Advanced Structural Representations for Question Classification and Answer Re-ranking

Silvia Quarteroni[1], Alessandro Moschitti[2], Suresh Manandhar[1], and Roberto Basili[2]

[1] The University of York, York YO10 5DD, United Kingdom
{silvia,suresh}@cs.york.ac.uk
[2] University of Rome "Tor Vergata", Via del Politecnico 1, 00133 Rome, Italy
{moschitti,basili}@info.uniroma2.it

Abstract. In this paper, we study novel structures to represent information in three vital tasks in question answering: question classification, answer classification and answer reranking. We define a new tree structure called PAS to represent predicate-argument relations, as well as a new kernel function to exploit its representative power. Our experiments with Support Vector Machines and several tree kernel functions suggest that syntactic information helps specific task as question classification, whereas, when data sparseness is higher as in answer classification, studying coarse semantic information like PAS is a promising research area.

1 Introduction

Question answering (QA) can be seen as a form of information retrieval where one or more answers are returned to a question in natural language in the form of sentences or phrases. The typical QA system architecture consists of three phases: question processing, document retrieval and answer extraction [1].

In question processing, useful information is gathered from the question and a query is created; this is submitted to an information retrieval engine, which provides a ranked list of relevant documents. From these, the QA system must extract one or more candidate answers, which can then be reranked following various criteria such as their similarity to the query. Question processing is usually centered around question classification (QC), the task that maps a question into one of k expected answer classes. Such task is crucial as it constrains the search space of possible answers and contributes to selecting answer extraction strategies specific to a given answer class. Most accurate QC systems apply supervised machine learning techniques, e.g. Support Vector Machines (SVMs) [2] or the SNoW model [3], where questions are encoded using various lexical, syntactic and semantic features; it has been shown that the question's syntactic structure contributes remarkably to the classification accuracy.

The retrieval and answer extraction phases consist in retrieving relevant documents [4] and selecting candidate answer passages [5,1] from them. A further phase called answer re-ranking is optionally applied. It is especially relevant in the case of non-factoid questions, such as those requiring definitions, where the answer can be a whole sentence or paragraph. Here, too, the syntactic structure of a sentence appears to provide more useful information than a bag of words.

An effective way to integrate syntactic structures in machine learning algorithms is the use of tree kernel functions [6]. Successful applications of these have been reported for question classification [2,7] and other tasks, e.g. relation extraction [8,7]. However, such an approach may be insufficient to encode syntactic structures in more complex tasks such as computing the relationships between questions and answers in answer reranking. The information provided by parse trees may prove too sparse: the same concept, expressed in two different sentences, will produce different, unmatching parses. One way to overcome this issue is to try to capture semantic relations by processing shallow representations like predicate argument structures proposed in the PropBank (PB) project [9] (`www.cis.upenn.edu/~ace`). We argue that semantic structures can be used to characterize the relation between a question and a candidate answer.

In this paper, we extensively study new structural representations, namely parse trees, bag-of-words, POS tags and predicate argument structures for question classification and answer re-ranking. We encode such information by combining tree kernels with linear kernels. Moreover, to exploit predicate argument information - which we automatically derive with our state-of-the-art software - we have defined a new tree structure representation and a new kernel function to process its semantics. Additionally, for the purpose of answer classification and re-ranking, we have created a corpus of answers to TREC-QA 2001 description questions obtained using a Web-based QA system.

Our experiments with SVMs and the above kernels show that (a) our approach reaches state-of-the-art accuracy on question classification and (b) PB predicative structures are not effective for question classification but show promising results for answer classification. Overall, our answer classifier increases the ranking accuracy of a basic QA system by about 20 absolute percent points.

This paper is structured as follows: Section 2 introduces advanced models to represent syntactic and semantic information in a QA context; Section 3 explains how such information is exploited in an SVM learning framework by introducing novel tree kernel functions; Section 4 reports our experiments on question classification, answer classification and answer reranking; Section 5 concludes on the utility of the new structure representations and sets the basis for further work.

