Neural Attention for Learning to Rank Questions in Community Question Answering

Salvatore Romeo, Giovanni Da San Martino,
Alberto Barrón-Cedeño and **Alessandro Moschitti**
Qatar Computing Research Institute, HBKU, Doha, Qatar
{sromeo, gmartino, albarron, amoschitti}@qf.org.qa

Yonatan Belinkov, Wei-Ning Hsu,
Yu Zhang, **Mitra Mohtarami** and **James Glass**
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
{belinkov, wnhsu, yzhang87, mitram, glass}@mit.edu

Abstract

In real-world data, e.g., from Web forums, text is often contaminated with redundant or irrelevant content, which leads to introducing noise in machine learning algorithms. In this paper, we apply Long Short-Term Memory networks with an attention mechanism, which can select important parts of text for the task of similar question retrieval from community Question Answering (cQA) forums. In particular, we use the attention weights for both selecting entire sentences and their subparts, i.e., word/chunk, from shallow syntactic trees. More interestingly, we apply tree kernels to the filtered text representations, thus exploiting the implicit features of the subtree space for learning question reranking. Our results show that the attention-based pruning allows for achieving the top position in the cQA challenge of SemEval 2016, with a relatively large gap from the other participants while greatly decreasing running time.

1 Introduction

Previous work on modeling high-level semantic tasks, e.g., paraphrasing, recognizing textual entailment, and question answering (QA), has shown that syntactic and semantic structures are essential for boosting the accuracy of the applied machine learning algorithm. However, when dealing with real-world data, automatic parsers typically decrease their accuracy. Additionally, there is often a considerable amount of meaningless text for the task, contributing to generating noisy structures. This kind of phenomena can be clearly observed in new Web applications, such as community Question Answering (cQA), where the presence of informal, redundant, and often unrelated text constitutes a major challenge for learning the automatic detection of related topics. For example, given the original question:

Which all places are there for tourists to Qatar? My nephew 18 years on visit.

a cQA system has to infer whether the following question is related:

What are the tourist places in Qatar? I'm likely to travel in the month of june. Just wanna know some good places to visit.

This implies that the system has to (i) recognize the sentences, *My nephew 18 years on visit* and *I'm likely to travel in the month of june*, as irrelevant for the classification task; and (ii) deal with non-standard writing (e.g., *wanna*). If the above steps are carried out with *good* accuracy, complex semantic models can be applied to the selected text to achieve better performance. In particular, such models have to encode structural relations between the constituents of the two questions, e.g., *places are there for tourists* and *some good places to visit*, in the learning algorithm. This is a challenging task as we do not know which relations might be useful, whereas explicitly including all of them is an intractable problem.

One solution for handling this problem for the answer selection task of standard QA was proposed in (Severyn and Moschitti, 2012). They applied tree kernels (TK) to relational syntactic structures, i.e., a graph representing both question and answer structures. TKs allow for using all the substructures of the relational structures as features in the learning algorithm. However, in Web forums the TK performance is downgraded by the presence of noise and insignificant information, which also makes TKs too slow for processing large datasets. This suggests that text selection (TS) models, applied before TKs in both learning and classification phases have great potential to positively impact the final performance.

In this paper, we study several methods for TS: (i) unsupervised methods based on scalar products with and without TF$\times$IDF weights and (ii) supervised approaches based on attention weights learned by a Long Short-Term Memory (LSTM) network. Additionally, we apply the techniques above with two different strategies reflecting two different granularity levels for: (i) selecting the set of sentences constituting the question trees, and (ii) filtering out tree constituents, where the latter represent pairs of input questions to TKs. We measure the benefit of our TS models against a strong baseline based on TKs and domain specific features for the task of question reranking for cQA, which was recently proposed in SemEval 2016 (Nakov et al., 2016).

Our extensive experiments on the official SemEval dataset produced the following results. First, our basic model, which is based on a combination of (*i*) features using traditional text similarity measures, e.g., bag-of-word models, (*ii*) the initial rank provided by the search engine (providing the initial question rank), and (*iii*) TKs applied to the syntactic structure of the sentences of original and retrieved questions, is state of the art as it ranked 2nd at the official SemEval 2016 challenge (Nakov et al., 2016). Second, TS significantly impacts the performance of TKs by feeding them with better representations. In particular, when using supervised methods, i.e., attention model weights, in sentence selection and tree pruning strategies both provide better representations, and higher results than the models using all sentences or those selected by TF$\times$IDF approaches. Finally, our TS-based system outperforms the top system of the SemEval cQA challenge.

The rest of the paper is organized as follows. Section 2 overviews related work. Section 3 describes our learning-to-rank approach. Section 4 describes the application of LSTMs in TK-based ranking models. Section 5 describes our text selection strategies. Section 6 discusses our experiments and the obtained results. Finally, Section 7 concludes the paper.

