A Two-Step Neural Network Approach to Passage Retrieval for Open Domain Question Answering

Andrés Rosso-Mateus[1], Fabio A. González[1], and Manuel Montes-y-Gómez[2]

[1] MindLab Research Group,
Universidad Nacional de Colombia, Bogotá, Colombia,
{aerossom,fagonzalezo}@unal.edu.co
[2] Instituto Nacional de Astrofísica, Óptica y Electrónica, Computer Science Department, Puebla, Mexico,
mmontesg@ccc.inoep.mx

Abstract. Passage retrieval is an important subtask of question answering. Given a question and a set of candidate passages, the goal is to rank them according to their relevance to the question. This work presents a two-stage approach for solving this problem. Both stages are based on convolutional neural network architecture with a reduced set of parameters. In the first stage the network is used to identify the degree of similarity between question and candidate answers, then, in the second stage, the result of the first stage is used to re-rank the answers according to their similarity with the initial best-ranked answer in such a way that the most similar candidate answers are moved up. This approach is analogous to a pseudo-relevance feedback strategy. The experimental results suggest that the proposed method is competitive with the state-of-the-art methods, achieving a remarkable performance in three evaluation datasets.

Keywords: question answering, passage retrieval, answer ranking, pseudo-relevance feedback

1 Introduction

Question answering over open domain data is the focus of next generation web search engines [4]. This task considers three main subtasks: *i*) information retrieval, which collects the most relevant documents to the given question; *ii*) passage retrieval, which selects the passages from the returned documents more likely to contain the answer to the question; and *iii*) answer extraction, which analyzes the selected passages and extracts the exact answer to the question.

This work focuses on the passage retrieval subtask: given a question q and a set of candidate passages $\{s_1, s_2, ...s_n\}$, the goal is to rank the passages according to their similarity with the question.

Most of the state-of-the-art approaches exhibit a good performance in ranking the first candidate passage. That is, the first-ranked candidate passage frequently

contains a valid answer to the posed question. Based on this observation, in this paper, we propose a passage retrieval method based on a two-stage ranking approach. In the first stage, the passages (referred to as answers from now on) are ranked according to their similarity to the question. This initial ranking is generated by a convolutional neural network, which is applied to a matrix encoding query−answer term similarities, and returns a score that indicates the degree of similarity between the query and a candidate answer. In the second stage, passages are re-ranked based on their similarity to the first passage in the initial ranking. To generate this new ranking, a convolutional neural network is also applied but this time to a matrix encoding first_answer−passage term similarities. This strategy is analogous to the pseudo-relevance feedback method used in information retrieval where the highest ranked results are used to expand the query.

The paper is organized as follows. Section 2 presents some state-of-the-art methods for passage retrieval. Section 3 describes the method and the proposed architecture. Section 4 depicts the experimental setup in detail. Section 5 discusses the results achieved by the method in the two evaluation datasets, and finally, Section 6 presents our conclusions and future work directions.

2 Related Work

The Association for Computational Linguistics (ACL) has maintained a rank with the most successful methods for passage retrieval [1]. This list includes methods based on pure linguistic techniques, on statistical approaches, and more recently on deep learning.

Based on the hypothesis that questions can be generated from correct answers, Yu and the team of DeepMind [16] proposed a transformation model where question and answers are represented in the same space and their distance is used to get their similarity score. Le and Mikolov [6] proposed a paragraph vector (PV), which learns how to represent a variable size sentence in a vector. The learned vectors are then used to measure the similarity between question and answers. Severyn et al. [11] presented a convolutional neural network method (CNN), for ranking pairs of short texts. The goal is to learn an optimal representation of text pairs and a similarity function. A pairwise word interaction model (Pairwise CNN), is presented by He et al. [5]. They used a novel deep neural network architecture (BiLSTM) aimed to capture the relation between sequences of terms of two sentences read in both directions (left-right and right-left). They also proposed an attention mechanism to give importance to some component terms. Finally, Yang et al. [14] presented an attention-based model where the importance of terms is learned based on their correlations seen at the training phase. All these methods based on deep-learning techniques do not use any query expansion strategy to enhance the evaluation of question and answers similarities. The only method similar in spirit to ours, is the one proposed by Riezler et al. [10], which uses a Statistical Machine Translation (SMT)

technique for expanding questions with synonyms. Although it achieved good results, it could not outperform state of the art results.

3 Model Description

The method proposed in this paper is detailed in Figure 1, each of those steps is going to be detailed in the next section. The whole process consists of two stages: the training phase where the similarity model is obtained, and the testing phase where the calculated model is used to rank the question-answer pairs. During training: (1) question-answer pairs (qa-pairs) are pre-processed, (2) the similarity matrix between qa-pairs terms is calculated, (3) a convolutional neural network model is trained to predict the relevance of the answer to the question. Once the model is built it can be used to predict the rank order of candidate answers. At testing time, for a particular question, the model is applied to predict the relevance score of the set of candidate answers: (4) answers are ranked according to their scores, (5) answers are re-ranked according to their similarity with the highest ranked answer at step (4), producing a new ranking of the answers.