2 Advanced Models for Sentence Representation

Traditionally, the majority of information retrieval tasks have been solved by means of the so-called bag-of-words approach augmented by language modeling [10]. However, when the task requires the use of more complex semantics the above approach does not appear to be effective, as it is inadequate to perform fine-level textual analysis. To overcome this, QA systems use linguistic processing tools such as syntactic parsers to produce sentence parse trees. In our study we exploited two sources of syntactic information: deep syntactic parsers – the outcome of a well-studied technology [6,11] – and shallow semantic parsers, only recently the object of a consistent body of work.

2.1 Syntactic Structures

The syntactic parse tree of a sentence is a hierarchical representation of the syntactic relationships between its words. In such tree, each node with its children is associated with a grammar production rule, where the symbol at the left-hand side corresponds to the parent and the symbols at the right-hand side are associated with the children. The terminal symbols of the grammar are always associated with the leaves of the tree.

Parse trees have often been applied in natural language processing applications requiring the use of grammatical relations, e.g. extraction of subject/object relations. It has been shown [2,7] that syntactic information outperformed bag-of-words and bag-of-n-grams on question classification in QA. The advantage of computing parse tree-based sentence similarity with respect to purely lexical approaches is that trees provide structural relations hard to compute otherwise.

However, when approaching complex QA tasks, the use of parse trees has some limitations. For instance in definitional QA candidate answers can be expressed by long and articulated sentences or even paragraphs. Since the information encoded in a parse tree is intrinsically sparse, it does not contribute well to computing the similarity between such answers; shallow semantics however, being a more "compact" source of information, could prevent the sparseness of deep structural approaches and the noise of bag-of-word models.

2.2 Semantic Structures

Initiatives such as PropBank (PB) [9] have led to the creation of vast and accurate resources of manually annotated predicate argument structures[1]. Using these, machine learning techniques have proven successful in Semantic Role Labeling (SRL), the task of attaching semantic roles to predicates and their arguments. SRL is a fully exploitable technology: our SVM-based SRL system achieves a 76% accuracy on PB data [12]. Attempting an application of SRL to QA seems natural, as pinpointing the answer to a question relies on a deep understanding of the question and answer's semantics.

Let us consider a typical PB annotation for a sentence, such as: `[`$_{ARG0}$ `Compounded interest] [`$_{pred}$ `computes] [`$_{ARG1}$ `the effective interest rate for an investment] [`$_{ARGM-TMP}$ `during the current year].`

Such shallow semantic annotation is a useful information source. For instance, the PB annotation of a similar sentence would be: `[`$_{ARGM-TMP}$ `In a year][`$_{ARG1}$ `the bank interest rate] is [`$_{pred}$ `evaluated] by [`$_{ARG0}$ `compounded interest].` Such annotations can be represented via tree structures as those in Figure 1, which we call PASs. These attempt to capture the semantics of both sentences.

[1] The PB corpus contains 300,000 words annotated with predicative information on top of the Penn Treebank 2 Wall Street Journal texts. For each predicate, the expected arguments are labeled sequentially from $ARG0$ to $ARG5$, $ARGA$ and $ARGM$.

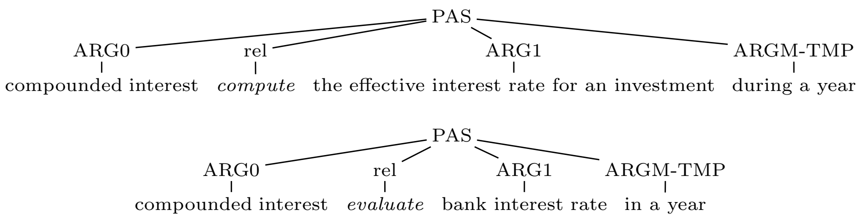

Fig. 1. Predicate argument structures of two sentences expressing similar semantics

We can improve such representation by substituting the arguments with their most important word – often referred to as the semantic head – as in Figure 2. It seems intuitive that data sparseness can be remarkably reduced by using this shallow representation instead of the BOW representation.

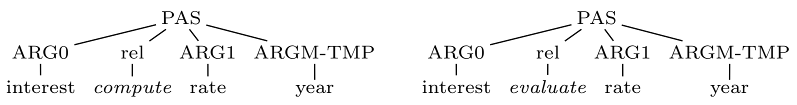

Fig. 2. Improved predicate argument structures of two different sentences

Knowing that parse trees and PASs may improve the simple BOW representation, we face the problem of representing tree structures in learning machines. Section 3 introduces a viable representation approach based on tree kernels.