2 Related Work

Question ranking in cQA has been central in the research community practically since the begining of cQA system design. Beside "standard" similarity measures, different characterizations and models have been explored. For instance, Cao et al. (2008) proposed a question recommendation system based on the questions' topic and Duan et al. (2008) added the question's focus into the formula. A different approach using topic modeling for question retrieval was introduced by Ji et al. (2012) and Zhang et al. (2014). Here, the authors use LDA topic modeling to learn the latent semantic topics that generate question/answer pairs and use the learned topic distribution to retrieve similar historical questions.

Various methods rely on machine-translation models. For instance, Jeon et al. (2005) and Zhou et al. (2011) used monolingual phrase-based translation models to compare the questions. Jeon et al. (2005) built their translator from a collection of previously identified similar questions whereas Zhou et al. (2011) used question–answer pairs.

Other approaches are based on syntactic representations. This is the case of Wang et al. (2009), who consider the number of common substructures of parse trees to estimate the similarity between two questions. Both Barrón-Cedeño et al. (2016) and Filice et al. (2016) use parse trees as well. The difference is that they use them directly within a tree kernel, with the use of the KeLP platform (Filice et al., 2015a). The latter two models were applied on the SemEval 2016 Task 3 challenge on cQA (Nakov et al., 2016), which proposed a task on question ranking (together with one on answer ranking). The best-performing system in this task was the one from Franco-Salvador et al. (2016), which used SVM^{rank} (Joachims, 2006) on a manifold of features, including distributed representations and semantic resources. To our knowledge, the only work exploring text selection for improving cQA or QA systems is (Barrón-Cedeño

Original Question

q$_o$: What are the tourist places in Qatar? I'm likely to travel in the month of June. Just wanna know some good places to visit.

G	*GS*	*R*	**Retrieved Questions**
1	-1	8	The Qatar banana island will be transfered by the end of 2013 to 5 stars resort called Anantara. Has anyone seen this island? Where is it? Is it near to Corniche?
2	+1	2	Is there a good place here where I can spend some quality time with my friends?
3	-1	7	Where is the best beach in Qatar? Maybe a silent and romantic bay? Where to go for it?
4	-1	9	Any suggestions on what are the happenings in Qatar on Holidays? Something new and exciting suggestions please?
5	-1	3	Where in Qatar is the best place for Snorkeling? I'm planning to go out next friday but don't know where to go.
6	-1	6	Can you give me some nice places to go or fun things to do in Doha for children 17-18 years old? Where can we do some watersports (just for once, not as a member), or some quad driving? Let me know please. Thanks.
7	+1	1	Which all places are there for tourists to Qatar? My nephew 18 years on visit.
8	-1	10	Could you suggest the best holiday destination in the world?
9	-1	5	I really would like to know where the best place to catch fish here in Qatar is. But of course from the beach. I go every week to Umsaeed but rerly i catch somthing! So experianced people your reply will be appreciated.

Table 1: A re-ranking example: we report the Google rank (G), the gold standard relevance (GS) and our rank (R) for each question.

et al., 2016), which exploits tree kernel function itself to auto-filter the non relevant subtrees. The main difference with the approach we present in the current paper is the use of neural networks for learning attention weights and thus modeling sentence or word pruning.

Neural Approaches Recent work has shown the effectiveness of neural models for answer selection (Severyn and Moschitti, 2015; Tan et al., 2015; Feng et al., 2015) and question similarity (dos Santos et al., 2015) in community question answering. For instance, dos Santos et al. (2015) used CNN and bag-of-words (BOW) representations of original and related questions in order to compute cosine similarity scores. Recently, Bahdanau et al. (2014) presented a neural attention model for machine translation and showed that the attention mechanism is helpful for addressing long sentences. We use an LSTM model (Hochreiter and Schmidhuber, 1997) with an attention mechanism for capturing long dependencies in questions for the question similarity task. The major difference with previous work is that we exploit the weights learned by the attention model for selecting important text segments (words, chunks and sentences) for improving syntactic tree-kernel models.

3 Learning to Rank Questions in cQA

This section describes the question reranking problem that we study in this paper and provides state-of-the-art models for its solution. As shown by some methods presented in SemEval, TKs are important for achieving top results.

Class	train	dev	test	overall
Relevant	1,083	214	233	1,530
Irrelevant	1,586	286	467	2,339
Total	2,669	500	700	3,869

Table 2: Class distribution in the training, development, and test partitions.

3.1 Problem Description

The task we focus on is defined as follows: given an original question, q_o, and l candidate questions, q_s, retrieved with a search engine, rerank them with respect to their relevance to q_o.

We use the SemEval 2016 cQA dataset (Nakov et al., 2016), which is composed of 386 user questions, each of which includes 10 potentially related questions. At construction time, the Google search engine, which represents also the strong baseline for the task[1], was used to select potentially relevant forum questions. Table 1 shows an example of the data: an original question on top, followed by several questions retrieved by Google. Nakov et al. crowdsourced the manual annotation of the relevance of the questions. Table 2 shows the amount of relevant and irrelevant instances in the different partitions of the corpus, whereas Table 3 illustrates the distribution of relevant/irrelevant forum questions per ranking position.

[1]The task assumes that the Google Rank is not optimal.

