A Shallow Convolutional Neural Network Architecture 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. This paper addresses the problem of answering a question by choosing the best answer from a set of candidate text fragments. This task requires to identify and measure the semantical relationship between the question and the candidate answers. Unlike previous solutions to this problem based on deep neural networks with million of parameters, we present a novel convolutional neural network approach that despite having a simple architecture is able to capture the semantical relationships between terms in a generated similarity matrix. The method was systematically evaluated over two different standard data sets. The results show that our approach is competitive with state-of-the-art methods despite having a simpler and efficient architecture.

Keywords: question answering, information retrieval, passage retrieval, answer sentence selection, answer ranking

1 Introduction

Next generation web search engines must understand user's information needs formulated in plain language, look for documents where the information resides and select an appropriate answer. The task of returning exact answers to natural language questions issued by users is a challenging task and takes an important place in the needs of user information access [4].

In general, the type of questions that users can pose define the approach that the system will use, for this reason, it is important to mention the type of questions that can be faced in question answering tasks. The following are two examples of popular question types:

- **Factoid Question:** Refers to the type of questions that drives to obtain an exact answer (fact) for a specific kind of question. For example, "Where Salma Hayek was born?", "In what month is temperature a maximum in New York?".

– **Complex Questions:** This category corresponds to questions that can not be answered by a simple fact. This includes definitional questions, such as "Who is Bill Gates?"; entity definition questions, such as "What is RNA?"; list questions, such as "What are the bones of the human body".

Addressing the problem of automatic question answering involves different subproblems: question processing, document retrieval, passage retrieval, and answer extractions are the main ones [9]. In this paper, we address the problem of passage retrieval. This task takes as input a question the set of sentences {s1 , s2 , . . . sn }, and identify the sentence that is most related to the question.

The paper presents a novel method for passage retrieval based on neural networks. The neural network learn to predict the degree of relatedness of a question and a candidate answer pair by finding semantical relationships between question's and answer's terms. The model uses a small convolutional architecture to capture the most important correlations and return a matching probability that is used to rank the candidate answers, the simplicity of the model is represented in a low number of parameters (3,197) compared with the million of parameters of other machine learning methods for the same task [7, 6].

The proposed method was validated using two standard public datasets: TrecQA Dataset [17] and WikiQA [21]. The 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).

The paper is organized as follows. Section 2 presents the related work done in passage retrieval subtask and show some of the particularities and limitations of them. Section 3 shows the proposed architecture and the design decisions used in the approach. In Section 4 the experimental setup is detailed including the evaluation metrics and the dataset where the method is validated. Section 5 shows the results achieved by the method in the proposed datasets. Finally, we conclude our paper and discuss future work in Section 6.

2 Related Work

Most of the relevant methods in passage retrieval are ranked by Association for Computational Linguistics [1] using TrecQA Dataset [17]. In this rank, there are statistical models, deep learning models, pattern-based models and pure linguistic approach methods. We are going to check some of the most interesting methods and discuss its particularities.

Using the idea that questions can be generated based on correct answers, they propose a transformation model that send the question and answer to the same space and then compare how close they are [23], this method was replicated in this work and was used to compare the performance of out proposed model.

Heilman and Smith [8] represent the semantical relatedness of question-answer pairs, as a tree edit distance model and employed a tree kernel as a heuristic in a greedy search routine to extract the features from the sequences. Those sequences are then fed into a logistic regression as features to be classified as related or not.

Wang and Manning [18] proposed a probabilistic method that models tree-edit operations based on Conditional Random Fields. They treat alignments as structured latent variables that are learned as state transitions which allow incorporating a diverse set of arbitrarily overlapping features.

More recently, Yao et al. [22] based its method on Heilman and Smith's tree edit distance approach [8] but using dynamic programming to find the optimal sequences, also they take advantage of some WordNet semantic features to improve the performance.

One deep learning method was proposed by He and Lin [7] and was based on Bidirectional Long Short-Term Memory Networks that can capture the relation between sequences of terms of two sentences read in both directions (left-right and right-left). This new approach captures the interactions between terms and proposes a similarity focus layer that highlights the most relevant relations based on the weights related to each pair of terms.