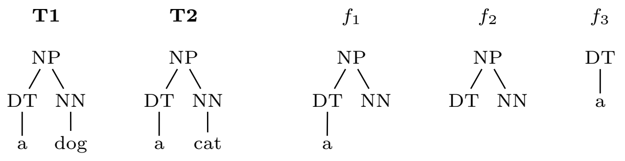

Fig. 3. T1 and T2 with their fragments f_1, f_2 and f_3 derived by the kernel function

3 Syntactic and Semantic Tree Kernels

As mentioned above, encoding syntactic/semantic information represented by means of tree structures in the learning algorithm is problematic. One possible solution is to use as features of a structure all its possible substructures. Given the combinatorial explosion of considering the subparts, the resulting feature space is usually very large. To manage such complexity we can define kernel functions that implicitly evaluate the scalar product between two feature vectors without explicitly computing such vectors.

Below, we report the tree kernel function devised in [6] computing the number of common subtrees between two syntactic parse trees and a new version evaluating the number of semantic structures shared between two PASs.

3.1 Syntactic Tree Kernel

Given two trees T_1 and T_2, let $\{f_1, f_2, ..\} = \mathcal{F}$ be the set of substructures (fragments) and $I_i(n)$ be equal to 1 if f_i is rooted at node n, 0 otherwise. We define

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \tag{1}$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. The latter is equal to the number of common fragments rooted in nodes n_1 and n_2. We can compute Δ as follows:

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

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(c_{n_1}^j, c_{n_2}^j)) \tag{2}$$

where $nc(n_1)^2$ is the number of children of n_1 and c_n^j is the j-th child of node n. As proved in [6], the above algorithm allows to evaluate Eq. 1 in $O(|N_{T_1}| \times |N_{T_2}|)$. A decay factor λ is usually added by changing the formulae in (2) and (3) to^3:

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

For instance, Figure 3 shows two trees and the substructures they have in common. It is worth to note that the fragments of the above Syntactic Tree Kernel (STK) are such that any node contains either all or none of its children. Consequently, `[NP [DT]]` and `[NP [NN]]` are not valid fragments. This limitation makes it unsuitable to derive important substructures from the PAS tree. The next section shows a new tree kernel that takes this into account.

3.2 Semantic Tree Kernel

As mentioned above, the kernel function introduced in Section 2 is not sufficient to derive all the required information from trees such as the PAS in Fig. 2: we would like to have fragments that contain nodes with only part of the children, e.g. to neglect the information constituted by ARGM-TMP. For this, we need to slightly modify the PAS and to define a new kernel function.

First, we change the PAS into the PAS+ structure as shown in Figure 2(a). Each slot node accommodates an argument label in the natural argument order.

2 Note that, since the productions are the same, $nc(n_1) = nc(n_2)$.

3 A normalization in the kernel space, i.e. $K'(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1) \times K(T_2, T_2)}}$, ensures a similarity score between 0 and 1.

