kernel which is faster and more accurate than those previously proposed; (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. In addition, we tested the above models on two different question classification corpora derived from Web and TREC documents, respectively, by also measuring their impact on a basic QA system.

The results suggest that:

- the new kernel for processing PASs is more efficient and effective than previous models so that it can efficiently be used in answer re-ranking systems.
- Kernels based on PAS, POS-tag sequences and syntactic parse trees improve on BOW on both datasets. On the TREC data the improvement is interestingly high, e.g. about 61%, making their application in retrieval systems worthwhile. Note that this goes far beyond our previous findings since in them we only observed a small accuracy increase over Web documents.
- Error analysis revealed that Web documents are slightly misleading in determining the role of linguistic structures since they contain many extra-linguistic patterns, e.g. "Learn more about X" which are easily captured by BOW. Thus, although the correctness of a definition does not *heavily* depend on words but rather on structures, BOW model is needed to carry out an important first level of answer filtering.
- Our best question/answer classifier, used as re-ranker, significantly improves the QA system accuracy, demonstrating its promising applicability.

Finally, the mathematical elegance of kernel methods allow for the separation of data, e.g. our presented linguistic structures, which are easily understood, from the feature space, which is automatically generated. The latter may be complex to study since it is implicitly generated but at the same time it is easy to use thanks to the availability of basic kernel functions for structured data.

Acknowledgments

I would like to thank the anonymous reviewers for their professional and competent reviews. Particular thanks are due to Silvia Quarteroni who carried out the data extraction and the evaluation of the QA systems. Many thanks to my friends, Sindy and Hatim, to have ruined their Friday in correcting the syntax of an early draft of this paper. This work has been partially supported by the European Commission - LUNA project, contract n. 33549.

7. REFERENCES

[1] J. Allan. Natural language processing for information retrieval. In *NAACL/ANLP (tutorial notes)*, 2000.

[2] M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured retrieval for Question Answering. In *Proceedings of ACM SIGIR*, 2007.

[3] S. Blair-Goldensohn, K. R. McKeown, and A. H. Schlaikjer. Answering definitional questions: A hybrid approach. In

M. Maybury, editor, *New Directions In Question Answering*. AAAI Press, 2004.

[4] N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. Word sequence kernels. *JMLR*, 3:1059–1082, 2003.

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

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

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

[8] K. Collins-Thompson, J. Callan, E. Terra, and C. L. Clarke. The effect of document retrieval quality on factoid QA performance. In *SIGIR'04*, New York, NY, USA, 2004.

[9] H. Cui, M. Kan, and T. Chua. Generic soft pattern models for definitional QA. In *SIGIR'05*, Salvador, Brazil, 2005. ACM.

[10] A. Culotta and J. Sorensen. Dependency Tree Kernels for Relation Extraction. In *ACL04*, pages 423–429, Barcelona, Spain, 2004.

[11] C. Cumby and D. Roth. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*, pages 107–114, Washington, DC, USA, 2003.

[12] J. Furnkranz, T. Mitchell, and E. Rilof. 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*, 1998.

[13] A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. Question answering with LCC's CHAUCER at TREC 2006. In *Proceedings of TREC'06*, 2006.

[14] R. Jackendoff. *Semantic Structures*. MIT Press, 1990.

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

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

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

[18] H. Kazawa, H. Isozaki, and E. Maeda. NTT Question Answering system in TREC 2001. In *TREC'01*, 2001.

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

[20] T. Kudo and Y. Matsumoto. Fast Methods for Kernel-Based Text Analysis. In E. Hinrichs and D. Roth, editors, *Proceedings of ACL*, pages 24–31, 2003.

[21] T. Kudo, J. Suzuki, and H. Isozaki. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*, Ann Arbor, Michigan, 2005.

[22] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92*, pages 37–50, Kobenhavn, DK, 1992.

[23] X. Li and D. Roth. Learning question classifiers: the role of semantic information. *Journ. Nat. Lang. Eng.*, 2005.

[24] M. Surdeanu, M. Ciaramita and H. Zaragoza. Learning to rank answers on large online QA collections. In *ACL*, 2008.

[25] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML'06*, 2006.

[26] A. Moschitti and R. Basili. Complex linguistic features for text classification: A comprehensive study. In S. McDonald and J. Tait, editors, *ECIR, Sunderland, UK*, 2004.

[27] A. Moschitti, B. Coppola, A. Giuglea, and R. Basili. Hierarchical semantic role labeling. In *CoNLL 2005 shared task*, 2005.

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

[29] S. Narayanan and S. Harabagiu. Question Answering based on semantic structures. In *COLING'04*, Geneva, Switzerland, 2004.

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

[31] D. Shen and M. Lapata. Using semantic roles to improve Question Answering. In *Proceedings of EMNLP-CoNLL*, 2007.

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

[33] S. Small, T. Strzalkowski, T. Liu, S. Ryan, R. Salkin, N. Shimizu, P. Kantor, D. Kelly, and N. Wacholder. Hitiqa: Towards analytical Question Answering. In *Proceedings of COLING 2004*, 2004.

[34] A. F. Smeaton. Using NLP or NLP resources for information retrieval tasks. In T. Strzalkowski, editor, *Natural language information retrieval*, pages 99–111. Kluwer Academic Publishers, Dordrecht, NL, 1999.

[35] T. Strzalkowski, J. P. Carballo, J. Karlgren, A. H. P. Tapanainen, and T. Jarvinen. Natural language information retrieval: TREC-8 report. In *Text REtrieval Conference*, 1999.

[36] T. Strzalkowski, G. C. Stein, G. B. Wise, J. P. Carballo, P. Tapanainen, T. Jarvinen, A. Voutilainen, and J. Karlgren. Natural language information retrieval: TREC-7 report. In *Text REtrieval Conference*, pages 164–173, 1998.

[37] K. Toutanova, P. Markova, and C. Manning. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In D. Lin and D. Wu, editors, *Proceedings of EMNLP 2004*, pages 166–173, Barcelona, Spain, July 2004.

[38] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[39] E. M. Voorhees. Using wordnet to disambiguate word senses for text retrieval. In *Proceedings of ACM-SIGIR, Pittsburgh, PA, USA, 1993*, 1993.

[40] E. M. Voorhees. Query expansion using lexical-semantic relations. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of ACM-SIGIR*. ACM/Springer, 1994.

[41] E. M. Voorhees. Overview of the TREC 2001 Question Answering track. In *Proceedings of the Thirteenth Text REtreival Conference (TREC 2004)*, 2004.

[42] Y. Wu, R. Zhang, X. Hu, and H. Kashioka. Learning unsupervised SVM classifier for answer selection in Web Question Answering . In *Proceedings of EMNLP-CoNLL*, 2007.

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

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