Learning to Rank Non-Factoid Answers: Comment Selection in Web Forums

Kateryna Tymoshenko
University of Trento
Povo (Trento), Italy
kateryna.tymoshenko@unitn.it

Daniele Bonadiman
University of Trento
Povo (Trento), Italy
d.bonadiman@unitn.it

Alessandro Moschitti*
Qatar Computing Research Institute
HBKU, Doha, Qatar
amoschitti@qf.org.qa

ABSTRACT

Recent initiatives in IR community have shown the importance of going beyond factoid Question Answering (QA) in order to design useful real-world applications. Questions asking for descriptions or explanations are much more difficult to be solved, e.g., the machine learning models cannot focus on specific answer words or their lexical type. Thus, researchers have started to explore powerful methods for feature engineering. Two of the most promising methods are convolution tree kernels (CTKs) and convolutional neural networks (CNNs) as they have been shown to obtain high performance in the task of answer sentence selection in factoid QA. In this paper, we design state-of-the-art models for non-factoid QA also carried out on noisy data. In particular, we study and compare models for comment selection in a community QA (cQA) scenario, where the majority of questions regard descriptions or explanations. To deal with such complex task, we incorporate relational information holding between questions and comments as well as domain-specific features into both convolutional models above. Our experiments on a cQA corpus show that both CTK and CNN achieve the state of the art, also according to a direct comparison with the results obtained by the best systems of the official cQA challenge, SemEval. This also shows the primary importance of coding relational information between question and answer text.

1. INTRODUCTION

Community Question Answering (cQA) enables users to freely ask questions in web forums and expect some good answers in the form of comments from other users. However, many comments are typically not pertinent to the user question, thus automatic approaches for selecting those useful are very valuable.

Previous work for automatic answer sentence selection, e.g., [22, 23, 18], has inspired recent international competitions on cQA, where answer sentences are replaced by entire comments of forum users. For example, SemEval-2016 Task 3[1] in Subtask A asks to rank the posts in comment threads that answer the question *well* before those that are just *bad* or *potentially useful*. Tab. 1 describes an example of the task: the participants are given a question, its author's ID, a set of comments along with the IDs of the users who wrote them, and their systems have to rerank the comments so that the good ones are pushed to the top. All the comments are manually annotated with their relevance with respect to the question and thus can be used for training and testing learning to rank (L2R) algorithms.

Question		
Author:	princess_naila	
Subject:	Psychiatrist in DOHA	
Body:	Could someone advise the best psychiatrist/psycologist in DOHA?	
Comments		
Author	**Comment**	**Label**
feba mariyam	i heard a good doctor in doha clinic...	**potentially useful**
princess _naila	ok..shall check that out..thank you :)	**Bad**
Equin0x	"Visit Psychiatrist clinic of Hamad Hospital located opposite ""The Center"" exactly facing KFC."	**Good**

Table 1: Qatar Living question and comments examples

The overall setting of this task is similar to that of the answer sentence selection in the factoid QA, therefore an intuitive approach would be to employ state-of-the-art answer sentence selection models. However, it should be noted that the cQA questions are typically non-factoid, e.g., asking for opinions or advice. Moreover, the text involved in this scenario is rather noisy, making it difficult to outperform simple bag-of-word models.

To tackle such complexity, we rely on state-of-the-art machine learning methods such as CTKs and CNNs . Thus, we design L2R models for comment selection using (i) CTKs based on our previous successful CTK models for factoid QA [18, 20, 21], and (ii) new CNNs. In particular, we improve the CTK models using some basic filtering strategies for building robust tree structures that can deal with the noisy and relatively larger size of forum comments. At the same time, we deal with the more challenging nature of non-factoid questions by encoding relational information in both CTKs and CNNs. The experiments with the official data of SemEval cQA challenge [14] show that CNNs are on par with CTKs: we attain state-of-the-art accuracy using two completely different models, although we also needed to define some task specific features. In particular, our models evaluated on the official test set produce a MAP of 78.80, just 0.4 less than the best system (79.19), and would have ranked second[2].