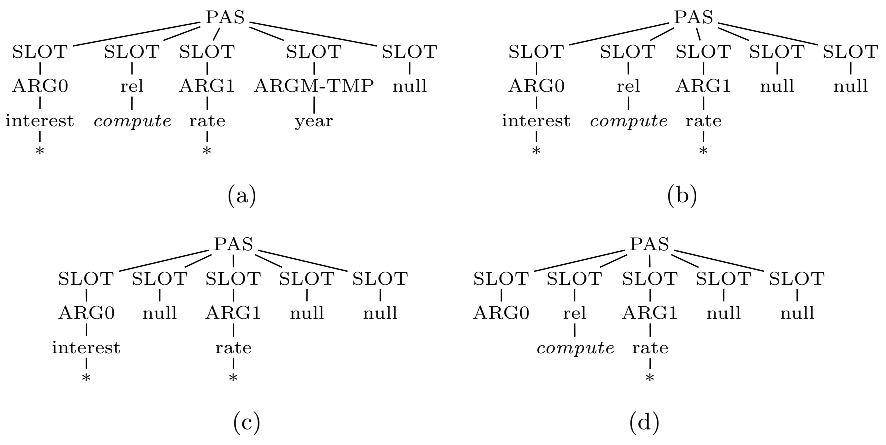

Fig. 4. A PAS+ with some of its fragments

Since diverse predicates or their different use may involve a different number of arguments, we provide additional slots, filled with *null* arguments. The figure shows just one slot to complete a structure of 5 arguments. More slots can be added to manage the maximum number of arguments that a predicate can have. The leaf nodes are filled with a wildcard character, i.e. *. They may alternatively accommodate additional information. The slot nodes are used in such a way that the adopted tree kernel function can generate fragments containing one or more children like for example those shown in frames (b), (c) and (d). As previously pointed out, if the arguments were directly attached to the root node, the kernel function would only generate the structure with all children (or the structure with no children, i.e. empty).

Second, we observe that the above approach generates many matches with slots filled with the null label. To solve this problem, we have set a new step 0:

0. if n_1 (or n_2) is a pre-terminal node and its child label is *null*, $\Delta(n_1, n_2) = 0$;

and by subtracting one unit to $\Delta(n_1, n_2)$, in step 3:

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

The new Δ in Eq. 1 defines a new kernel[4] that we call Shallow Semantic Tree Kernel (SSTK).

4 Experiments

The purpose of our experiments is to study the impact of the new structure representations introduced earlier for QA tasks. In particular, we focus on question classification and answer reranking for Web-based QA systems.

[4] By induction, it can be proven that SSTK applied to PAS+ generates the space of all possible k-ary relations derivable from a set of k arguments.

In the question classification (QC) task, we extend previous studies, e.g. [2,7], by testing a set of previously designed kernels and their combination with our new Shallow Semantic Kernel. SVMs are the learning machines used to build the multi-class classifiers based on the SSK, the kernel combinations being the sum of the individual models. This operation always produces a valid kernel [13].

In the answer reranking task, we approach the problem of detecting description answers (among the most complex in the literature [14,15]). We learn binary answer classifiers based on question-answer pairs constructed by querying our Web QA system, YourQA [16], with the same questions as the test set used in the QC experiment. Our experiments with different kernel combinations on question-answer pairs allow us to select the best performing classifier, which in turn is used to re-rank answers. The resulting ranking is compared with the ranking provided by Google and by YourQA.

4.1 Question Classification

As a first experiment, we focus on question classification (QC), because of its great impact on the quality of a QA system and because it is a widely approached task for which benchmarks and baseline results are available [2,3].

QC is defined as a multi-classification problem which consists in assigning an instance I to one of n classes, which generally belong to two types: factoid, seeking short fact-based answers (e.g. name, date) or non-factoid, seeking e.g. descriptions or definitions (see e.g. the taxonomy in [3]). We design a question multi-classifier by using n binary SVMs combined according to the ONE-vs-ALL scheme, where the final output class is the one associated with the most probable prediction. Question representation is based on the following features/structures: parse tree (PT), bag-of-words (BOW), bag-of-POS tags (POS) and predicate argument structure (PAS). We implemented the proposed kernels in the SVM-light-TK software available at `ai-nlp.info.uniroma2.it/moschitti/` which encodes the tree kernel functions in SVM-light [17][5]. The PAS structures were automatically derived by our SRL system [12].

As benchmark data, we use the question training and test set available at: `l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`, where the test set are the TREC 2001 test questions[6] [18]. We refer to this split as UIUC.

The performance of the multi-classifier and the individual binary classifiers is measured with accuracy resp. F1-measure. To collect more statistically significant information, we run 10-fold cross validation on the 6,000 questions.

Question Classification Results. Table 1.(a) shows the accuracy of different question representations on the UIUC split (Column 1) and the average accuracy

[5] We adopted the default regularization parameter (i.e., the average of $1/||\boldsymbol{x}||$) and tried a few cost-factor values to adjust the rate between Precision and Recall on the development set.

[6] The benchmark is manually partitioned according to the coarse-grained question taxonomy defined in [3] – i.e. ABBR, DESC, NUM, HUM, ENTY and LOC – and contains 5,500 training and 500 test instances.

