Convolutional Neural Networks and Convolution Kernels for Learning to Rank Answers

anonymous*
anonymous
anonymous
none, none none
anonymous@anonymous.com

ABSTRACT

Recent work has shown that convolution tree kernels (CTKs) and convolutional neural networks (CNNs) obtain the state of the art in answer sentence reranking. Additionally, their combination in Support Vector Machines (SVMs) is promising as it can exploit the structural syntactic properties captured by CTK in synergy with the representation layers learned by CNNs. However, such layers are constructed according to a classification function, which is not directly exploitable in the preference ranking algorithm of SVMs. In this work, we define a new learning algorithm combining ranking with CTKs along with classification with CNNs. Our experiments on two well-known and rather different datasets, WikiQA for answer sentence selection and SemEval Community Question Answering for comment selection, clearly show the benefit of our approach.

CCS CONCEPTS

•**Information systems** →**Learning to rank;**

KEYWORDS

Community Question Answering, Learning to rank

ACM Reference format:
anonymous. 2017. Convolutional Neural Networks and Convolution Kernels for Learning to Rank Answers. In *Proceedings of SIGIR conference, Tokyo, Japan, August 2017 (SIGIR'17),* 4 pages.
DOI: XXX

1 INTRODUCTION

Recent work on learning to rank (L2R) has shown that deep learning and kernel methods are two very effective approaches, given their ability of engineering features. In particular, in question answering, Convolutional Neural Networks (CNN), e.g., [7, 12, 19] can automatically learn the representation of question and answer passage (Q/AP) in terms of word embeddings and their non-linear transformations. These are then used by the other layers of the network to measure Q/AP relatedness. In contrast, Convolution Tree Kernels (CTK) can be applied to relational structures built on top of syntactic/semantic structures derived from Q/AP text [16]. CNNs as well as CTKs can achieve the state of the art in ranking questions or APs. Considering their complementary approach for generating features, studying ways to combine them is very promising. Tymoshenko et al. [16] investigated the idea of extracting layers from CNNs and using them in a kernel function to be further combined with CTKs in a composite reranking kernel [14]. The resulting kernel was used in an SVM^{Rank} [3] model, which obtained a significant improvement over the individual methods. However, the simple use of CNN layers as vectors in a preference ranking approach is intutively not optimal since such layers are basically learnt in a classification model, thus they are not optimized for SVM^{Rank}.

In this work, we further compare and investigate different ways of combining CTKs and CNNs in the reranking settings. In particular, we follow the intuition that as CNNs learn the embeddings in a classification setting they should be used in the same way for building the reranking kernel. Therefore, we propose a hybrid reranking-classification kernel based on preference ranking, which consists in (i) a standard reranking kernel based on CTKs applied to the Q/AP structural representations; and (ii) a classification kernel based on the embeddings learned by neural networks. The intuition over the hybrid models is to add CNN layer vectors and not their difference to the difference between CTKs. That is, CNN layers are still used as in the classification setting whereas CTKs follow the standard SVM^{Rank} approach.

We tested our proposed models on the answer sentence selection benchmark, WikiQA [18], and the benchmark from cQA SemEval-2016 Task 3.A[1] corpus. We show that the proposed hybrid kernel consistently outperforms standard reranking models in all settings.

2 ANSWER SENTENCE SELECTION

We focus on two question answering subtasks: answer sentence selection task (AST) and the comment selection task from cQA.

AST consists in selecting correct answer sentences (i.e., an AP compose of only one sentence) for a question Q from a set of candidate sentences, $S = \{s_1, ..., s_N\}$. In factoid question answering, Q typically asks for an entity or a fact, e.g., time location and date. S is typically a result of so-called *primary search*, a result of fast-recall/low-precision search for potential answer candidates. For example, it could be a set of candidate APs returned when running a search engine over a large corpus using Q as a search query. Many such APs are typically not pertinent to the original question, thus automatic approaches for selecting those useful are very valuable.

cQA proposes a task similar to AST, where Q is a question asked by a user in a web forum and S are the potential answers to the question posted as the comments by other users. Again, many