Another deep learning approach was proposed by Yang et al. [20], they introduce an attention neural matching model for ranking short answer text. They adopt a value shared weighting scheme for incorporate term importance learning and in this way make the relevant terms more important than the others.

The most recent work of Rao, He and Lin [13] make use of Noise-Contrastive Estimation (NCE) modeling. Where the deep neural model should learn differentiable positive and negative samples in order to improve the ability of discriminate a good sample from its neighboring bad samples.

Nearly all of the recent works in this field make use of deep neural networks architectures, which brings an important generalization ability but with a huge number of parameters to be found. Our approach is relatively shallow compared with state of the art methods, but the performance is still competitive.

3 Model Description

The main assumption of the model presented in this paper is that the question and the answer are semantically related term by term, it means that the correlation between the question and the answer is given by the correlation between their component terms.

Figure 1 shows the overall architecture of the method. The method starts by generating a similarity matrix M, where the entry $M_{i,j}$ represents the semantic similarity of the i-th question term and the j-th answer term. The generated similarity matrix is weighted by a salience matrix that takes into account the syntactical function of terms.

A convolutional layer is applied over the weighted similarity matrix. Patterns identified by the convolutions 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 may be interpreted as a degree of relatedness between the query and the answer.

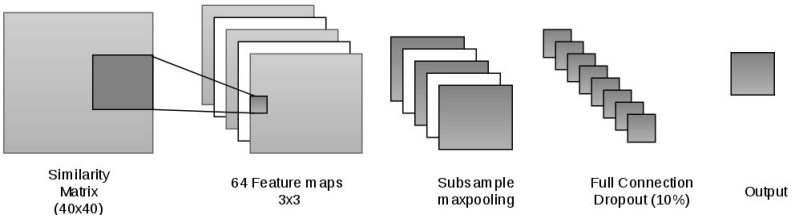

Fig. 1: Convolutional neural network model architecture.

In order to calculate the similarity matrix, we look for semantical relations term by term. We obtain the word2vec [12] vector representation for both terms (q_i, a_j) and measure their cosine similarity using the following equation.

$$scos(q_i, a_j) = 0.5 + \frac{q_i \cdot a_j}{2 \left\|q_i\right\|_2 \left\|a_j\right\|_2} \tag{1}$$

As not all terms are equally informative, we most weight the most salient terms in both question and answer. Based on the work done by Liu et al, and Dong et al. we give more importance to verb, noun and adjective terms [11, 3].

This task is done by extracting the part-of-speech tagging information and if both terms are important the score is 1, if just one is important the score is 0.6 or 0.3 if none of them is important.

This yields the following salience function:

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

Where $imp(t_k)$ is a indicator functions that returns 1 if the term is a verb, noun or adjective and 0 otherwise. Finally, the weighted similarity matrix is calculated as Equation 3 shows.

$$M_{i,j} = scos(q_i, a_j) \circ sal(q_i, a_j), \tag{3}$$

where $\circ$ indicates the element-wise product of the matrices.

4 Experimental evaluation

The goal of this section is to evaluate the performance of the model in two public question answering datasets. The experimental setup, as well as the results, are discussed in the following subsections.

4.1 Experimental setup

The implementation was carried on with Keras and NLTK libraries. The code is available at Github [3].

Test datasets

- **TrecQA** was provided by Mengqiu Wang and collects data from the Text Retrieval Conference (TREC) QA track (8-13). It was first used in [19]. The dataset contains question and candidate answers selected from each question's document pool. The dataset has two partitions: TRAIN y TRAIN-ALL. In TRAIN partition the correctness of each answer was label manually while in TRAIN-ALL the correctness of candidate answer sentences was identified by regular expressions matching the answer. The last partition is much larger in a number of questions but noisier, the statistics of the referred dataset is in Table 1.
- **Wiki QA** is an open domain dataset released in 2015 by Microsoft research Group [21], that contains question-answer pairs. 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.
  As a quality indicator, the queries that were submitted by at least 5 users and that have a corresponding Wikipedia page were selected and introduced in the dataset.
  The answers were collected using the Wikipedia page related to the query and using every sentence of the summary section as candidate answer. The statistics of the dataset are presented in 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

Preprocessing The text of both questions and answers was processed using the following steps:

- Tokenization: Delimits the terms using the space and punctuations symbols.
- Lowercase: standardizes the terms lowercasing them. This is clearly important for the POS tagger which tends to tag most words that have a capital letter as a proper noun.
- POS tagging: NLTK pos-tagger is used to extract syntactical information [2].