± standard deviation on the cross validation splits (Column 2) whereas Table 1.(b) reports the F1 for the individual classes using the best model, PTBOW. The analysis of the above data suggests that:

Firstly, the STK on PT and the linear kernel on BOW produce a very high result, i.e. about 90.5%. This is higher than the best outcome derived in [2], i.e. 90%, obtained with a kernel combining BOW and PT. When our BOW is combined with STK, it achieves an even higher result, i.e. 91.8%, very close to the 92.5% accuracy reached in [3] by using complex semantic information derived manually from external resources. Our higher results with respect to [2] are explained by a highly performing BOW, the use of parameterization and most importantly the fact that our model is obtained by summing two separate kernel spaces (with separate normalization), as mixing BOW with tree kernels does not allow SVMs to exploit all its representational power.

Secondly, model PTBOW shows that syntactic information can be beneficial in tasks where text classification is vital, such as QA. Here, syntax can give a remarkable contribution in determining the class of a question; moreover, the lexical information (BOW) has a limited impact due to the little number of words forming a question.

Thirdly, the PAS feature does not provide improvement. This is mainly due to the fact that at least half of the training and test questions only contained the predicate "to be", for which a PAS cannot be derived by our PB-based shallow semantic parser. Also, PT probably covers most of the question's semantic information encoded by PAS.

Next, the 10-fold cross-validation experiments confirm the trends observed in the UIUC split. The best model is PTBOW which achieves an average accuracy of 86.1%. This value is lower than the one recorded for the UIUC split. The explanation is that the test set in UIUC is not consistent with the training set (it contains the TREC 2001 questions) and it includes a larger percentage of easily classified question types, e.g. the numeric (22.6%) and description classes (27.6%) while their percentage in training is 16.4% and 16.2%, respectively. This shows the importance of cross-validation results that, given the very low values the standard deviation, also suggest that the superior accuracy of the PTBOW over the BOW model is statistically significant.

Finally, for individual binary classification, the most accurate is the one carried out for NUM, which generally exhibits easily identified cues such as "how much/many". The more generic ENTY type proves hardest in both the UIUC and cross-validation experiments, while LOC and HUM remain well-classified in both cases also thanks to their regular patterns ("where" and "who" identifiers).

4.2 Answer Classification and Reranking

Question Classification does not allow to fully exploit the predicate argument potential since questions tend to be short and with no predicates. A different scenario is answer classification, i.e. deciding if a passage/sentence correctly answers the question: here, the semantics that the classifier has to generate are

Table 1. Accuracy of the question classifier with different feature combinations and performance of the best classifier by question class

(a)

Features	Acc (UIUC)	Acc (xval.)
PT	90.4	84.8±1.4
BOW	90.6	84.7±1.4
PAS	34.2	43.0±2.2
POS	26.4	32.4±2.5
PTBOW	**91.8**	**86.1±1.3**
PTBOWPOS	91.8	84.7±1.7
PASBOW	90.0	82.1±1.5
PASBOWPOS	88.8	81.0±1.7

(b)

Q. class	P (UIUC)	R (UIUC)	F1 (UIUC)	F1 (xval.)
ABBR	87.5	77.8	82.4	78.5± 7.0
DESC	95.8	99.3	97.5	84.6±2.3
ENTY	73.6	83.0	78.0	75.7±1.3
HUM	89.6	92.3	90.9	86.8±2.0
LOC	86.6	85.2	85.7	88.9±1.5
NUM	99.0	86.7	92.5	94.2±1.4
Multi-Class. Accuracy			91.8	86.1±1.3

not constrained to a small taxonomy and the length of an answer may make the representation based on PT too sparse.

We learn answer classification with a binary SVM which determines if a answer is correct for the target question: consequently, the classification instances are the ⟨question, answer⟩ pairs. Each pair component can be encoded with PT, BOW, POS and PAS representations and processed with the previouskernels.

The output of the binary classifier can be used to rerank the list of candidate answers of a QA system. Starting from the top answer, each instance is classified based on its correctness with respect to the question. If it is classified as correct its rank is unchanged; otherwise it is pushed down, until a lower ranked incorrect answer is found.