2. STRUCTURES FOR CTKs

*Professor at University of Trento, DISI.

[1] http://alt.qcri.org/semeval2016/task3/

[2] http://alt.qcri.org/semeval2016/task3/index.php?id=results

CIKM'16 , October 24 - 28, 2016, Indianapolis, IN, USA

ISBN 978-1-4503-4073-1/16/10. . . $15.00

DOI: http://dx.doi.org/10.1145/2983323.2983906

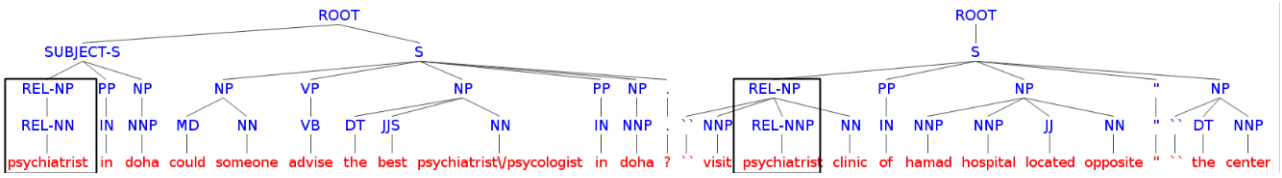

Figure 1: SH tree for the Q/Comment pair.

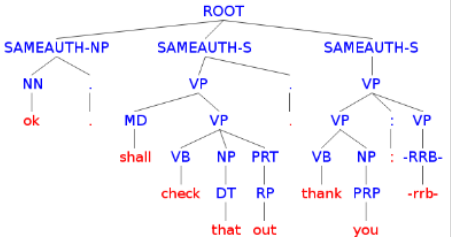

Figure 2: CONST tree enriched with question author information

Severyn et al. [18] built a reranker for QA applying CTKs to a shallow structural representation of question and answer sentences, called **SH**. This is constituted by two trees with lemmas, at leaf level and their part-of-speech (POS) tags at the preterminal level. The authors mark the lemmas that occur in both question and answer passage by prepending the **REL** tag to the labels of the corresponding POS-tag nodes and their parents, thus encoding information about their relatedness.

In our work, we adjust their representation to fit the cQA task, i.e., we enrich the structures with additional relational information and cQA-specific thread knowledge. Additionally, we found that using a constituency structure **CONST** is more effective than SH. Moreover, differently from [18], we do not remove the punctuation marks from the trees, as, for example, the comments that ask additional questions and thus contain a question mark are unlikely to answer the original question.

Most importantly, we encode the following cQA-related information: (i) the question subject is a separate subtree under the **SUBJECT-S** root in the question tree; (ii) following the intuition that the question author unlikely will correctly answer his/her own question, we add the **SAMEAUTH** label to the sentence nodes in the comment tree. Additionally, when constructing the CONST representation of the question tree, we do not use the question body, but only its subject[3]. Figure 1 provides an example of the **REL-** encoding in the SH tree with lemma *psychiatrist* marked with REL- in both trees and question subject separated from its body. Fig. 2 illustrates a CONST representation of the author's comment to her own question.

3. RELATIONAL NNs FOR cQA

We propose a NN for matching sentence pairs on the same research line of answer selection [17, 9, 5, 23]. The general architecture is depicted in Fig. 3. It includes two main components (i) two sentence encoders that map input documents i into fixed size m-dimensional vectors x_{s_i}, and (ii) a feed forward NN that computes the similarity between the two sentences in the input.