[3] GitHub passage retrieval code `https://github.com/andresrosso/passage_retrieval`

– Word2vec Vector Representation: We use word2vec with the 300-dimensional GoogleNews trained model to represent each word [12].

It is important to mention that using stop word removal in question or answer terms can cause the loss of important information, as for example wh-words, verbs as "to be", etc. To avoid these extreme cases, we did not filter stop words.

Due to the variation in the sentence length, we had to limit the number of terms to calculate the similarity. The number of maximum terms was calculated adding one standard deviation (σ) to the mean (μ) of the number of terms.

$$max\_terms = \mu + \sigma \tag{4}$$

So, for TrecQA the number of terms found was 36, and for WikiQA was 34. We use a maximum sentence length of 40 terms with zero padding.

Model parameters The model hyper-parameters were tunned using hyper-parameter exploration with cross-validation. The parameters chosen are listed next.

– **Convolution Parameters:** The number of convolutional filters used are 64, width 3 and length 3, the stride used is 1 without padding.
– **Convolution Activation Function:** After a convolutional layer, it is a convention to apply a nonlinear layer [5]. We choose a ReLU layer since it has been reported to have better performance in training without losing accuracy [16].
– **Pooling Layers:** For the pooling layer, we used max pooling.
– **Dropout Layer:** We add a dropout layer to help alleviate the overfitting problem [15]. The percentage of dropout was set to 10% since it produced the best results in cross-validation.

Sampling strategy WikiQA and TrecQA dataset are very unbalanced. Most of the question and answer pairs are not related (less than 10 percent are related). This may cause that the model is biased towards negative predictions. Therefore, it is essential to balance the dataset and we have alleviated the problem choosing randomly the same number of positive and negative samples in training and validation stage.

Model training The model training was done using rmsprop optimization algorithm with 256 samples in mini-batch. The number of maximum epochs was defined to 500, but as regularization strategy, we use early stopping with a patience value of 20 epochs, where the mean average precision (MAP) calculated over validation partition must be improved otherwise the learning is stopped.

Evaluation metrics We report two standard evaluation measures commonly used in IR and QA research: mean average precision (MAP) and mean reciprocal rank (MRR). All results are produced using the standard trec-eval tool. To calculate this performance measures we must define the metrics on which MAP and MRR are based on:

- **Precision**: Is calculated as the fraction of the documents retrieved, that are relevant based on the query posed.

$$P = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \tag{5}$$

- **Average Precision**: This metric combine the precision and the recall metric, to give some importance to the order of retrieved documents.

$$AveP = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{\text{number of relevant documents}} \tag{6}$$

$$rel(k), \text{an indicator function equaling}$$

$$1, \text{if the item at } rank_k \text{ is a relevant document}$$

$$0, \text{otherwise}$$

- **Mean Average Precision**: Is the mean of the average precision scores for each query, over a set of Q querys.

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{7}$$

- **Mean Reciprocal Rank** : Is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{8}$$

  $rank_i$, is the position of the first relevant document returned for the i query.

Baseline models We compare the performance of our method against three baselines models and some of the most remarkable methods in the state-of-the-art.

The baselines models implemented are Word Count, Weighted Word Count and a model proposed by Google Deepmind research team. The first method is based on counting the number of common non-stop-words that occur in the question and in the answer sentence, the second is a variation that re-weights

the counts using semantical information [21]. The last one is a semantic parsing method that learns a distributed representation of the data [23].

Some state-of-the-art methods are also compared against our approach. Yang et al. [20] present an attention-based model that incorporates the term importance learned (aNMM). Severyn et al. [14] presents a deep convolutional method for ranking pairs of short texts, the objective is to learn an optimal representation of text pairs (CNN). He et al. presents a deep pairwise word interaction model for Semantic Similarity Measurement (Pairwise CNN) 2016 [7].

4.2 Results and discussion

Table 3 summarizes the results for the TrecQA dataset and WikiQA dataset. In the case of TrecQA two configurations were evaluated: the TRAIN partition and the TRAIN ALL partition, which were described in Subsection 4.1.