R	train	dev	test	overall	R	train	dev	test	overal
1	0.21 ± 0.05	0.24 ± 0.07	0.40 ± 0.11	0.25 ± 0.07	6	0.08 ± 0.02	0.09 ± 0.02	0.05 ± 0.01	0.08 ± 0.02
2	0.14 ± 0.03	0.18 ± 0.02	0.12 ± 0.02	0.14 ± 0.03	7	0.08 ± 0.02	0.07 ± 0.01	0.05 ± 0.01	0.07 ± 0.02
3	0.11 ± 0.02	0.10 ± 0.01	0.08 ± 0.01	0.10 ± 0.02	8	0.06 ± 0.01	0.04 ± 0.01	0.03 ± 0.00	0.05 ± 0.01
4	0.12 ± 0.03	0.08 ± 0.01	0.10 ± 0.03	0.11 ± 0.03	9	0.07 ± 0.02	0.06 ± 0.01	0.04 ± 0.01	0.07 ± 0.02
5	0.09 ± 0.02	0.09 ± 0.01	0.08 ± 0.02	0.09 ± 0.02	10	0.05 ± 0.01	0.05 ± 0.01	0.04 ± 0.01	0.05 ± 0.01

Table 3: Average fraction (and standard deviation) of Relevant questions at different ranking positions.

Although relevant questions tend to concentrate towards the top of the Google-generated ranking, they are fairly spread on the entire ranking scale.

3.2 The Reranking Approach

Reranking can be modeled by a scoring function $r : \mathcal{Q} \times \mathcal{Q} \rightarrow \mathbb{R}$, where $\mathcal{Q}$ is the set of questions. In turn, r can be modeled as a linear function $r(q_o, q_s) = \vec{w} \cdot \phi(q_o, q_s)$, where $\vec{w}$ is the model and $\phi()$ provides a feature vector representation of the pair.

Binary classifiers, such as SVMs (Joachims, 1999), can be applied for implementing r, where the ranked list is derived from the prediction scores.[2] We model $\phi(q_o, q_s)$ with different advanced feature sets: (*i*) TKs applied to the syntactic structures of question pairs, (*ii*) similarity features computed between q_o and q_s, and (*iii*) a rank feature, i.e., the rank assigned to the question by the search engine.

3.3 Tree Kernels for Question-to-Question Similarity

Kernelized SVMs can express $\vec{w}$ as $\sum_{i=1}^{n} \alpha_i y_i \phi(q_o^i, q_s^i)$, where n are the number of training examples, α_i are weights, y_i are the example labels, $\phi(q_o^i, q_s^i)$ is the representation of pairs of the original and candidate questions. This leads to the following scoring function:

$$r(q_o, q_s) = \sum_{i=1}^{n} \alpha_i y_i \phi(q_o, q_s) \cdot \phi(q_o^i, q_s^i) = \sum_{i=1}^{n} \alpha_i y_i K\big(\langle q_o, q_s\rangle, \langle q_o^i, q_s^i\rangle\big),$$

where the kernel, $K(\cdot,\cdot)$, intends to capture the similarity between pairs of objects constituted by the original and retrieved questions.

The definition of effective Ks for QA and other relational learning tasks, e.g., textual entailment and paraphrasing, has been studied in a large body of work, e.g., (Zanzotto and Moschitti, 2006; Filice et al., 2015b). Given the high similarity between question ranking in cQA and passage ranking in QA, we opted for the state-of-the-art model proposed by Severyn and Moschitti (2012). It should be noted that we apply TK models to pairs of questions rather than questions with their passages.

Figure 1 displays an example of the structure we used for representing the original question, q_o and the seventh candidate question, q_s, in Table 1. The graph is composed by two macro-trees, one for each question, which in turn are constituted by the syntactic trees of the sentences composing the two questions[3]. Additionally, we link the two macro-trees by connecting phrases, e.g., NP, VP, PP, when there is at least lexical match between the phrases of q_o and q_s. Such links are marked with the presence of a REL tag. Finally, we apply the following kernel:

$$K\big(\langle q_o, q_s\rangle, \langle q_o^i, q_s^i\rangle\big) = TK(t(q_o, q_s), t(q_o^i, q_s^i)) + TK(t(q_s, q_o), t(q_s^i, q_o^i)) \ ,$$

where TK is a tree kernel function, e.g., the partial tree kernel by Moschitti (2006) and $t(x, y)$ returns the syntactic tree from the text x, enriching it with the REL tags computed with respect to the syntactic tree of y.

[2]In a set of preliminary experiments, we compared, a true re-ranker, SVM^{rank} (Joachims, 2002), with a standard SVM, the results were comparable.

[3]We have used the OpenNLP tool to build the trees: `https://opennlp.apache.org`.