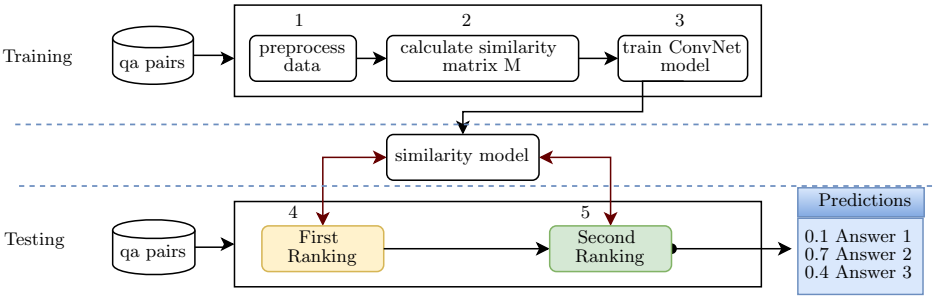

Fig. 1: Process of ranking and re-ranking qa pairs.

3.1 Step 1. Preprocess Data

Questions and candidate answers are processed using: tokenization to delimit terms; lowercasing to standardize the terms; pos-tagging, using the nltk pos-tagger [2], to extract syntactical information that will be used in salience weighting; and transforming terms to a word2vec vector representation [9], to make possible their semantic similarity comparison.

3.2 Step 2. Calculate Similarity Matrix

The similarity matrix M represents the semantic relatedness of the i-th question term and the j-th answer term according to a similarity measure. Each element $M_{i,j}$ of this matrix is a composition of a similarity score and a salience score as described by the Eq. 1.

$$M_{i,j} = scos(q_i, a_j) * sal(q_i, a_j) \tag{1}$$

Similarity Score. The similarity score for a question-answer pair terms (q_i, a_j) is calculated by means of the cosine distance between their word2vec vectors as indicated by Formula 2.

$$scos(q_i, a_j) = 0.5 + \frac{q_i \cdot a_j}{2 \left\| q_i \right\|_2 \left\| a_j \right\|_2} \tag{2}$$

In the case that there does not exist the word2vec representation for one of the terms, their similarity is measured based on their distance in Wordnet [13]. In particular, we use as similarity measure the edge distance between the first common concept related with q_i and a_j. If there is not a common concept between the terms, then we calculate the Levenshtein distance between the words [7], defined as the number of operations (insertions and eliminations of characters) needed to transform q_i to a_j.

Salience Weighting. As not all terms are equally informative for measuring text similarities [8, 3], we consider weighting the terms from the question and the answer based on part of speech functions: verbs, nouns, and adjectives are considered to be the most relevant. We model this information through a salience score.

The salience score is calculated as follows. If both terms are relevant then their score is 1. If only one of the terms is important then the score is 0.6, in the case none of them is relevant the score is 0.3. The salience function is defined in the Formula 3.

$$sal(q_i, a_j) = \begin{cases} 1 & if\ imp(q_i) + imp(a_j) = 2 \\ 0.6 & if\ imp(q_i) + imp(a_j) = 1 \\ 0.3 & if\ imp(q_i) + imp(a_j) = 0 \end{cases} \tag{3}$$

Where $imp(q_i)$ and $imp(a_j)$ are the evaluation of importance weighting function for every question and answer term. The related function returns 1 if the term is a verb, noun or adjective, otherwise, returns 0.

Finally, we sort the calculated matrix M leaving the most related terms in the top left cell, and if the number of rows or columns exceeds 40, the remaining data is truncated. This step provides an invariable representation of the similarity patterns that can be exploited by the convolutional network.

3.3 Step 3. Convolutional Model

Convolutional neural networks (CNN) are a popular method for image analysis thanks to their ability to capture spatial invariant patterns. In the proposed method, they play a similar role, but instead of receiving an input image the CNN receives the similarity matrix M. The hypothesis is that it will be able to identify term-similarity patterns that help to determine the relevance of a question-answer pair. Patterns identified by the CNN are sub-sampled by a pooling layer. The output of the pooling layer feeds a fully-connected layer. Finally, the output of the model is generated by a sigmoid unit. This output corresponds to a score, $\mathbf{simScore}(q, a)$, that can be interpreted as a degree of relatedness between the question q and the answer a.

The architecture of the convolutional model is depicted in Figure 2.