Let d be the size of the word embeddings, the sentence encoders are composed of (i) a sentence matrix $\mathbf{s}_i \in \mathbb{R}^{d \times |\mathbf{i}|}$ obtained by concatenating the word vector of the corresponding word in the input sentence $\mathbf{w}_j \in \mathbf{s}_i$, and (ii) a sentence model $f : \mathbb{R}^{d \times |\mathbf{i}|} \rightarrow \mathbb{R}^m$, which maps the sentence matrix to a fixed size sentence embedding $x_{s_i} \in \mathbb{R}^m$. The choice of the sentence model plays a crucial role as the resulting representation of the input sentences affects the successive steps of computing their similarity. Previous work uses different types of sentence models for matching sentence pairs such as LSTM, distributional sentence model, and convolutional sentence model. In particular, the latter is composed of a sequence of convolution-pooling feature maps and has achieved the state of the art in various NLP tasks [7, 8].

Here, we implement a sentence model with a convolutional operation followed by a k-max pooling layer with $k = 1$. This choice has been proved successful for factoid QA [17]. The sentence vectors, x_{s_i}, are concatenated together and given as input to standard NN layers, which are constituted by a non-linear hidden layer and a sigmoid output layer. The sentence encoder, $x_{s_i} = f(s_i)$, outputs a fixed-size vector representation of the input sentence s_i. In the remainder of the paper, we will refer to the question and the comment representations as the question embedding (QE) and the comment embedding (CE). Similarly the output of the penultimate layer of the network (the hidden layer whose output feeds the final classification layer) is a compact representation of the input question and comment pair, which we call Joint Embedding (JE).

3.1 Injecting Relational features

Despite the fact that NNs can automatically generate embeddings from raw inputs, adding manually engineered features can be useful for solving complex tasks such as cQA. In particular, we can add any discrete features to the model, as described in [2], i.e., use an additional lookup layer for every feature, $feat$. Word features can be added to the model by augmenting the word embedding with the corresponding feature embedding. Given a word, w_j, its final word embedding is defined as $\mathbf{w}_j \in \mathbb{R}^d$, where $d = d_w + d_{feat}$. Specific cQA features, $\mathbf{x}_{feat}$, can be inserted into the join layer using feature embedding; the resulting vector is $\mathbf{x}_{s_1+s_2+x_{feat}} \in \mathbb{R}^t$, where $t = 2m + d_{feat}$.

Sec. 2 shows that establishing relational links (REL nodes) between a question (Q) and comment (C) is very important for solving the cQA tasks. We enable our NNs to capture the connections between words shared by Q and C by encoding relational links as binary word features. In particular, we associate each word w in the input sentences with a *word overlap* feature $o \in \{0, 1\}$, where $o = 1$ means that w is shared by both Q and C, i.e., overlaps, $o = 0$ otherwise. Basically the word-overlap vector plays the role of the REL tag added to the CTK structures (see Sec 2).

Nicosia et al. [15] describe other important features for cQA, including whether (i) the author of the comment is the same of the question (SAMEAUTH); and (ii) the comment in consideration is followed by another comment from the author of the question (SAMEAUTH$^+$). These two sentence features are binarized and added to the network at the join layer.

4. EXPERIMENTS

In these experiments, we compare CTKs and CNNs, also combining them with traditional feature vector representations.

4.1 Experimental setup

Experimental dataset. We use the Subtask A *Question-Comment*

[3]Our preliminary experiments showed that adding question body to the CONST representation does not affect the performance.