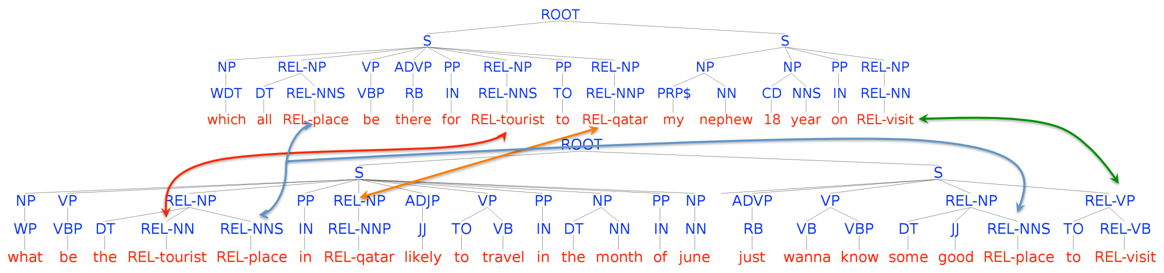

ROOT
S S
NP REL-NP VP ADVP PP REL-NP PP REL-NP NP NP PP REL-NP
WDT DT REL-NNS VBP RB IN REL-NNS TO REL-NNP PRP$ NN CD NNS IN REL-NN
which all REL-place be there for REL-tourist to REL-qatar my nephew 18 year on REL-visit
ROOT
S S
NP VP REL-NP PP REL-NP ADJP VP PP NP PP NP ADVP VP REL-NP REL-VP
WP VBP DT REL-NN REL-NNS IN REL-NNP JJ TO VB IN DT NN IN NN RB VB VBP DT JJ REL-NNS TO REL-VB
what be the REL-tourist REL-place in REL-qatar likely to travel in the month of june just wanna know some good REL-place to REL-visit

Figure 1: Representation of two questions as syntactic trees. Related nodes are enriched with REL links.

3.4 Feature Vectors

We combine the kernel above with an RBF kernel applied to feature vectors composed of similarity features. These are computed between the original and the related question and the Google rank. Such **text similarity features** (***sim***) are 20 similarities $sim(q_o, q_s)$ using word n-grams ($n = [1, \dots, 4]$), after stopword removal, using greedy string tiling (Wise, 1996), longest common subsequences (Allison and Dix, 1986), Jaccard coefficient (Jaccard, 1901), word containment (Lyon et al., 2001), and cosine similarity. We also add a structural similarity obtained by comparing the syntactic trees of the questions of an example pair using the partial tree kernel, i.e., $TK(t(q_o, q_s), t(q_s, q_o))$. Note that the operands of the kernel function are members of the same pair. The **ranking-based feature** (***rank***) is computed using the ranking generated by the baseline Google search engine system. Each candidate question is located in one position in the range $[1, \dots, 10]$. We exploit this information as the inverse of the position.

4 Long Short-Term Memory Networks for TK-based Reranking

As shown in Section 2, several neural approaches have been successfully applied to QA tasks. Unfortunately, question retrieval in cQA is heavily affected by a large amount of noise and a rather different domain, which make it difficult to effectively use out-of-domain embeddings to pre-train neural networks. This probably prevented the participants to SemEval tasks from achieving satisfactory results with such models (Nakov et al., 2016). In this work, we also tried to exploit neural models using their top-level representations for the (q_o, q_s) pair and fed them into the TK classifier as proposed by Tymoshenko et al. (2016), but this simple combination proved to be ineffective as well. In contrast, neural embeddings and weights can be useful for selecting better representations for TK models. In the reminder of this section, we present LSTM networks for question retrieval and our approach for incorporating them into TK-based rerankers.

We approach question ranking as a classification task: given a pair (q_o, q_s), we need to classify q_s as relevant or irrelevant. In order to evaluate the neural classifiers on our ranking task, we can rank candidates, q_s, according to their posterior probability. Among the different models, we were interested in having feedback on the most important pieces of text, thus we opted for LSTM (Hochreiter and Schmidhuber, 1997), which can easily incorporate attention models. LSTMs have proven to be useful in a number of language understanding tasks. Recently, Rocktäschel et al. (2016) adapted an attentional LSTM model (Bahdanau et al., 2014) to textual entailment, and a similar model has been applied to cQA (Hsu et al., 2016). We follow the same setup of the latter: given a pair (q_o, q_s), we learn two serial LSTM models. First, LSTM$_o$ reads the words' vectors of q_o, one by one, and records the corresponding memory cells and hidden states. Second, the final memory cell is used to initialize LSTM$_s$, which reads the words' vectors of q_s. The final hidden state of LSTM$_s$, $\vec{h}_{s,N}$, is used as a feature vector to feed a multi-layer perceptron with one hidden layer, followed by a softmax classifier. The objective function is the cross-entropy objective over binary relevant/irrelevant target labels. We refer to (Hsu et al., 2016) for more details on the architecture and only define the attention model here, as it will be used for TS.