*anonymous

SIGIR'17, Tokyo, Japan

DOI: XXX

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

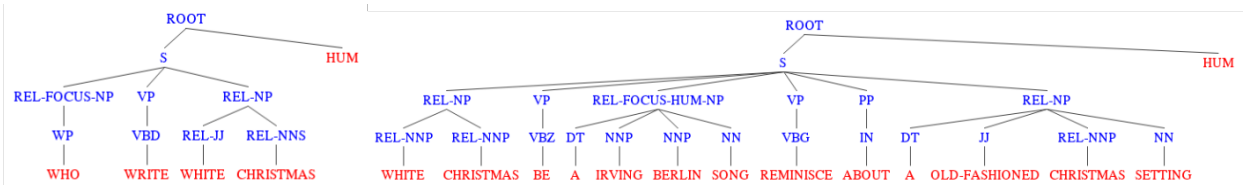

Figure 1: Shallow chunk-based tree representation of a question in the Q/AP pair: Q: *"Who wrote white Christmas?"*, AP: *"White Christmas is an Irving Berlin song"*.

comments in a cQA thread do not contain an answer to the original question, thus raising the need for automatic comment selection.

The crucial features for both tasks consist in information about the relations between Q and a AP. Manual feature engineering can provide competitive results [10], however, it requires significant human expertise in the specific domain and is time-consuming. Thus, machine learning methods for automatic feature engineering are extremely valuable.

3 STANDALONE CTK MODELSs

Our baseline are the standalone CTK and CNN models originally proposed in [13] and further advanced in [16, 17]. The following subsections provide a brief overview of these models.

3.1 CTK structures

The CTK models are applied to syntactic structural representations of Q and AP. We used a shallow chunk-based and constituency tree representations in cQA [16] and AST [17], respectively. We follow the tree construction algorithms provided in the work above. Due to the space restrictions we present only high-level details below.

A shallow chunk-based representations of a text contains lemma nodes at leaf level and their part-of-speech (POS) tag nodes at the preterminal level. The latter are further grouped under the chunk and sentence nodes. A constituency tree representation is an ordinary constituency parse tree. In all representations, we mark lemmas that occur in both Q and AP by prepending the **REL** tag to the labels of the corresponding preterminal nodes and their parents. Moreover, in AST setting, that is a factoid question answering with well-researched question classification mechanisms [5], we enrich the representation with the question class and focus information. Additionally, we mark AP chunks containing name entities that match the expected answer type of the question by prepending *REL-FOCUS-<QC>* to them. Here, the $< QC >$ placeholder is substituted with the actual question class. Fig. 1 illustrates a shallow chunk-based syntactic structure enriched with relational links.

3.2 Convolutional Neural Networks

A number of NN-based models have been proposed in the research line of answer selection [2, 20]. Here, we employ the NN model described in [16] and depicted in Fig. 2. 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.

We use a sentence model built with a convolution operation followed by a k-max pooling layer with $k = 1$. The sentence vectors, x_{s_i}, are concatenated together and given in 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 (we will refer to $f(s_i)$ as question embedding, QE, and answer embedding, AE, respectively).

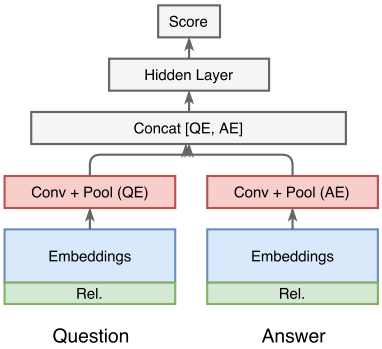

Figure 2: CNN architecture to compute the similarity between question and answer.

Additionally, we encode the relational information between Q and AP, by injecting relation features into the network. In particular, we associate each word w of the input sentences with a *word overlap* binary feature indicating if w is shared by both Q and AP.

It should be noted that the network is trained in a classification setting (e.g., with a cross-entropy loss function). By construction, at each layer the network performs non-linear transformations of the input space so that the resulting vector in input to the output layer can be easily classified as a positive or negative example. The final ranking is then given by sorting according to the output score. This means that the layers trained with the network are optimized for a classification task and are in general non-optimal to be used in preference reranking. Indeed, the latter, given two vectors $\vec{x}_i$ and $\vec{x}_j$ can be seen as a classification model over $\vec{x}_i - \vec{x}_j$. In case, such vectors are NN layers the difference may delete useful information derived during the convolutional operations. Therefore, intuitively a classification of $\vec{x}_i$, rather than $\vec{x}_i - \vec{x}_j$ may produce better results.

