Structural Relationships for Large-Scale Learning of Answer Re-ranking

Aliaksei Severyn  
Department of Computer Science and Information Engineering  
University of Trento  
38123 Povo (TN), Italy  
severyn@disi.unitn.it

Alessandro Moschitti  
Department of Computer Science and Information Engineering  
University of Trento  
38123 Povo (TN), Italy  
moschitti@disi.unitn.it

ABSTRACT

Supervised learning applied to answer re-ranking can highly improve on the overall accuracy of question answering (QA) systems. The key aspect is that the relationships and properties of the question/answer pair composed of a question and the supporting passage of an answer candidate, can be efficiently compared with those captured by the learnt model. In this paper, we define novel supervised approaches that exploit structural relationships between a question and their candidate answer passages to learn a re-ranking model. We model structural representations of both questions and answers and their mutual relationships by just using off-the-shelf shallow syntactic parser. We encode structures in Support Vector Machines (SVMs) by means of sequence and tree kernels, which can implicitly represent question and answer pairs in huge feature spaces. Such models together with the latest approach to fast kernel-based learning enabled the training of our rerankers on hundreds of thousands of instances, which previously rendered intractable for kernelized SVMs. The results on two different QA datasets, e.g., Answerbag and Jeopardy! data, show that our models deliver large improvement on passage re-ranking tasks, reducing the error in Recall of BM25 baseline by about 18%. One of the key findings of this work is that, despite its simplicity, shallow syntactic trees allow for learning complex relational structures, which exhibits a steep learning curve with the increase in the training size.

Categories and Subject Descriptors

I.2.7 [Natural Language Processing]: [Language parsing and understanding, Text analysis]

General Terms

Algorithms, Experimentation

Keywords

Question Answering, Kernel Methods, Large-Scale Learning, Support Vector Machines, Structural Kernels

1. INTRODUCTION

A critical step for the design of Question Answering (QA) systems is the passage retrieval and ranking module as even the most powerful approach to answer extraction would inevitably fail if applied to an input passage that either contains no correct answer or simply is not adequate for supporting its correctness. Traditional approaches exploit similarity measures between a question and an answer passage but of course they show obvious limitations, e.g., the passage: Ron Paul is not the president of the U.S., is neither the correct passage nor contains the right answer to the question, Who is the president of the U.S.? but it would be easily ranked in the first position according to word overlap measures. TREC QA track 1 has shown that models using syntactic structures can help to overcome the limitations of pure similarity-based models. Unfortunately, syntactic representations are difficult to design when the answer is supported by multiple sentences as (i) discourse processing still requires development to reach a sufficient level of performance and (ii) language models taking into account the dependencies between words spanning several sentences (i.e., long distance dependencies) are rather difficult to model. For example, let us consider a real question-answer pair from the Answerbag community-based QA collection2 (we will use it as a running example throughout the rest of the paper):

Q1: Is movie theater popcorn vegan?

A1: Any movie theater popcorn that includes butter and therefore dairy products is not vegan. However, the popcorn kernels alone can be considered vegan if popped using canola, coconut or other plant oils which some theaters offer as an alternative to standard popcorn.

An individual analysis of the two answer sentences produces both a positive and a negative piece of evidence making the answer passage candidate unreliable (and thus it would be ranked lower3 than other candidates). In contrast, the availability of the cross-sentential structure popcorn is not vegan, however would suggest that the candidate is promising as it solves the contradiction. Unfortunately, handcrafting these patterns or rules is not cost-effective and typically requires to replicate the effort when the application domain changes.

A viable alternative is to apply supervised methods to learn answer/passage rerankers, e.g., [35, 15]. Nevertheless,

1http://trec.nist.gov/data/qa.html
2http://www.answerbag.com
3Very simple models would unawarely rank it in top position for the repeated presence of the words popcorn and vegan.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR ’12, August 12–16, 2012, Portland, Oregon, USA.  
Copyright 2012 ACM 978-1-4503-1472-5/12/08 ...$15.00.
machine learning algorithms applied to similarity scores between questions and answers would not be able to learn patterns such as the one above. In contrast, in [23], we proposed to apply structural kernels [28] to the syntactic representation of question/answer passage pairs. However, such model could only take into account very short passages composed of only one sentence and the experiments were carried out on a small dataset of about 2,000 examples. Most importantly, we did not considered the relational information linking question and answer passages.