As output of the basic QA we use Google rank along with the YourQA [16] system. YourQA uses the Web documents corresponding to the top 20 Google results for the question. Then, each sentence in each document is compared to the question to compute the Jaccard similarity, which, in the answer extraction phase, is used to select the most relevant sentence. A passage of up to 750 bytes is then created around the sentence and returned as an answer.

As test data, we collected the 138 TREC 2001 test questions labeled as "description" and for each, we obtained a list of answer paragraphs extracted from Web documents using YourQA. Each paragraph sentence was manually evaluated according to whether it contained an answer to the corresponding question; moreover, to simplify the classification problem, we isolated for each paragraph the sentence which obtained the maximal judgment (in case more than one sentence in the paragraph had the same judgment, we chose the first one). We collected a corpus containing 1123 sentences, 401 of which – labeled as "+1" – answered the question either concisely or with noise; the rest – labeled as "-1"– were either irrelevant to the question or contained hints relating to the question but could not be judged as valid answers[7].

[7] 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".

Answer Classification and Reranking Results. To gather statistically significant data, we ran 5-fold cross-validation, with the constraint that two pairs $\langle q, a_1 \rangle$ and $\langle q, a_2 \rangle$ associated with the same question q could not be split between training and testing. The answer classification experiment results are in Tab. 2.

We note that: first, the contribution of the POS feature in answer classification is much higher than in question classification and even outperforms the PT feature (see Table (a)). This is because on the one side we work with Web data, a noisy input wich can drastically reduce parser performance; on the other, POS tagging is a more robust operation and yields less errors. Moreover, while question classification is a multi-classification task where the POS feature must be used to determine a semantic category, definition answer classification is a binary classification task – hence statistically simpler.

Second, although the accuracy of PAS as a standalone was inferior to that of PT, when coupled with BOW it yielded higher accuracy[8]; in this case, its ability to generalize the answer information allowed to overcome the erroneous/noisy information provided by the PT on Web data.

Table 2. Answer classifier with different feature combinations, baseline classifiers accuracy and MRR of the best reranker compared to the baseline

(a)

Features	P	R	F1
PT	56.4	70.0	59.6±4.0
BOW	58.5	85.9	69.3±6.6
POS	52.4	84.1	64.0±5.9
PAS	52.4	71.1	58.6±5.6
PTBOW	59.8	79.7	68.1±8.0
PASBOW	**64.1**	**79.2**	**70.7±5.9**
PTBOWPOS	63.8	71.7	67.4±7.6
PASBOWPOS	64.4	75.2	69.2± 6.5

(b)

Baseline	P	R	F1
Gg@1	39.7	9.4	15.2±3.1
QA@1	45.3	10.9	17.6±2.9
Gg@all	35.8	100	52.7±6.2
QA@all	35.8	100	52.7±6.2

	Gg	QA	Reranker
MRR	54.8±6.7	60.1±4.1	79.2±0.9

Third, we compared the answer classifier with two baselines built using the YourQA and Google rankings. For this, we considered the top N ranked results as correct definitions and the remaining ones as incorrect for different values of N. Table 2.(b) shows the results for $N = 1$ and the maximum N (all), i.e. all the available answers. Each measure is the average of the Precision, Recall and F1 of the three systems on the cross-validation splits. The F1 of Google (Gg) and YourQA (QA) are greatly outperformed by the classifier, even when all answers are considered ($N = all$) and the low standard deviations ensure the statistical relevance of the results.

[8] Although the standard deviation in this case is high, as the complexity can vary across splits, since the PAS and PASBOW models are similar, the standard deviation of their difference is lower, i.e. 2.03. When we performed the t-test on such value, we confirmed that PASBOW is superior to BOW with a 90% level of confidence.

Finally, we implemented the simple re-ranking algorithm described previously and assessed its performance with the MRR[9] metric adopted in TREC 2001[10].

YourQA's MRR outperforms the Google MRR (Tab. 2.(b), last row) as Google ranks are based on whole documents, not on single passages, so documents where no passage contains all of the question's keywords may be ranked higher than others containing them all. When the answer classifier is applied to improve the QA ranking, MRR reaches .792, rising by nearly 20 points.