4 HYBRID LEARNING TO RANK MODEL

We represent a Q/AP pair as $p = (q, a, \vec{x})$, where q and a are the structural representations of Q and AP (as in Sec. 3), and $\vec{x}$ is a feature vector that incorporates the features characterizing the Q/AP pair (e.g., similarity features between Q and AP or their embeddings learned by a neural network).

Reranking kernel. This kernel captures differences between two Q/AP pairs, p_1 and p_2, and predicts which pair should be ranked higher, i.e., in which pair, AP better answers Q. In the reranking setting, a training/classification instance is a pair of Q/AP pairs, $\langle p_1 = (q, a_1, \vec{x}_1),\quad p_2 = (q, a_2, \vec{x}_2)\rangle$. The instance is positive if p_1 is ranked higher than p_2, and negative otherwise. One approach for producing training data is to form pairs both using $\langle p_1, p_2\rangle$ and $\langle p_1, p_2\rangle$, thus generating both positive and negative examples. However, since these are clearly redundant as formed by the same

Algorithm 1 Generating training data for reranking

Require: S_{q+}, S_{q-} - $(q, a, \vec{x})$-triplets for correct and wrong answer sentences per question Q

```
E+ ← ∅, E− ← ∅, flip ← true
for all s+ ∈ Sq+ do
    for all s− ∈ Sq− do
        if flip == true then
            E+ ← E+ ∪ (s+, s−)
            flip ← false
        else
            E− ← E− ∪ (s−, s+)
            flip ← true
return E+, E−
```

members, it is more efficient training with a reduced set of examples such that members are not swapped. Algorithm 1 describes how we generate a more compact set of positive (E_+) and negative (E_-) training examples for a specific Q.

Given a pair of examples, $\langle p_1, p_2\rangle$ and $\langle p'_1, p'_2\rangle$, we used the following preference kernel [14]:

$$R_K(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle) = K(p_1,p'_1)+ \\ +K(p_2,p'_2) - K(p_1,p'_2) - K(p_2,p'_1), \quad (1)$$

which is equivalent to the dot product between vector subtractions, i.e., $\big(\phi(p_1) - \phi(p_2)\big) \cdot \big(\phi(p'_1) - \phi(p'_2)\big)$, used in preference reranking, where ϕ is a feature map. Additionally, we indicate (i) with R_{TK} the preference kernel using TKs applied to q and a trees, i.e., $TK(p_i,p_j) = TK(q_i,q_j) + TK(a_i,a_j)$; and (ii) with R_V the preference kernel applied to vectors, i.e., $V(p_i,p_j) = V(\vec{x}_i,\vec{x}_j)$. Our final reranking kernel is:

$$RK(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle) = R_{TK}(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle)+ \\ +R_V(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle) \quad (2)$$

Now if we substitute the explicit form for R_V, we have:

$$RK(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle) = R_{TK}(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle)+ \\ +V(p_1,p'_1) + V(p_2,p'_2) - V(p_1,p'_2) - V(p_2,p'_1)$$

Since our vectors are internal network layers in order to not lose important information with differences (operated by R_V), we only keep $V(p_1,p'_1)$ (or equivalently $V(p_2,p'_2)$), i.e.,

$$RK(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle) = R_{TK}(\langle p_1,p_2\rangle,\langle p'_1,p'_2\rangle) + V(p_1,p'_1) \quad (3)$$

Note that our approach also works when using Alg. 1.

5 EXPERIMENTS

In our experiments, we compare various methods of combining CTKs and CNNs, using reranking and a hybrid reranking kernel.

5.1 Experimental setup

Sentence selection dataset: WikiQA is a dataset created for open domain QA with questions sampled from the Bing query logs and candidate answers extracted from the summary paragraphs of the associated Wikipedia pages. Table 1 provides the statistics regarding this dataset. As in [19], in our evaluations, we discard questions that have either only correct or only incorrect answer candidates.