In this paper, we define novel relational structural models for passage re-ranking in QA systems based on supervised machine learning. We train our models using structural representations of question and answer passages, which can include multiple sentences. We utilize an off-the-shelf shallow syntactic parser to group words in sentences’ constituents, which in turn are organized in a tree structures. Based on the naïve syntactic matching between the word lemmas of a given question and its candidate answer, we mark the corresponding constituents in the resulting tree to capture similar semantic relationships between the two. This way mutual structural relationships between a question and its answer is made available in the tree nodes. To effectively and efficiently use such structures in Support Vector Machines (SVMs), we apply structural kernels, e.g., sequence and tree kernels.

We experimented with our models and the fast SVM algorithm for structural kernels, which allowed us to train our rerankers on datasets up to hundreds of thousands instances. The results on two different QA datasets: Answerbag and Jeopardy! show that our models deliver significant improvements on passage re-ranking tasks, highly improving two different baselines, i.e., reducing the error in Recall by about 18-20% of BM25 and Watson primary search [11], respectively. The very interesting aspect is that the learning curves show that (i) complex relational structures can be learned using large data and (ii) further improvement is possible using more data.

In the remainder of the paper, Section 2 reports on the related work, Section 3 introduces kernel methods for structured data, Section 4 presents our models for re-ranking passages/answers, including the question/answer relational model, Section 5 illustrates our experiments and finally Section 6 derives our conclusions.

2. RELATED WORK

The study carried out in this paper primarily focuses on the use of shallow syntactic/semantic structures for training answer re-ranking with large data. Early related work on the use of syntax and semantics in Information Retrieval was carried out in [38, 39, 32] and in [34, 33]. The results showed that the use of advanced linguistic information was not effective for document retrieval. In contrast, QA work shows that semantics and syntax are essential to retrieve concise answers, e.g., [14, 40, 31]. However, successful approaches in TREC-style systems were based on several interconnected modules exploiting complex heuristics and fine tuning. The effective combination of such modules was strongly dependent on manual setting, which often remained undisclosed.

In our study, we focus on passage re-ranking that can be plugged into virtually any QA system. It can also be directly used as a back-end reranker for ranking non-factoid answers. Reranking can be seen as a typical text categorization task, i.e., the classification of pairs of text fragments constituted by a question and an answer or pairs of questions, e.g., [25, 15]. In this context, a large body of work regards definition/description questions, e.g., [4, 6, 29, 1, 23, 35].

In [6], answer ranks were computed based on the probabilities of bigram language models generating candidate answers. Language modeling was also applied to definitional QA in [9] to learn soft pattern models based on bigrams. Other related work, such as [26, 36], was also very tied to bag-of-words features. Our approach is different from the above as we are able to automatically learn structural relationships between a question and candidate answer passages. Such relationships have also been targeted in [10] by creating ad-hoc joint question-answer representations. Additionally, [29, 1, 35] report significant improvement by exploiting expensive linguistic approaches, e.g., predicate argument structures, for re-ranking candidate answer lists. Our work in [23, 21] was the first to exploit kernel methods for modeling answer re-ranking using syntactic and shallow semantic tree kernels based on predicate argument structures. However, our method lacked the use of important relational information between a question and a candidate answer, which is essential to learn accurate relational patterns. Additionally, the low computational efficiency of such approach prevented its application to large datasets. In contrast, we integrate our shallow tree re-ranking models within a much more efficient training algorithm for SVMs with structural kernels, which allows us to carry out experiments on much larger datasets (several hundred thousands examples). Along with our relational representation this appears to be the key aspect for the high improvement over basic models. Finally, models for factoid questions using linguistic structures has been carried out in [2, 3]. Again, the proposed methods rely on manual design of features whereas our approach is more general for passages and/or answer re-ranking.