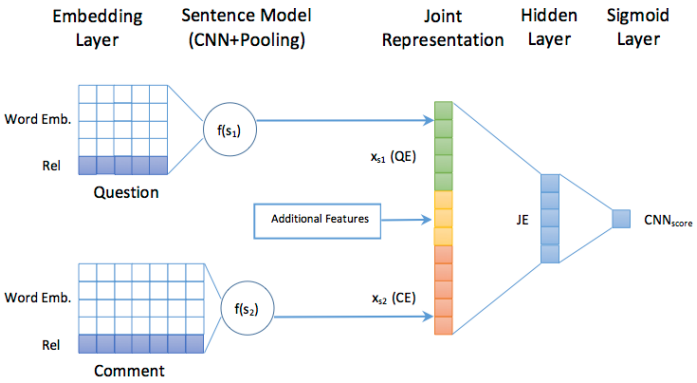

Figure 3: Neural Network model

Similarity subset of the Semeval-2016 Task 3 English cQA dataset[4]. This enables a direct comparison with results of challenge. The dataset of questions and comments was extracted from the *Qatar Living forum*[5]. For each question, the first ten comments were collected and manually annotated as good, i.e., answering the question, and potentially useful or bad. The Potentially useful class was relabeled as bad during the challenge evaluation. The train, development and test sets contain 1790, 244 and 327 questions.

QA metrics. We report our results in terms of Mean Average Precision (MAP)[6] and Mean Reciprocal Rank (MRR).

CTKs. We trained our models with SVM-light-TK[7]. It enables the use of CTKs [13] in SVM-light [6]. We used the partial and the subset tree kernels (PTK and STK) with their default parameters and the polynomial kernel (P) of degree 3 on all feature vectors.

Preprocessing We truncate all the comments to 2000 symbols and all the sentences to 70 words. When generating the structural representations described in Section 2, we used the first three sentences of a question or a comment. We used the Illinois chunker [16] and the Stanford CoreNLP [10] toolkit for the needed preprocessing.

Signatures. If a specific user always signs their posts with the same phrase, e.g., "*The tough gets going.*.", we consider this phrase irrelevant and remove it. More specifically, we delete all the strings with length exceeding 20 characters that occur at the end of more than one comment by the same user.

cQA-relevant features vectors (QF). We used the thread-level features by one of the top-performing SemEval-2015 systems [15]. Among others, they include the features encoding whether the comment (c) is authored by the question (q) author, whether c contains a question, an acknowledgment word or a URL, and other intuitions.

4.2 Neural Network experiments

We pre-initialize the word embeddings with skipgram embedding of dimensionality 50 trained on the English Wikipedia dump [12]. The input sentences are encoded with fixed-sized vectors using a convolutional operation of size 5 and a k-max pooling operation with $k = 1$. We use a single non-linear hidden layer (with hyperbolic tangent activation, Tanh), whose size is equal to the size of the sentence embeddings, i.e., 200. The network is trained using SGD with shuffled mini-batches using the Rmsprop update rule. The model is trained until the validation loss stops improving.

For computational reasons, we decided to limit the size of the input sentences to 100 words. This did not degrade the observed performance. To improve learning generalization and avoid co-adaptation of features (which negatively impact the transfer capabilities of the sentence embeddings), we opted for adding dropout [19] between the layers of the network. Table 2 reports the results of our NNs on the development set. The first sentence model is a word embedding averaging model (W2V), where the sentence embeddings are calculated by averaging the word vectors. Then, two strong baseline sentence models are presented, a Long-Short Term Memory (LSTM) sentence model [4] and our vanilla CNN, presented in Section 3. Our CNN, without any additional features, outperforms W2V, obtaining similar accuracy than LSTM. Thus, we decided to use the convolutional sentence model mainly for computational reasons. The second part of the table shows CNN incrementally enhanced by the $overlap$, SAMEAUTH and SAMEAUTH$^+$ features (see Sec. 3.1) for further improving its performance.

Encoders	MAP	MRR	Features	MAP	MRR
W2V	0.6284	68.53	CNN	0.6496	71.26
LSTM	0.6477	**71.78**	$+overlap$	0.6648	73.46
CNN	**0.6496**	71.26	+SAMEAUTH	0.6684	72.56
			+SAMEAUTH$^+$	**0.6741**	**73.64**

Table 2: NNs results on the development dataset

4.3 Results and Discussion

Table 3 reports the performance of our systems and compares it to the state of the art. Its first four lines report the performance of the top systems in the Semeval-2016 Subtask A competition[8]. The participants were allowed to submit one primary and two contrastive runs. The teams were ranked according to the MAP score obtained by their primary system (the official rank is reported in parentheses). The remainder of the table describes the performance of our systems on the development (DEV) and on the test (TEST) sets. We did not use DEV to tune any parameters.