Dataset	Q	AP	Q with AP
WikiQA-train	2,118	20,360	873
WikiQA-test	633	6,165	243
WikiQA-dev	296	2,733	126

Table 1: WikiQa statistics.

cQA SemEval-2016 dataset: We employ the Semeval-2016 Task 3, Subtask A English cQA dataset[2]. It was used in the Semeval competition thus enabling the direct comparison to the state of the art. It contains questions collected from the *Qatar Living forum*[3] with the first ten comments per question manually annotated. The train, dev. and test sets contain 1790, 244 and 327 questions, respectively.
QA metrics. We report our results in terms of Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and P@1.
CTKs. We trained our models with SVM-light-TK[4] and the partial and subset tree kernels (PTK and STK)[5] with their default parameters and the polynomial kernel (P) of degree 3 on all feature vectors.
Text Preprocessing. We used the Illinois chunker [11] and the Stanford CoreNLP [6] toolkit, v3.6.0. When experimenting with SemEval-2106, we perform preprocessing as in [17], e.g., truncate all the comments to 2000 symbols and sentences to 70 words.
Neural Network (CNN) setup. We employ the same setup as [17]: we pre-initialize the word embeddings with skipgram embedding of dimensionality 50 trained on the English Wikipedia dump [8]. We use a single non-linear hidden layer (with hyperbolic tangent activation, Tanh), whose size is equal to the size of the previous layer, i.e., the join layer. The network is trained using SGD with shuffled mini-batches using the Rmsprop update rule [15]. The model is trained until the validation loss stops improving. The size of the sentences embedding (QE and AE) and of the join layer is set as 200. We set the same parameters as [16]. We use the concatenated QE and AE vector, i.e., V, since it is reported to be the best in previous work.

5.2 Ranking with trees and embeddings

In these experiments, we evaluate the combination techniques proposed in Sec. 4 on the SemEval-2016, Task 3.A and WikiQA development (DEV) and test (TEST) sets.

We cannot run any cross-validations on the training (TRAIN) sets, since the embeddings learned in Sec. 3.2 are trained on TRAIN by construction, and therefore cross-validation on TRAIN would exhibit unrealistically high performance. Thus, we employed a *Cross Validation* setting. We train 5 models as in traditional 5-fold cross-validation on TRAIN. Then, we merge WikiQA DEV and TEST sets, split the resulting set into 5 subsets, and used i-th subset to test the model trained in i-th fold (i=1,..,5).

Table 2 reports the performance of the models. Here, Rank. corresponds to traditional reranking model described by Eq. 2 in Sec. 4. Hybrid refers to our new reranking/classification kernels described by Eq. 3. V means that the model uses a kernel applied to the embedding feature vectors only. T specifies that the model employs structural representations with a tree kernel. Finally, V+T means that both embedding feature vectors and trees are used.

[2] http://alt.qcri.org/semeval2016/task3/index.php?id=description-of-tasks
[3] http://www.qatarliving.com/forum
[4] http://disi.unitn.it/moschitti/Tree-Kernel.htm
[5] We use PTK for WikiQA and STK for SemEval as they were shown to be effective on the respective corpora in the previous work [17]

	WikiQA									SemEval-2016, Task 3.A			
	DEV			TEST			Cross Validation			DEV		TEST	
	MRR	MAP	P@1	MRR	MAP	P@1	MRR	MAP	P@1	MAP	MRR	MAP	MRR
Rank.: T	72.23	71.9	59.02	71.25	69.62	56.54	70.75±2.32	69.69±2.3	55.61±2.5	64.92	73.39	75.96	84.46
Class.: V	70.89	70.66	58.20	66.69	65.00	53.16	68.29±2.80	67.02±2.48	55.35±3.91	65.63	72.69	75.15	82.37
Rank.: V	69.35	68.76	54.92	67.49	66.41	54.43	67.00±3.41	66.01±2.25	52.14±3.92	65.29	72.65	74.48	81.77
Rank.: T+V	71.13	70.64	58.2	71.45	69.96	57.38	70.82±3.96	69.86±3.65	56.76±4.83	66.46	74.02	74.82	82.37
Hybrid: T+V	**75.5**	**74.52**	**64.75**	**73.88**	**71.99**	**61.18**	**73.52±1.87**	**71.99±1.49**	**60.73±1.19**	**68.09**	**75.09**	**77.05**	**83.75**
CNN	72.04	71.73	59.84	70.34	68.73	56.12	—	—	—	66.48	73.46	76.17	81.32
Rank.:T+K [16]	71.29	70.79	57.94	72.51	71.29	59.26	—	—	—	—	—	—	—
ABCNN [19]	—	—	—	71.27	69.14	—	—	—	—	—	—	—	—
KeLP-p (#1) [1]	—	—	—	—	—	—	—	—	—	—	—	79.19	86.42
ConvKN-p (#2) [4]	—	—	—	—	—	—	—	—	—	—	—	77.66	84.93