In summary, the main distinctive property of our approach with respect to the previous work adopting syntactic and semantic structures is that we can define the latter without requiring neither a tedious manual linguistic analysis nor computationally expensive linguistic processors. We do not carry out feature engineering since we simply let kernel functions automatically generate very large feature sets (tree fragments or substrings) to effectively represent structural syntactic/semantic information found in the question/answer pairs. Moreover, our models can be trained on larger datasets, which allows for high improvement over the basic QA system.

Finally, structural kernels have been applied for several natural language tasks, e.g., syntactic parsing [8, 17], Semantic Role Labeling [19, 12, 13, 22], Relation Extraction [24, 42], Pronominal Coreference [37], Recognizing Textual Entailment (RTE) [18] and text categorization [5].

3. KERNELS FOR STRUCTURED DATA

Kernel methods and structural kernels, in particular, are
very effective means for automatic feature engineering for natural language texts. In kernel machines both learning and classification algorithms only depend on the ability to evaluate an inner product between instances, which corresponds to computing a similarity score. In several cases, we consider the equation to evaluate STK and PTK\(^4\).

3.2.1 Syntactic Tree Kernels (STK)

To compute STK is enough to compute \(\Delta_{STK}(n_1, n_2)\) as follows (recalling that since it is a syntactic tree kernels, each node can be associated with a production rule): (i) if the productions at \(n_1\) and \(n_2\) are different then \(\Delta_{STK}(n_1, n_2) = 0\); (ii) if the productions at \(n_1\) and \(n_2\) are the same, and \(n_1\) and \(n_2\) have only leave children then \(\Delta_{STK}(n_1, n_2) = \lambda\); and (iii) if the productions at \(n_1\) and \(n_2\) are the same, and \(n_1\) and \(n_2\) are not pre-terminals then \(\Delta_{STK}(n_1, n_2) = \lambda \prod_{i=1}^{l(n_1)} (1 + \Delta_{STK}(c_{l_1}^{i}, c_{l_2}^{i}))\), where \(l(n_1)\) is the number of children of \(n_1\) and \(c_{l_1}^{i}\) is the \(i\)-th child of the node \(n_1\). Note that, since the productions are the same, \(l(n_1) = l(n_2)\) and the computational complexity of STK is \(O(|N_{n_1}| + |N_{n_2}|)\) but the average running time tends to be linear, i.e., \(O(|N_{n_1}| + |N_{n_2}|)\), for natural language syntactic trees [20].

For example, Figure 1 shows the shallow syntactic tree of the question Q1. This is a flat tree encoding part-of-speech tags and words. Figure 2 shows some of its fragments generated by STK, highlighting an important limitation on the type of substructures it can generate: production rules, i.e., children of a node cannot be split in the substructures.

3.2.2 The Partial Tree Kernel

The computation of PTK is carried out by the following \(\Delta_{PTK}\) function: if the labels of \(n_1\) and \(n_2\) are different then \(\Delta_{PTK}(n_1, n_2) = 0\); else \(\Delta_{PTK}(n_1, n_2) = \lambda \left( \sum_{f_1, f_2} \prod_{i=1}^{l(n_1)} \Delta_{PTK}(c_{l_1}^{i}, c_{l_2}^{i}) \right)\), where \(d(\hat{I}_1) = \hat{I}_1 - \hat{I}_1\) and \(d(\hat{I}_2) = \hat{I}_2 - \hat{I}_2\). This way, PTK penalizes both larger trees and child subsequences with gaps. PTK is more general than STK, e.g., the shared subsequences containing all children of nodes implement STK. The computational complexity of PTK is \(O(p^2|N_{n_1}| |N_{n_2}|)\) [20], where \(p\) is the largest subsequence of children that we want consider and \(\rho\) is the maximal outdegree observed in the two trees. However, the average running time again tends to be linear for natural language syntactic trees [20].