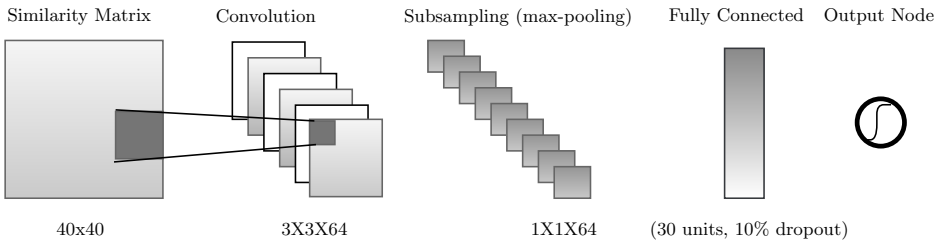

Fig. 2: Convolutional neural network model architecture.

3.4 Step 4 and 5. Two Ranking Stages

During the testing phase, a new query along with candidate answers are presented to the method. The candidate answers $(a_1, a_2, ..., a_k)$ are ranked using the CNN model producing the first rank of them. Based on the premise that the first candidate answer, a^*, is expected to be highly correlated with the question q, a second score, $simScore(a^*, a_k)$, is calculated by comparing each candidate answer with the highest ranked answer. A new ranking is calculated by using a new score corresponding to a linear combination of the first and second score as is shown in Eq. 4.

$$finalScore(q, a_k) = (1 - \alpha) * simScore(q, a_k) + \alpha * simScore(a^*, a_k) \quad (4)$$

As we are introducing a weighting term α to scale the second score, we calculated this term based on the exploration carried out in validation partition, which gives 0.32 as the optimal value.

This strategy promotes candidate answers which share similar terms with the highest ranked answer. This is a strategy analogous to pseudo-relevance feedback in information retrieval [10], where the original query is extended with terms from the highest ranked documents.

4 Experimental Setup

4.1 Test Datasets

The proposed method was compared to baseline and state-of-the-art methods using two information retrieval performance measures Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP). MAP is defined as the mean of the average precision scores for each query over a set of Q queries and MRR evaluates the relative ranks of correct answers in the candidate sentences of a question [15]. To evaluate the method, two standard data sets for the passage retrieval task were used. The baseline and state-of-art methods were evaluated over the same dataset using the same experimental setup.

– **TrecQA**: was provided by Mengqiu Wang and collects data from Text REtrieval Conference (TREC) QA track (8-13). It was first used in [12]. The dataset has two partitions. In TRAIN partition the correctness of answer was carried out manually while in TRAIN-ALL the correctness of candidate answer sentences, was identified by pattern regular expressions matching answer, which induce noise in data, the statics of the related dataset are presented in Table 1.
– **Wiki QA**: is a dataset released in 2015 by Microsoft Research Group [15], that contains Question-Answer pairs for open domain. The Microsoft research group collected Bing Search Engine query logs and extract the questions the user submit from May of 2010 to July of 2011, and the answers are sentences of Wikipedia summary page, Table 2.

Table 1: TrecQA dataset

Split	#Questions	#Pairs
TRAIN ALL	1,229	53,417
TRAIN	94	4,718
DEV	82	1,148
TEST	95	1,517

Table 2: WikiQA dataset

Split	#Questions	#Pairs
TRAIN	2,118	20,358
DEV	296	2,716
TEST	633	6,156

4.2 Baseline Models

Three baseline models were implemented to evaluate the performance of the proposed method. 1) Word Count, which is a word matching method that counts the number of non-stopwords that occur both in the question and in the answer sentences. 2) Weighted Word Count, a modified approach which weights the word counts using semantical information [15]. 3) DeepMind model [16], a semantic parsing method based on similarity metric learning and latent representations.

The list of comparative methods are the following: Word Count, Weighted Word Count, DeepMind model [16], Paragraph Vector (PV) [6], Attention-Based Model (aNMM) [14], Convolutional Neural Network Method (CNN) [11], Pairwise Word Interaction Model (Pairwise CNN) [5], and the proposed model without rerank (This Work) and with rerank (This Work Rerank).

5 Results

Table 3 summarizes the results of all the evaluated methods applied to both TrecQA and WikiQA datasets. In the case of TrecQA two configurations were evaluated: the TRAIN partition and the TRAIN ALL partition, which were described in Subsection 4.1.

In TrecQA dataset, the proposed method presents the best performance of all the evaluated methods. This is consistent in both configurations. Also, we can observe that the use of re-ranking improves the method performance in terms of MAP. The main reason is that in most cases the first ranked answer is relevant we can be evidenced by the high value of the MRR measure.