The *Kernel* column reports the name of the kernel used in SVMs. V_{QF} is an SVM using a polynomial kernel over the QF feature vector. CNN is the convolution neural network described in Sec. 3. CTK_{SH} and CTK_C are the CTK-based SVMs trained on the SH and CONST representations described in Sec. 2, respectively. CTK+V denotes the composite kernel that is a sum of two kernels.

Notably, both CTK models that encode only one cQA task-specific intuition (i.e., whether the question and the comment have the same author) are only slightly outperformed by CNNs and achieve an MAP only one point lower than ConvKN-primary, the second best-ranking system in official ranking of the competition.

Additionally, we observe that CTK_{SH} with the PTK and CTK_C with STK obtain almost the same performance. This is important, as PTK generates a richer feature space but STK has a lower computational complexity. Training an SVM with PTK and STK on CTK_C took 4213.93 and 1006.58 cpu-seconds, respectively. Next, the CTK and V combinations, CTK_C+V_{QF} (STK) and CTK_{SH}+V_{QF} (PTK), obtained the MAPs of 78.78 and 78.80 on TEST, respectively. They would have ranked second in the competition.

Finally, since both CNNs and CTKs achieve the state of the art on cQA, and since our earlier work [21] showed that combining CNNs and CTKs improves factoid QA, we experimented with combining the two approaches for cQA. We used the question, comment and joint embeddings learned by CNNs as feature vectors in V and combined them with QF and CTK_C. We have experimented with all possible combinations: Sec. 3 of Table 3 lists the best ones. The subscripts QE and CE indicate the use of the question and com-

[4] http://alt.qcri.org/semeval2016/task3/index.php?id=description-of-tasks

[5] http://www.qatarliving.com/forum

[6] the official Subtask A metric

[7] http://disi.unitn.it/moschitti/Tree-Kernel.htm

[8] http://alt.qcri.org/semeval2016/task3/data/uploads/semeval2016_task3_submissions_and_scores.zip

Models	Kernel	DEV		TEST	
		MAP	MRR	MAP	MRR
1. Baseline models					
Kelp-primary [3] (#1)	n/a	n/a	n/a	79.19	86.42
ConvKN-contrastive1 [1]	n/a	n/a	n/a	78.71	86.15
SUper team-contrastive1 [11]	n/a	n/a	n/a	77.68	84.76
ConvKN-primary [1] (#2)	n/a	n/a	n/a	77.66	84.93
2. CNN and CTK models					
V_{QF}	P	63.45	70.51	73.50	82.98
CNN	n/a	67.41	73.64	77.13	83.85
CTK_{SH}	PTK	64.10	71.97	76.67	83.53
CTK_C	STK	65.30	73.24	75.42	82.35
CTK_C	PTK	63.82	70.53	76.39	82.94
CTK_{SH}+V_{QF}	PTK, P	68.45	74.49	**78.80**	86.16
CTK_C+V_{QF}	STK, P	67.26	74.07	**78.78**	86.26
3. Combining CTK and CNN models					
$V_{QE\|CE}$	P	65.63	72.69	75.15	82.37
$V_{QE\|CE\|QF}$	P	**68.17**	75.32	**77.22**	83.98
CTK_C+$V_{QE\|CE}$	STK,P	66.71	**75.18**	**76.25**	83.33
CTK_C+$V_{QE\|CE\|QF}$	STK,P	**68.92**	**76.61**	**77.25**	**84.16**

Table 3: Performance of CTK, CNN and QF models on the development (DEV) and test (TEST)

ment embeddings (Sec. 3) as the feature vector, whereas | denotes that the respective vectors were concatenated into a single vector.