PTK can generate any subset of connected nodes of a tree \(T\), whose edges are in \(T\). For example, Fig. 3 shows the fragments of the tree in Fig. 1. Note that each fragment

\(^4\)To have a similarity score between 0 and 1, a normalization in the kernel space, i.e., \(\frac{TK(T_1, T_2)}{\sqrt{TK(T_1, T_1) \cdot TK(T_2, T_2)}}\) is applied.
Passages
Q
pairs
reduced to a binary classification problem by creating
approach [30], the re-ranking problem – learning to pick
that is supposed to improve the original ranking obtained
presentation. The system outputs a reordered list of answer passages
chunks, for each sentence. These annotations are then used
parser that produces a set of annotations, e.g., POS-tags and
answer passages for a given question. Next, the question to-
iteratively to retrieve most relevant an-
the space of pairs
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
K
WNSS taggers are more computationally demanding preprocessing tasks. Below, we describe a variety of models for representing question-answer pairs that rely only on the shallow syntactic and semantic analysis described above.

### 4.2.2 Shallow representations of questions and answer passages

As the first preprocessing step, each question and its answer passage are split into sentences and tokenized. Next, we run a shallow parser, which annotates each token in the sentence with four types of tags: POS, chunks, NER and WNSS tags. As a shallow parser we used SST-light [7] tagger, which carries out tokenization, POS-tagging, NER, WNSS tagging and lemmatization, which is useful to increase the matching probability between the words in the question and the answer.

An example of annotation graph obtained for the question of our running example Q1 is provided in Fig. 6. Each word is tagged with its POS tags. Words in the phrase **movie theater popcorn vegan** are organized into a single chunk **NP**. Also the parser finds an NE, **theater**, which is annotated with the tag **CORPORATION**. Typically, NER tags are fairly scarce, while WNSS has better coverage of words in a sentence on average, but many of its semantic categories are rather general. A similar graph can be obtained for each sentence in the answer. It conveniently allows for experimenting with different representations of question-answer pairs.

When structural kernels are applied to the shallow tree in Fig. 6 (we consider a tree root connecting the top POS-tag nodes or the top chunk nodes), they can generate rich features, i.e., n-grams combining tags and words, from the question or from the answer passages. However, to allow kernels capture relations between a question and its answer passage, we need to establish relational links between them. For this purpose, we adopt the following simple strategy: we mark the parent node (POS-tags) of matching leaf nodes (i.e., lemmas) with a special placeholders REL (which carries out tokenization, POS-tagging, NER, WNSS tagging and lemmatization, which is useful to increase the matching probability between the words in the question and the answer. An example of annotation graph obtained for the question and answer of our running example Q1 is provided in Fig. 7. Each word is tagged with its POS tags. Words in the phrase **any movie theater popcorn vegan** that include butter and therefore dairy is not vegan.

### 4.2.3 Structure Pruning

Providing an efficient representation for the answer passages that contain multiple sentences is not a trivial task. We deal with this problem by simply putting trees representing individual sentences of the passage under a single common root. Since answer passages typically consist of more than two sentences each 10 to 20+ words long the resulting shallow tree may be very large. This makes structural kernels that consider all possible substructures inefficient for such large trees. To alleviate such problem, we propose a simple yet efficient pruning strategy, which reduces the size of the tree down to a manageable size, thus gaining substantial computational savings while preserving the accuracy.

A straightforward approach is to prune away all the nodes that are not marked by the REL tag, as they presumably play insignificant role in relating question and its answer passage. However, intuitively, the nodes that surround the relational nodes may carry important contextual information, which can help the learning algorithm to generalize better. Hence, we also keep the nodes that are within a specified distance from the relational nodes (i.e., the ray of pruning is an additional parameter of our models). An example of pruning with ray equal to 1 between example question Q1 and the first sentence of the answer is shown in Fig. 9. The result is a shorter version of the original sentence while preserving the most essential parts (nodes marked with the REL tag) that encode relational information between the answer and the question. Additionally, the nodes that surround the REL tag within the specified distance can serve as a source of additional contextual features. Since questions are typically one sentence long and contain fewer words than answers, we only apply pruning on answer passages.

\(^7+\) is the operator of a regular expression meaning at least one occurrence.