Related Work on Definitional QA. Unfortunately, no results are known to the authors concerning a Web-based answer classifier for the same question set and few are available on the performances computed over description questions alone on the NIST corpus; for instance, NTT's system achieved an MRR of .247 on description questions using a heuristic searching for appositions [15].

Interesting related work on definition answer reranking [20] was conducted by comparing the use of an SVM classifier predictions to induce a ranking and of the Ranking SVM algorithm [17]. In [21], ranks were computed based on the probabilities of biterm language models generating candidate answers.

5 Conclusion

In this paper, we introduce new structures to represent textual information in three question answering tasks: question classification, answer classification and answer reranking. We define a new tree structure called PAS to represent predicate-argument relations, which we automatically extract using our SRL system. We also introduce a new kernel function to exploit its representative power.

Our experiments with SVMs and such new functions suggest that syntactic information helps specific tasks such as question classification. On the other hand, the coarse-grained semantic information contained by the PAS gives promising results in answer classification, which suffers more from data sparseness. Moreover, our simple answer reranker, based on the answer classifier output, obtains a 20% more accurate ranking than our baseline QA system.

In the future, we will study the utility of PASs for other tasks affected by noisy data and apply a true SVM reranker trained with the proposed information.

References

1. Kwok, C.C.T., Etzioni, O., Weld, D.S.: Scaling question answering to the web. In: WWW. (2001)
2. Zhang, D., Lee, W.S.: Question classification using support vector machines. In: Proceedings of SIGIR, ACM Press (2003)

[9] The Mean Reciprocal Rank is defined as: $MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{rank_i}$, where n is the number of questions and $rank_i$ is the rank of the first correct answer to question i.

[10] Although since the TREC 2003 definition track [19] answers were expected in the form of bags of information "nuggets", we still believe the MRR meaningful for QA.

3. Li, X., Roth, D.: Learning question classifiers: The role of semantic information. Journal of Natural Language Engineering (2005)
4. Collins-Thompson, K., Callan, J., Terra, E., Clarke, C.L.: The effect of document retrieval quality on factoid question answering performance. In: Proceedings of SIGIR, New York, NY, USA, ACM Press (2004)
5. Pasca, M.: Open-Domain Question Answering from Large Text Collections. CSLI Studies in Computational Linguistics (2003)
6. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: ACL. (2002)
7. Moschitti, A.: Efficient convolution kernels for dependency and constituent syntactic trees. In: Proceedings of ECML. (2006)
8. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. JMLR (2003)
9. Kingsbury, P., Palmer, M.: From treebank to propbank. In: Proceedings of LREC. (2002)
10. Allan, J., et al.: Challenges in information retrieval and language modeling. In: Workshop at University of Amherst. (2002)
11. Charniak, E.: A maximum-entropy-inspired parser. In: Proceedings of NAACL. (2000)
12. Moschitti, A., Coppola, B., Giuglea, A.M., Basili, R.: Hierarchical semantic role labeling. In: Proceedings of the CoNLL 2005 shared task. (2005)
13. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
14. Cui, H., Kan, M.Y., Chua, T.S.: Generic soft pattern models for definitonal question answering. In: Proceedings of SIGIR. (2005)
15. Kazawa, H., Isozaki, H., Maeda, E.: NTT question answering system in TREC 2001. In: Proceedings of TREC. (2001)
16. Quarteroni, S., Manandhar, S.: User modelling for adaptive question answering and Information Retrieval. In: Proceedings of FLAIRS. (2006)
17. Joachims, T.: Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. (1999)
18. Voorhees, E.M.: Overview of the TREC 2001 QA track. In: TREC. (2001)
19. Voorhees, E.M.: Overview of TREC 2003. In: TREC. (2003)
20. Xu, J., Cao, Y., Li, H., Zhao, M.: Ranking definitions with supervised learning methods. In: Special interest tracks and posters of WWW, New York, NY, USA, ACM Press (2005)
21. Chen, Y., Zhou, M., Wang, S.: Reranking answers from definitional question answering using language models. In: Proceedings of ACL. (2006)