Table 3: Overview of results QA answer selection task datasets. We also include the results of the base line models. ('-' is Not Reported)

	TREC TRAIN ALL		**TREC TRAIN**		**WikiQA**	
Method	MAP	MRR	MAP	MRR	MAP	MRR
BaseLines						
Word Count	0.6402	0.7021	0.6402	0.7021	0.4891	0.4924
Weighted Word Count	0.6512	0.7223	0.6512	0.7223	0.5099	0.5132
DeepMind model	0.6531	0.6885	0.6689	0.7091	0.5908	0.5951
aNMM [14]	0.7385	0.7995	0.7334	0.8020	-	-
CNN [11]	0.7459	0.8078	0.7329	0.7962	-	-
Pairwise CNN [5]	0.7588	0.8219	-	-	**0.7090**	**0.7234**
PV [6]	-	-	-	-	0.5110	0.5160
This Work	0.7644	**0.8414**	0.7605	0.8344	0.6368	0.6614
This Work (Rerank)	**0.7737**	0.8403	**0.7750**	**0.8350**	0.6351	0.6583

In WikiQA dataset, the best result is obtained by the Pairwise CNN method [5], however the proposed method has a competitive performance that clearly outperforms the other evaluated methods. As evidenced by the overall performance of all the methods, this dataset seems to be more challenging. One problem of this dataset is that it contains several questions without a valid answer in the dataset. The re-ranking strategy produces an important improvement for TrecQA dataset, while it did not improve the performance for the WikiQA dataset. Our conclusion is that the lower MRR in this dataset means that the top ranked answer is less likely to be relevant and thus it has less probability of improving the ranking of relevant answers.

In general, we can say that the proposed method exhibits a very competitive performance when compared to state-of-the-art methods. However, its main strength is the fact that it is simpler than the other methods. This can be objectively measured by counting the number of parameters that the learning algorithm has to adjust during training. Table 4 shows the number of parameters for some of the evaluated methods. Clearly the proposed method has orders of magnitude less parameters than the other methods. This has a positive impact on the amount of computational resources which are required during training and testing.

Table 4: Number of Parameters

Split	# Of Parameters
aNMM [14]	14,000
CNN [11]	100,000
Pairwise CNN (2016)[5]	1.7 million
This Work	**3,198**

6 Conclusions

This work presents a novel method for question-answer ranking based on convolutional neural networks and a pseudo-relevance-feedback-inspired re-ranking strategy. The experimental results shows that the proposed method is competitive when compared to state-of-the-art methods, despite being a simple model with a reduced set of parameters. Taking into account the lack of complexity of the proposed model, the obtained results are very promising.

The experimental evaluation shows an improvement of 2% in the MAP score when the proposed method was applied. Such results suggest that the first ranked answer contains information that can help to rank the subsequent answers.

The future work will focus on exploiting the contextual relationships of terms as well as the inclusion of attention mechanisms that have shown very good results in other text analysis tasks.

References

1. ACL, A.f.C.L..: Question Answering (State of the art) (2007), `http://www.aclweb.org/aclwiki/index.php`, [Online; accessed 1-May-2017]
2. Bird, S.: Nltk: the natural language toolkit. In: Proceedings of the COLING/ACL on Interactive presentation sessions. vol. 1, pp. 69–72. ACL (2006)
3. Dong, L., Wei, F., Zhou, M., Xu, K.: Question answering over freebase with multi-column convolutional neural networks. In: ACL. vol. 1, pp. 260–269 (2015)
4. Etzioni, O.: Search needs a shake-up. Nature 476(7358), 25–26 (2011)
5. He, H., Lin, J.J.: Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In: HLT-NAACL. vol. 1, pp. 937–948 (2016)
6. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. vol. 14, pp. 1188–1196 (2014)
7. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
8. Liu, F., Pennell, D., Liu, F., Liu, Y.: Unsupervised approaches for automatic keyword extraction using meeting transcripts. In: Proceedings of human language technologies. vol. 1, pp. 620–628. ACL (2009)
9. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space (2013)
10. Riezler, S., Vasserman, A., Tsochantaridis, I., Mittal, V., Liu, Y.: Statistical machine translation for query expansion in answer retrieval. In: ACL (2007)
11. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: 38th ACM SIGIR (2015)
12. Wang, M., Smith, N.A., Mitamura, T.: What is the jeopardy model? a quasi-synchronous grammar for qa. 7, 22–32 (2007)
13. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: 32nd Proceedings ACL. vol. 1, pp. 133–138. Association for Computational Linguistics (1994)
14. Yang, L., Ai, e.a.: anmm: Ranking short answer texts with attention-based neural matching model. In: Proceedings of the 25th ACM ICIKM. ACM (2016)
15. Yang, Y., Yih, W.t., Meek, C.: Wikiqa: A challenge dataset for open-domain question answering. In: EMNLP. vol. 1, pp. 2013–2018 (2015)
16. Yu, L., Hermann, K.M., Blunsom, P., Pulman, S.: Deep learning for answer sentence selection. NIPS deep learning workshop (2014)