---

\(^6\)http://sourceforge.net/projects/supersensetag/

---

**Figure 6:** Annotation graph obtained from a shallow parser for the example question Q1.

**Figure 8:** Question and answer sequences aligned by lexical matching on word lemmas. Matched nodes are marked with a special REL marker.
4.2.4 Chunked Representation

Next, we consider the model using chunker tags (CH). Chunks introduce additional level into the shallow tree hierarchy by organizing syntactically related words into constituents. We propagate the REL tag from the relational POS-tags up to their parent chunk nodes. Here we prune the tree on the chunk level, i.e., we keep only the chunk nodes and its subtrees that are within the specified distance from the chunks marked with REL. This results in fewer nodes being pruned compared to POS-tags based pruning, but at the same time, it preserves all contextual words that appear in the same chunk as the matched words. Another very important advantage is that the tree kernel can generate more effective features to characterize the role of relational constituents. For example, Fig. 10(a) reports the shallow tree for the answer sentence reorganized into chunks. The noun phrase any movie theater popcorn generalizes the patterns exemplified above thus allowing longer dependency patterns, e.g., NP is not adj. Fig. 7 demonstrates the entire answer passage organized in a shallow tree after pruning with radius 1. Thanks to the effective pruning, which allows for preserving important properties and at the same time reducing the size of the inter-sentence structure, our kernels can extract important long distance dependency patterns such as: NP not adj however alone can be considered adj, which encode shallow discourse relations.

4.2.5 Semantic Tags

We also consider shallow structures enhanced with additional semantics from NER, and WNSS tags (Fig. 10(b)). If any of the words in a chunk have a NER tag, we propagate it up to the parent chunk tag. If there are multiple words having NER tags inside the same chunk we use the last one. Since NER tags are fairly scarce, we also consider the model where NER tags are augmented with additional WNSS tags (see Fig. 10(b)). For example, when the super sense is used (and considering the adjective grounded in vegan), the following pattern is also automatically generated food not vegan however alone can be considered vegan.

Figure 7: Representation of the entire answer passage using CH+REL model with pruning (ray=1)

Figure 9: Pruning with ray 1. Original sentence (top) and its pruned version (bottom). Nodes in the middle of the top sequence are pruned away, while the gray-shaded nodes of the bottom sentence are kept as the context.

Figure 10: (a) Pruned answer trees with constituents re-organized in chunks. REL marker is propagated up from POS to chunk nodes. (b) Enhanced representation with additional semantics from NER and WNSS.

4.3 Efficient training of the relational ranking model

As discussed in Section 4.1, the task of learning a kernel-based re-ranker using supervised classifiers such as SVMs can be reduced to a problem of binary classification. While this approach allows for the application of a wide range of binary classifiers it may require to carry out learning on very large datasets, since for each question in the training set we need to form all combinations of question answer pairs between the correct and candidate answer passages. Even when the number of candidate answers per question is relatively small, for example 10, the number of training examples generated for the binary classifier increases by an order of magnitude.

Hence, to learn a re-ranker on medium sized datasets (from thousands to tens of thousands of questions) would require to train binary classifiers on datasets of hundreds of thousands to millions of examples. While this poses little limitation for the SVMs with linear kernels for which efficient \( O(n) \) learning algorithms have been recently introduced, using SVMs with non-linear kernels, structural kernels in particular, can represent a major computational challenge.

Kernelized SVMs scale quadratically with the number of training examples which prohibits their application on large data. This is why previous research that focused on application of non-linear kernels to question-answer re-ranking tasks has reported experiments on very small datasets (several thousands of examples).

The major bottleneck for the application of SVMs with structural kernels stems from the necessity to carry out learning in the dual space. Training can be seen as an iterative process that requires to compute an inner product be-
tween the current model represented by the weight vector \( \vec{w} \) and a training example \( \delta_i \). It can be shown (see, for example, [27]) that this involves quadratic number of kernel evaluations (Eq. 2) at each iteration, which makes the overall scaling behavior of the learning algorithm \( O(n^2) \). In this paper, we adopt a Cutting Plane Algorithm with sampling originally introduced in [41]. The algorithm uses sampling to effectively reduce the number of kernel evaluations while preserving theoretical bounds on accuracy and convergence. We follow the method in [27] developed for SVM learning with structural kernels, which has been shown to provide substantial speedups in training over conventional training methods for non-linear kernels. The Cutting Plane Algorithm also allows for easy parallelization, which makes it possible to experiment with larger data.