Given the hidden states produced by LSTM$_o$, we compute a weighted representation of q_o:

$$\vec{h}_o = \sum_{i=1}^{L} \beta_i \vec{h}_{o,i} \quad ,$$

where $\vec{h}_{o,i}$ are the hidden states corresponding to the words of the original question, and the attention weights are computed as:

$$\beta_i = \frac{exp(a(\vec{h}_{o,i}, \vec{h}_{s,N}))}{\sum_{j=1}^{L} exp(a(\vec{h}_{o,j}, \vec{h}_{s,N}))} \quad .$$

Here $a()$ is parameterized as multi-layered perceptron (MLP) with one hidden layer and a *tanh* non-linearity (Rocktäschel et al., 2016). The input to the MLP is then a concatenation of $\vec{h}_o$ and $\vec{h}_{s,N}$. We also concatenate a one-hot vector encoding of the search engine rank (this worked better than scaling the rank into a real-valued feature as $\frac{1}{\text{rank}}$).

Intuitively, β_i assigns a higher weight to words in q_o, if they are useful for determining the relation to q_s. As we will see, these attention weights turn out to be useful for selecting important parts of the questions for the TK models. Note also that the attention here is one-sided, only on q_o. In practice, we train another model, with attention on q_s, and use its weights.

5 Text Selection

Our goal is to select important sentences and/or their constituents to improve the representation of the TK-based system proposed in Section 3. We consider two strategies: selecting a subset of sentences among those composing each question or pruning tree nodes in a bottom-up fashion. For each strategy, we consider both supervised methods by LSTM attention weights and unsupervised methods for TS.

5.1 Sentence selection

Given an original question q_o, with its sentences $\{s_1^o, ..., s_n^o\}$, and a related question q_s, with its sentences $\{s_1^s, ...s_m^s\}$, we aim at selecting those sentences that are most relevant for determining the similarity between q_o and q_s. For this purpose, we create two ranked sentence lists, Σ_o and Σ_s, such that the top-k (from each set) will constitute the sets to be parsed and used to build the macro-trees described in Section 3.3. We experiment with different values for the k parameter. Our sorting algorithm is rather simple: we (i) find the most similar pair of sentences, $(\sigma_o, \sigma_s) \in q_o \times q_s$, using a scoring function, $sim(\sigma_o, \sigma_s)$; (ii) store them individually in Σ_o and Σ_s, respectively, by also preserving the insertion order; and (iii) remove such sentences from q_o and q_s. We continue executing these steps until $q_o \times q_s$ is empty. The scoring (similarity) function can be modeled with both supervised and unsupervised methods.

Unsupervised methods We generate the vector representation of each sentence in q_o and q_s by averaging 300-dimensional word2vec (Mikolov et al., 2013a; Mikolov et al., 2013b) vector representations after stopword removal (we use the Google News default model). Then, we compute sim as the standard cosine similarity. We call this model **sim(nw)**. As a second approach, we apply the same algorithm above by weighing the word2vec vectors with TF×IDF (**sim(tf-idf)**), where IDF is derived from the entire dataset. Note that averaging has the consequence of ignoring the word order in the sentence. We leave the exploration of more sophisticated sentence representation models (e.g., Skip-Thought (Kiros et al., 2015)) for future work.

Supervised Methods We conjectured that using the information encoded by question labels to guide TS may improve our ability to measure question similarity. For this purpose, we exploit the attention weights learned by our LSTM model (Section 4). In more detail, the sentence scores for σ_o and σ_s are computed with the average attention weights β_i over sentence words. Since the attention model is one-sided, it generates weights for either the original or the related question. We thus train two separate models, for q_o and q_s, and rank each sentence list independently, on the basis of the attention weights[4].

[4]In practice, we found it useful to consider only the top-m weights in the average weights computation. In our experiments, we set $m = 3$, which worked well in preliminary experiments.

Model	**MAP**	AvgRec	MRR
Google baseline	71.35	86.11	76.67
Kernel-based			
sim	64.80	82.52	73.73
sim + rank	69.82	85.91	77.17
sim + rank + TK	73.58	89.10	79.83
LSTM	68.06	84.22	74.33

(a) Performance on the development set.

Model	**MAP**	AvgRec	MRR
Google baseline	74.75	88.30	83.79
Kernel-based			
sim	70.70	85.78	80.58
sim + rank	74.58	89.09	83.57
sim + rank + TK	76.15	90.79	84.76
LSTM	67.96	85.03	76.63

(b) Performance on the test set

Table 4: Different feature combination methods and learning models on the development and test sets.

We call this model ***attention***. Additionally, similarly to the unsupervised approaches, we compute a cosine similarity score for sentence pairs, where each sentence is represented as a bag-of-lemmas vector whose entries are the corresponding attention weights. Note that the attention model may assign different weights to identical words that appear more than once in the sentence. In this case, we use the average attention weight over identical lemmas. We name this model ***sim(att)***.