None of the combinations improved over the CTKs models with QF features, however, an interesting finding is that, in general, V systems, which use only embeddings as features outperform V_{QF}. Concatenating the QE, CE and QF into a single feature vector results in a further improvement, e.g., $V_{QE|CE|QF}$ achieves an MAP of 77.22 on the TEST, which is only 0.44 points lower than the MAP of the **#2** ranked system. CTK_C+$V_{QE|CE}$, a combination of QE and CE with CTK_C (without QF) slightly improves over CTK_C (using STK) on both DEV and TEST sets. However, the CTK_C+$V_{QE|CE|QF}$, a combination of CTKs, CNN and QF, scores lower than $CTK_C + V_{QF}$, thus additional investigation is needed to understand how to successfully combine CTK and CNN.

5. CONCLUSION

In this paper, we studied state-of-the-art QA methods and applied them to cQA. Several challenges have to be tackled in this kind of real-world applications: the questions are mainly non-factoid and the text is typically informal and noisy. We provided two solutions based on tree kernels applied to robust syntactic structures and for the first time applied them to comment reranking tasks in cQA. Additionally, we designed new CNNs for comment reranking, which exploit cQA features as well as important relational information between questions and comments. The results show that tree kernels and CNNs can achieve the state of the art. The high quality of our solutions is also confirmed by the official results of a cQA challenge, among which our best system would have achieved the second position. The most important aspect of our future work will be to investigate ways of successfully combining CTKs and CNNs.

Acknowledgements

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action) and by an IBM Faculty Award.

References

[1] A. Barrón-Cedeño, G. Da San Martino, S. Joty, A. Moschitti, F. Al-Obaidli, S. Romeo, K. Tymoshenko, and A. Uva. ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on Arabic and English fora. In *SemEval-2016*. ACL, 2016.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[3] S. Filice, D. Croce, A. Moschitti, and R. Basili. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *SemEval-2016*. ACL, 2016.

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *NIPS*. 2014.

[6] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*. ACM, 2002.

[7] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *ACL*, 2014.

[8] Y. Kim. Convolutional neural networks for sentence classification. Doha, Qatar, 2014.

[9] Z. Lu and H. Li. A deep architecture for matching short texts. In *NIPS*, 2013.

[10] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: System Demonstrations*, 2014.

[11] T. Mihaylova, P. Gencheva, M. Boyanov, I. Yovcheva, T. Mihaylov, M. Hardalov, Y. Kiprov, D. Balchev, I. Koychev, P. Nakov, I. Nikolova, and G. Angelova. Super team at semeval-2016 task 3: Building a feature-rich system for community question answering. In *SemEval-2016*. ACL, 2016.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS 26*, 2013.

[13] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*. Springer, 2006.

[14] P. Nakov, L. Màrquez, A. Moschitti, W. Magdy, H. Mubarak, A. A. Freihat, J. Glass, and B. Randeree. SemEval-2016 task 3: Community question answering. In *SemEval*. ACL, 2016.

[15] M. Nicosia, S. Filice, A. Barrón-Cedeño, I. Saleh, H. Mubarak, W. Gao, P. Nakov, G. Da San Martino, A. Moschitti, K. Darwish, L. Màrquez, S. Joty, and W. Magdy. QCRI: Answer selection for community question answering - experiments for Arabic and English. In *SemEval*, 2015.

[16] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS*, 2001.

[17] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*. ACM, 2015.

[18] A. Severyn, M. Nicosia, and A. Moschitti. Learning adaptable patterns for passage reranking. *CoNLL-2013*, page 75, 2013.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[20] K. Tymoshenko and A. Moschitti. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *CIKM*. ACM, 2015.

[21] K. Tymoshenko, D. Bonadiman, and A. Moschitti. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *NAACL-HLT*. ACL, 2016.

[22] M. Wang and C. D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *ACL*, 2010.

[23] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *CoRR*, 2014.