5. EXPERIMENTS

To test the efficacy of our re-ranking approach based on relational features, we conduct experiments on two diverse QA collections: Answerbag and Jeopardy!. For the Answerbag dataset, we built the entire answer passage re-ranking system as outlined in Fig. 4. Unlike the Answerbag collection, the Jeopardy! dataset already comes with ranked answer passages for each question.

5.1 Setup

To train our re-rankers we used uSVM-TK software, which implements highly efficient Cutting Plane Algorithm with sampling [27] within the framework of SVM-Light-TK [20, 16]. This enables the use of structural kernels, e.g., SK, STK and PTK (see Sec. 3.2). We extended the software to handle re-ranking on pairs of multiple trees corresponding to question-answer pairs. We used default parameters to favor replicability of our results.

We ran all the experiments on a machine equipped with 12 CPUs Intel® Xeon® 2.93GHz CPUs carrying 98Gb of RAM under Linux. The uSVM-TK software allows for training and testing in parallel mode; the reported runtimes refer to the use of four CPUs.

To measure the accuracy of our re-ranker system, we used two metrics: Recall of 1 at rank=1 (R1@1) i.e., the percentage of questions with at least a correct answer ranked at the top position of the candidate list, and Mean Reciprocal Rank (MRR) computed as follows: \( MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \), where \( \text{rank}_i \) is the position of the correct answer in the candidate list.

5.2 Corpora

Answerbag. The first corpus we experimented with represents a large sample of Answerbag QA collection\(^5\). This community-driven Web QA collection is very active and is similar to other popular QA sites such as Yahoo! Answers\(^6\) and WikiAnswers\(^7\). One of the reasons that makes this site attractive for its users is that it contains a very large portion of questions tagged as “professionally researched”. For this type of questions, website moderators provide succinct and exhaustive answers, which makes it ideal for building a high quality question answering corpus. We scraped 180,000 question-answer pairs from 20 different categories.

As an answer retrieval component for this collection, we use an off-the-shelf search engine: Terrier\(^1\) using BM25 scoring function with other settings set to default. We retain top 15 candidates for each question. We omit questions for which the search engine failed to retrieve a correct answer in the top 15 positions, since these cases carry no value for training or evaluation of our re-ranker models. This leaves us with roughly 60% of the questions from the original data for which the correct answer is found in the top 15 positions. From this set of questions we form training sets of increasing size starting from 25k and up to 750k pairs. For testing, we use 30k re-ranking examples which corresponds to roughly 2k questions.

Jeopardy! The second dataset represents a small subset of the output from the Primary Search retrieval component of IBM Watson QA system\(^12\) built specifically to provide answers for questions from the Jeopardy! challenge\(^13\). Different from the setting for the Answerbag collection, where we had to use our own passage retrieval system, here each question is provided together with the set of answer passages ranked by Watson. This allows us to compare our re-ranker to a very strong baseline (IBM Watson is a state-of-the-art QA system). The collection is composed of 517 questions where each question comes with 50 candidate passages. Unlike the Answerbag dataset, most of the questions in this collection have more than one relevant answer passage. On average there are 6 relevant out 50 candidate passages for each question. We use 70% (259 questions) for training and 30% (116 questions) for testing. Since each question has multiple correct answer passages, to create a training set, we paired up each correct answer with all incorrect candidates. As a result we obtain 63,361 examples for training and 5,706 testing.

5.3 Accuracy on Answerbag

Table 1 summarizes the results for various models described in the Section 4.2 trained on the set of 25,000 examples (1,676 questions). W stands for bag-of-words, POS - bag-of-POS-tags, REL - bag-of-REL nodes, CH indicates models with sentence constituents organized in chunks, PR-1 indicates pruning with ray 1. NER corresponds to tags obtained from named-entity recognizer and WNSS are tags from homonymous tagger.