5.2 Tree pruning

Our second approach to TS works on the syntactic tree nodes and it is illustrated by Algorithm 1. Its main idea is to filter out the leaf nodes of the parse tree corresponding to words associated with weights lower than a user-defined threshold, where the word weights are provided by either β_i or TF×IDF. The most important step of Algorithm 1 is the recursive function *pruneNode*, which is initially invoked for the root node of the tree. Function *pruneNode* checks whether the node n is a leaf (Line 4) and then applies the appropriate strategy: (i) for non-leaf nodes, *pruneNode* is invoked for the children of n, then n is removed if all of its children are removed; and (ii) a leaf node is removed if its weight is lower than the user-defined threshold, h. Additionally, since the REL tagging has proved to be effective for paraphrasing and textual entailment tasks (Filice et al., 2015b), we experiment with the simple rule no REL-tagged node (see Section 3.3) is removed, independently of its weight and number of children.

```
1  Function PruneTree ($T$, $h$);
   Input  : a tree T;
            a pruning threshold $h$;
   Output: a pruned version of $T$
2  pruneNode(root($T$), $h$);
3  Function pruneNode ($n$, $h$);
4  if $|children(n)| > 0$ then
5      for $c \in children(n)$ do
6          pruneNode($c$, $h$);
7      end
8      if $|children(n)| = 0$ && !REL_Node(n)) then
9          remove ($n$, $T$);
10     end
11 else
12     if $n.weight < h$ && !REL_Node(n)) then
13         remove ($n$, $T$);
14     end
15 end
```

Algorithm 1: Function *PruneTree* for pruning a tree according to tf-idf or attention weights.

Finally, it should be noted that different thresholds determine different percentages of pruned nodes.

6 Experiments

In these experiments, we first evaluate our state-of-the-art reranker to establish a strong baseline. Then, we measure the impact of our different TS models with respect to accuracy and speed.

6.1 Testing the Baseline Models

We use binary SVMs to generate our rankings, where TKs are enabled by the KeLP toolkit[5]. KeLP allows for combining our three types of features within different kernels, namely RBF for the similarity features, TKs for the parse trees, and RBF kernels for the ranking-based feature[6]. We set the C parameter of SVMs to 1 in all experiments whereas we used the default values for TK and RBF kernel parameters. MAP is computed over all questions and then averaged.

[5]`https://github.com/SAG-KeLP`

[6]RBF proved superior than linear kernels for both similarity and ranking features in our internal experiments.

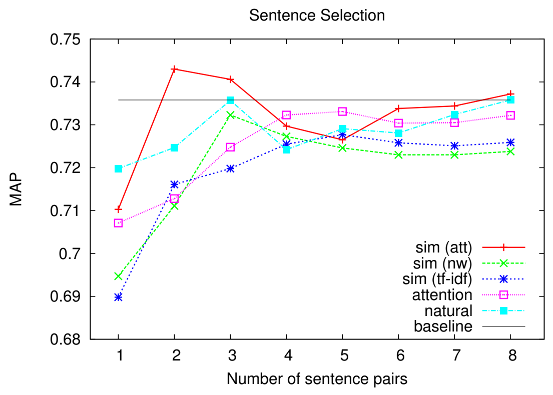


Figure 2: MAP evaluated on the dev. set according to the number of selected sentences. The different text selection methods are the natural order, the average attention weight, and cosine similarity computed with three weighting models: attention (att), no weight (nw), and tf-idf. The $baseline$ uses all the sentences.

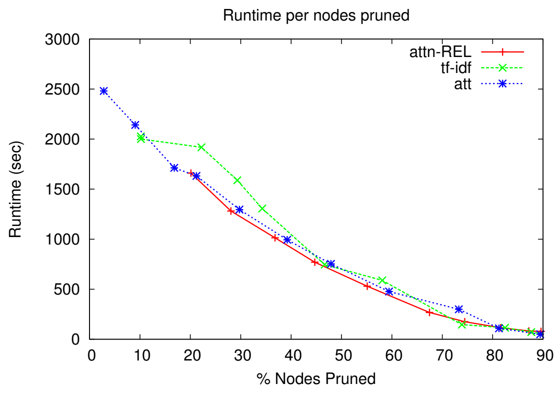


Figure 3: Reranker training time with respect to the percentage of pruned nodes using: attention weights (attn), attention weights without pruning REL nodes (attn-REL), and TF×IDF weights without pruning REL nodes (tf-idf).

Tree Kernels and Feature Combinations We first experimented with the features defined in Section 3.4: similarities (*sim*) and the *rank* feature. Additionally, we combine these features with the TKs described in Section 3.3. Tables 4a and 4b report the obtained performance on the development and test datasets. We note that (i) the Google rank baseline is rather strong; (ii) the MAP of *sim* alone is below the baseline; (iii) combining *rank* and *sim* produces an increase but the result is still below the baseline; and (iv) only the full combination, *sim+rank+TK*, improves on Google. Table 5b reports the three top systems of the SemEval cQA competition, where ConvKN (Da San Martino et al., 2016; Barrón-Cedeño et al., 2016) is the model we described in Sec. 3 whereas KeLP (Filice et al., 2016) shares a number of features with ConvKN along with the TK-based approach.