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 [20]	0.7385	0.7995	0.7334	0.8020	-	-
CNN [14]	0.7459	0.8078	0.7329	0.7962	-	-
Pairwise CNN [7]	**0.7588**	**0.8219**	-	-	**0.7090**	**0.7234**
PV [10]	-	-	-	-	0.5110	0.5160
This Work	0.7064	0.7947	**0.7367**	**0.8215**	0.6062	0.6171

In TrecQA the results reveal that our approach overcomes the baseline models and is very competitive with state-of-the-art methods. The proposed model performs better when the training is carried out with TRAIN partition, this could be explained because in TRAIN ALL the question-answer pairs are more noisy as was explained in section 4.1.

The Mean Reciprocal Rank (MRR) score in TREC dataset is high, which is an evidence that our approach returns the correct answer in the first two positions.

In the case of the WikiQA dataset, the method with the highest score is Pairwise CNN [7], however the proposed method is still competitive with the baselines models. Compared with the rest of the state-of-the-art methods, the obtained scores are quite close to DeepMind model [23] and overcomes the result reported by Le and Mikolov [10] in distributed vector model.

The results in WikiQA dataset are not as good as in TREC dataset, the reason could be the small number of samples and the large number of negative samples in the last dataset.

The proposed model is simpler than state-of-the-art models [6, 7]. In the proposed approach the number of parameters are in order of thousands while in state-of-the-art approaches are in order of millions as is reported in Table 4. A less complex computational model makes easier and faster training and evaluation stages.

Table 4: Number of Parameters

Split	# Of Parameters
Multi-Perspective CNN (2015) [6]	10.0 million
Pairwise CNN (2016)[7]	1.7 million
This Work	**3,197**

5 Conclusion

This paper presents a low parametrized convolutional neural network for answer selection task, the method was tested in two of the most used question answering datasets: WikiQA and TrecQA. The method is based on a convolutional neural network to identify similarity patterns that are present in a calculated weighted cosine similarity matrix. Despite the simplicity of the architecture, the results show that our method is competitive with state-of-the-art methods. Trained with the trec 'TRAIN' partition is able to overcome the Pairwise CNN method [7], achieving the best results. The reduced number of parameters is an advantage of the proposed model which makes the training process computationally less costly.

As future work, we plan to explore the use of other linguistic features that can provide additional semantical relations between terms. The high mean reciprocal rank score obtained, reveals that the convolutional neural network is highly accurate identifying the most relevant candidate answer. This information can be used to make a re-ranking based on those relations in order to gain semantical information.

References

1. Association for Computational Linguistics — ACL. Question Answering (State of the art), 2007.
2. Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

3. Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *ACL (1)*, pages 260–269, 2015.
4. Oren Etzioni. Search needs a shake-up. *Nature*, 476(7358):25–26, 2011.
5. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.
6. Hua He, Kevin Gimpel, and Jimmy J Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*, pages 1576–1586, 2015.
7. Hua He and Jimmy Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of NAACL-HLT*, pages 937–948, 2016.
8. Michael Heilman and Noah A Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies, ACL, pages=1011–1019, year=2010, organization=Association for Computational Linguistics*.
9. Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *natural language engineering*, 7(04):275–300, 2001.
10. Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.
11. Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of human language technologies, ACL*, pages 620–628. Association for Computational Linguistics, 2009.
12. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. word2vec, 2014.
13. Jinfeng Rao, Hua He, and Jimmy Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM*, pages 1913–1916. ACM, 2016.
14. Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings ACM SIGIR Conference, pages=373–382, year=2015, organization=ACM*.
15. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
16. Sachin S. Talathi and Aniket Vartak. Improving performance of recurrent neural network with relu nonlinearity. *CoRR*, abs/1511.03771, 2015.
17. C Wang, a Kalyanpur, and B K Boguraev. Relation extraction and scoring in DeepQA. *IBM Journal of Research and Development*, 56(3):9:1–9:12, 2012.
18. Mengqiu Wang and Christopher D Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *ACL Proceedings*, pages 1164–1172. Association for Computational Linguistics, 2010.
19. Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. (June):22–32, 2007.
20. Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings ACM*, pages 287–296. ACM, 2016.
21. Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018. Citeseer, 2015.
22. Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer, 2013.
23. Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *NIPS deep learning workshop*, 2014.