Bags of features. As expected models that are based on just bags of features (W, POS, W+POS) perform poorly showing negative or no improvement over the baseline. Interestingly, we notice that introducing a relational REL tag to mark the matched nodes in the question and its answer (models: POS+REL, W+POS+REL) provides a small increase in accuracy compared to the models that do not exploit such relational information. Even for completely unstructured features, relational nodes can serve as a useful cue to discriminate between the correct and incorrect question-answer pairs. Surprisingly, the model combining all of the features including additional features from NER and WSSN taggers (W + POS + NER + WSSN + REL) results in severe over-fitting. After performing manual inspection, we concluded that, since the search engine (SE) already sets a very strong baseline for this QA collection (MRR: 71.63% and REC1@1: 59.14%), using features similar to those of

\(^{11}\)http://terrier.org/


\(^{9}\)http://www.answers.yahoo.com

\(^{10}\)http://wiki.answers.com
the SE for re-ranking cannot provide improvement. Additionally, NER and WSNN taggers that assign words to very general categories are not helpful for discriminating between the similar answer passages retrieved by the SE.

In summary, the top part of the Table 1 provides a clear indication that using unstructured features gives little to no benefit for the re-ranker. To confirm this behavior does not change on larger data, we also carried out training on larger samples of the training set but observed essentially flat learning curves, which means that the models relying only on the bags of features do not improve with more data.

Linear sequence models. In contrast, using SK applied to POS tags generates all possible skip n-grams. Coupled with the relational information captured by the REL marker (POS+REL) it provides interesting improvement over the baseline. At the same time, when applied on the sequence of POS-tags without relational information (POS), we observe no improvement. We also show the results using different values for the maximum subsequence length in SK: (SK 1), (SK 2), (SK 3), (SK 4), and (SK 5). For example, (SK 3) will consider all skip n-grams up to the length 3. The results show that complex features up to 5 elements provide the highest accuracy. This suggests that the modeled structures provide an adequate generalization level for SK to extract effective 5-grams (constituted by POS-tags and words).

Structural features. Next, we consider the application of structural kernels, e.g. STK and PTK, to shallow trees (POS+REL). We first note, that STK does not work well on shallow representations, due to its limited expressive power as discussed in Section 3.2.1. Conversely, PTK that generates all possible tree fragments is able to pick up the relevant features showing significant improvement over the baseline. To make STK more efficient and effective, we consider chunks of text nodes are preserved, demonstrate slightly higher accuracy than other models with softer levels of pruning. This is essential to speed up kernel evaluations, which would allow for better scaling to larger data. The study of runtime savings for different levels of pruning is presented in Table 2.

5.4 Efficiency analysis - large scale learning

As discussed in Sec. 4.2 pruning is an effective way to reduce the size of a tree by removing less pertinent nodes. This is essential to speed up kernel evaluations, which would allow for better scaling to larger data. The study of runtime savings for different levels of pruning is presented in Table 2.

We first observe that applying pruning with rays $\in \{0, 1\}$ on POS+REL trees for both SK and PTK results in runtime speedups up to 2x. A similar picture is observed for the CH+REL tree representations. Interestingly, models with the most aggressive pruning using ray=0, such that no context nodes are preserved, demonstrate slightly higher accuracy than other models with softer levels of pruning. This can be explained by the fact that keeping additional context nodes around the relational nodes produces richer features.

The top part of the Table 1 provides a clear indication that using unstructured features gives little to no benefit for the re-ranker.
5.5 Jeopardy! experiments

Since the size of Jeopardy! dataset does not allow for building a meaningful learning curve we report the plot of R1@r, which measures the percentage of questions with at least one correct answer in the first x positions. We experimented with PTK applied to CH+REL structures also encoding NER and WNSS. Figure 13 shows that for any rank position, the simplest model outperforms semantic models. Most importantly, the Primary Search of Watson is improved up to 5 points for an error reduction of 20%.

ACKNOWLEDGEMENTS

We thank the IBM Watson team and in particular Jennifer Chu-Carroll for providing us with the Jeopardy! dataset. Many thanks to the SIGIR reviewers. This work has been supported by the IBM’s Open Collaboration Research (OCR) awards program. Partial support has been also received by the EC framework FP7/2007-2013 under the grants #247758: ETERNALS – Trustworthy Eternal Systems via Evolving Software, Data and Knowledge, and #288024: LiMoSInE – Linguistically Motivated Semantic aggregation engiNes.
7. REFERENCES