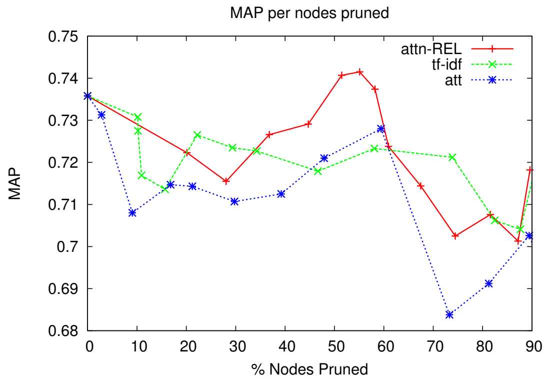
Neural Networks Tables 4a and 4b also show the results obtained with the LSTM, which does not improve upon the strong Google baseline, even though its rank is incorporated as an additional feature. This is in line with (Hsu et al., 2016), where a combination method was necessary to obtain a better performance. The low MAP can be attributed to the small dataset size (only 2,669 training examples). Nevertheless, the attention weights learned by the neural model are still useful for TS (see next section).

6.2 Measuring the Impact of the Text Selection Methods

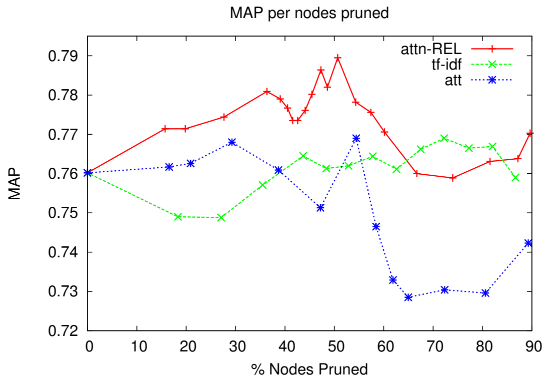
In these experiments, we focus on reducing the text used for generating the trees (the other features are computed on the standard text). We use sentence selection and tree pruning, as described in Section 5.

Sentence selection results We test different algorithms for sorting sentences for both original and related questions, i.e., to obtain the sorted lists, Σ_o and Σ_s (see Sec. 5.1), and then use the top k sentences to build the TK representations. Figure 2 shows the results on the dev. set of our reranker using different sorting algorithms: *natural* is the natural order (it can be considered as a competitive baseline), *attention* uses the average attention weight to sort sentences, and *sim(att)*, *sim(nw)* and *sim(tf-idf)* apply cosine similarity, where the sentence vector weights are constituted by attention weights, no weight (nw), and TF×IDF weights, respectively. The system *baseline* uses all the sentences, i.e., it is the best model tested in the previous section.

We note that (i) all the models using only one sentence perform lower than the *baseline*, i.e., using all sentences. (ii) The first sentence is intuitively very important as *natural* with one sentence is the best model. (iii) As soon as the number of sentences increases, the supervised models, i.e., *sim(att)* and *attention*, perform better than the unsupervised approaches, i.e., *sim(nw)* and *sim(tf-idf)*. In particular,



(a) On the development set.



(b) On the test set.

Figure 4: MAP scores after pruning the input texts using: attention weights (attn), attention weights without pruning REL-tagged nodes (attn-REL), and TF×IDF weights without pruning REL-tagged nodes (tf-idf).

sim(att) outperforms the *baseline*, e.g., 74.30 vs. 73.58 (although only when using 2 or 3 sentences). In any case, almost all the results are interesting since, when the number of sentences decreases, the computational complexity of TK applied to the reduced trees becomes much lower. We measured a decrease of 5 times of the training time required by the reranker. However, sentence selection may be considered too coarse-grained to be effective.

Tree pruning results Given word-level weights, either by TF×IDF or the attention model, we prune the parse trees according to the strategies described in Section 5.2: pruning using the weights of the (i) attention model (*attn*); (ii) attention model but retaining all REL-tagged nodes (*attn-REL*); (iii) TF×IDF model preserving all REL nodes (*tf-idf*). The experiments with TF×IDF without REL-tagging are not reported due to poor performance.

Figure 4 compares several pruning thresholds in terms of MAP on both the development and the test sets. The x-axis values indicate the percentage of nodes that were removed after pruning the question parse trees. Pruning results rather beneficial: after an initial performance decrease, MAP improves the *baseline* model (state of the art). For example, according to the results on the dev. set, *attn-REL*, which uses attention weights and also preserves the REL nodes, allows us to reduce the size of the trees by 55%, obtaining a MAP value of 74.15 ($+0.57$ with respect to using the full trees). This improvement is also statistically significant at 95% with respect to the $baseline$. The increase in performance is much more evident on the test set, where the best MAP of *attn-REL* is 78.95, obtained by pruning 50% of the nodes. Note that the best pruning threshold on the development set, i.e., corresponding to 55% of filtering, when used for the test set, achieves a MAP of 77.82, which, even if lower than the top result, 78.95, still outperforms the 76.70 MAP of the winner system of Semeval 2016 (Nakov et al., 2015); compare tables 5a and 5b. The other two models, based on TF×IDF and attention weights without preserving REL nodes, do not improve MAP, although they can produce a great speedup with a small loss in MAP. Figure 3 reports the running times of the three pruning models above with respect to the percentage of pruned nodes. All pruning strategies can reduce the size of the trees significantly, and consequently reduce the running time, for all weight sources. For example, the most accurate model on the dev. set (see Figure 4) filters 55% of the nodes, speeding up training by 5 times, i.e., 530 versus $2,580$ seconds.