Table 2: Experimental results on WikiQA and SemEval-2016 Task 3.A corpora

The experiments show that, in general, a standalone model with tree kernels applied to the syntactic structures (*Rank.:T*) outperforms the standalone feature-based models using embeddings as feature vectors (V). Then, the straightforward combination of tree and polynomial kernels applied to the syntactic structural representations and embeddings (*Rank.: T+V*) does not improve over the *Rank.: T* model. At the same time, the Hybrid model consistently outperforms all the other models in all possible experimental configurations, thus confirming our hypothesis that the classification setting is more appropriate when using embeddings as feature vectors in the kernel-based ranking models. Additionally, for reference, we report the performance of the CNN we used to obtain the embeddings. It is consistently outperformed by the Hybrid model on the datasets.

Finally, in the last four lines of Tab. 2, we report the performance of the state-of-the art models from previous work on our experimental datasets. Here [16] is the traditional reranking model, which we reimplemented with *Rank.:T+V*. Our implementation obtains comparable performance on WikiQA-DEV and slightly lower performance on WikiQA-TEST. This is due to difference preprocessing.

ABCNN [19] is another state-of-the-art system based on advanced attention-based convolutional networks. All our models involving tree kernels outperform it. KeLP-p (#1) and ConvKN-p (#2) are the two top-performing SemEval 2016, Task 3.A competition systems [9]. ConvKN-p (#2) is similar to our approach, as they also employ tree kernels and embeddings. However, they employ cQA-domain-specific handcrafted features, which also consider the thread-level information, while in our work, we do not manually engineer features. Nevertheless, even though we do not employ domain-specific features, the performance of our *Hybrid:T+V* models on SemEval TEST is comparable to that of ConvKN-p (#2).

6 CONCLUSION

In this paper, we have studied and compared state-of-the-art feature engineering approaches, namely CTKs and CNNs, on the related tasks of AST and cQA. We investigated the ways of combining the two approaches into a single model and proposed a hybrid reranking-classification kernel for combining the structural representations and embeddings learned by CNNs. We showed that the combination of CTKs and CNNs with a hybrid kernel in the reranking setting outperforms the state of the art on AST and is comparable to the state of the art in cQA. In particular, in cQA, a combination of CTKs and CNNs performs comparably to the systems using domain specific features that were manually engineered.

REFERENCES

[1] S. Filice, D. Croce, A. Moschitti, and R. Basili. 2016. KeLP at SemEval-2016 Task 3: Learning semantic relations between questions and answers. *Proceedings of SemEval* 16 (2016), 1116–1123.

[2] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*.

[3] T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of ACM KDD*. ACM, 133–142.

[4] S. Joty, A. Moschitti, F. A Al Obaidli, S. Romeo, K. Tymoshenko, and A. Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and question selection for question answering on Arabic and English fora. *Proceedings of SemEval* (2016), 896–903.

[5] Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING*. Association for Computational Linguistics, 1–7.

[6] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of ACL: System Demonstrations*. 55–60.

[7] Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural Variational Inference for Text Processing. *arXiv preprint arXiv:1511.06038* (2015).

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in NIPS 26*. 3111–3119.

[9] P. Nakov, L. Màrquez, A. Moschitti, W. Magdy, H. Mubarak, A. A. Freihat, J. Glass, and B. Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. In *Proceedings of SemEval '16*. ACL.

[10] 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. 2015. QCRI: Answer Selection for Community Question Answering - Experiments for Arabic and English. In *SemEval*. http://www.aclweb.org/anthology/S15-2026

[11] V. Punyakanok and D. Roth. 2001. The Use of Classifiers in Sequential Inference. In *NIPS*. 995–1001.

[12] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*. ACM.

[13] A. Severyn, M. Nicosia, and A. Moschitti. 2013. Learning Adaptable Patterns for Passage Reranking. *CoNLL-2013* (2013), 75.

[14] L. Shen and A. K. Joshi. 2003. An SVM Based Voting Algorithm with Application to Parse Reranking. In *CONLL (CONLL '03)*. Edmonton, Canada, 9–16. DOI: http://dx.doi.org/10.3115/1119176.1119178

[15] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* 4 (2012).

[16] Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL-HLT*. 1268–1278.

[17] Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Learning to Rank Non-Factoid Answers: Comment Selection in Web Forums. In *CIKM*. ACM, 2049–2052.

[18] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *EMNLP*. ACL.

[19] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *arXiv preprint arXiv:1512.05193* (2015).

[20] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. *CoRR* (2014).