7 Conclusions

In this paper, we showed that TK-based models achieve the state of the art in cQA. Nevertheless, such models are affected by noise in the syntactic structure and redundant information, typically added by Web users when formulating or answering forum questions. We proposed to alleviate this problem selecting more significant text segments from the question text. For this purpose, we used unsupervised meth-

Model	**MAP**	AvgRec	MRR
sim+rank+TK (*baseline*)	76.15	90.79	84.76
Tree Pruning (attn+REL)	**77.82**	91.31	84.64
sim(tf-idf)	76.28	90.05	83.38
sim(att)	75.85	89.95	81.31

(a) Results of selected models on the test set using the best pruning threshold of the development set.

Model	**MAP**	AvgRec	MRR
UH-PRHLT	76.70	90.31	83.02
ConvKN	76.02	90.70	84.64
KeLP	75.83	91.02	82.71

(b) SemEval top 3 systems on the test set (Nakov et al., 2016).

Table 5: Comparison between the best systems of this paper and the SemEval Challenge.

ods and supervised methods based on LSTM with attention weights. The results show that supervised text selection can improve unsupervised models, thus enhancing the performance of a tree-kernel-based reranker. Additionally, using less text produces a boost in the speed of tree kernels —up to five times the speed of the model using all the sentences— while also improving MAP. Finally, our model achieves a top result with respect to the top performing systems submitted to the SemEval 2016 task on cQA. In the future, we would like to experiment with more advanced techniques that have been shown successful for traditional text summarization, e.g., using discourse structure.

Acknowledgements

This research was performed by the Arabic Language Technologies (ALT) group at the Qatar Computing Research Institute (QCRI), HBKU, part of Qatar Foundation, within the Interactive sYstems for Answer Search (Iyas) project, which is developed in collaboration with MIT-CSAIL.

References

Lloyd Allison and Trevor Dix. 1986. A Bit-string Longest-common-subsequence Algorithm. *Inf. Process. Lett.*, 23(6):305–310, December.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and Question Selection for Question Answering on Arabic and English Fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 896–903, San Diego, California, June. Association for Computational Linguistics.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Salvatore Romeo, and Alessandro Moschitti. 2016. Selecting sentences versus selecting tree constituents for automatic question ranking. In *Proceedings of the 26th International Conference on Computational Linguistics*, Osaka, Japan.

Yunbo Cao, Huizhong Duan, Chin-Yew Lin, Yong Yu, and Hsiao-Wuen Hon. 2008. Recommending Questions Using the Mdl-based Tree Cut Model. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 81–90, New York, NY, USA. ACM.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, and Alessandro Moschitti. 2016. Learning to re-rank questions in community question answering using advanced features. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM '16, Indianapolis, IN, October. Association for Computational Linguistics.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning Hybrid Representations to Retrieve Semantically Equivalent Questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching Questions by Identifying Question Topic and Question Focus. In *ACL*.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying Deep Learning to Answer Selection: A Study and An Open Task. *CoRR*, abs/1508.01585.

Simone Filice, Giuseppe Castellucci, Danilo Croce, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. 2015a. KeLP: a Kernel-based Learning Platform in Java. In *The workshop on Machine Learning Open Source Software (MLOSS): Open Ecosystems*, Lille, France, July. International Conference of Machine Learning.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015b. Structural Representations for Learning Relations between Pairs of Texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1003–1013, Beijing, China, July. Association for Computational Linguistics.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1116–1123, San Diego, California, June. Association for Computational Linguistics.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. UH-PRHLT at SemEval-2016 Task 3: Combining Lexical and Semantic-based Features for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 814–821, San Diego, California, June. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Wei-Ning Hsu, Yu Zhang, and James R. Glass. 2016. Recurrent Neural Network Encoder with Attention for Community Question Answering. *CoRR*, abs/1603.07044.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *CIKM*.

Thorsten Joachims. 1999. Making Large-scale Support Vector Machine Learning Practical. In *Advances in Kernel Methods*. MIT Press, Cambridge, MA, USA.

Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. KDD.

Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA. ACM.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP '01, pages 118–125, Pittsburgh, PA, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 318–329. Springer Berlin Heidelberg.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. SemEval-2015 Task 3: Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 269–281, Denver, Colorado, June. Association for Computational Linguistics.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *International Conference on Learning Representations*.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer reranking. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 741–750, Portland, Oregon, USA.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based Deep Learning Models for non-factoid answer selection. *CoRR*, abs/1511.04108.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional Neural Networks vs. Convolution Kernels: Feature Engineering for Answer Sentence Reranking. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1268–1278, San Diego, California, June. Association for Computational Linguistics.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR*.

Michael Wise. 1996. YAP3: Improved Detection of Similarities in Computer Program and Other Texts. In *Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '96, pages 130–134, New York, NY, USA.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 44, page 401.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *CIKM*.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *ACL*.