A Neural Network-based Framework for Non-factoid Question Answering

Nam Khanh Tran, Claudia Niederée
L3S Research Center, Leibniz Universität Hannover
Hannover, Germany
{ntran,niederée}@L3S.de

ABSTRACT

In this paper, we present a neural network based framework for answering non-factoid questions. The framework consists of two main components: *Answer Retriever* and *Answer Ranker*. In the first component, we leverage off-the-shelf retrieval models (e.g. bm25) to retrieve a pool of candidate answers regarding to the input question. Answer Ranker is then used to select the most suitable answer. In this work, we adopt two typical deep learning based frameworks for our Answer Ranker component. One is based on Siamese architecture and the other is the Compare-Aggregate framework. The Answer Ranker component is evaluated separately based on popular answer selection datasets. Our overall system is evaluated using FiQA dataset, a newly released dataset for financial domain and shows promising results.

KEYWORDS

Non-factoid question answering, representation learning

ACM Reference Format:

Nam Khanh Tran, Claudia Niederée. 2018. A Neural Network-based Framework for Non-factoid Question Answering. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23-27, 2018, Lyon, France*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3184558.3191830>

1 INTRODUCTION

Question answering (QA) is an important end-user task at the intersection of natural language processing (NLP) and information retrieval (IR). QA itself can be divided into factoid QA, which enables the retrieval of facts, and non-factoid QA, which enables finding of complex answer texts such as descriptions, opinions, or explanations. Table 1 shows an example of a question with a ground-truth answer from financial domain. In this paper, we give more attention to non-factoid QA and propose a two-step framework to tackle this problem: i) *Answer Retriever* - question analysis and retrieval of candidate answers; ii) *Answer Ranker* - ranking and selecting of the most suitable answer. While the first step can be accomplished by using off-the-shelf retrieval models (e.g. tf-idf, bm25), the second step brings more challenges.

One main challenge of the second task lies in the complex and versatile semantic relations that can be observed between questions and answers. In non-factoid QA, different from many other

Table 1: An example of a question with an answer from the FiQA dataset.

Question: Having a separate bank account for business/investing, but not a business account?

Ground-gruth answer: Having a separate checking account for the business makes sense. It simplifies documenting your income/expenses. You can explain every dollar entering and exiting the account without having to remember that some of them were for non-business items... I don't see the need for a separate checking account for investing. The money can be kept in a separate savings account that has no fees, and can even earn a little interest....

matching tasks, the linguistic similarities between questions and answers may or may not be indicative for the good answers; depending on what the question is looking for, a good answer may come in different forms. Sometimes a correct answer completes the question precisely with the missing information, and in other scenarios, good answers need to elaborate part of the question to rationalize it. In other cases, the best answers can also be noisy and include extraneous information irrelevant to the question. In addition, while a good answer must relate to the question, they might not share common lexical units.

Recently, deep learning based approaches are proposed to tackle these challenges and have shown competitive results [5, 9, 12]. In principle, they are based on Siamese architecture [4] and Compare-Aggregate framework [11]. In the first architecture, the same neural network encoder (e.g., a CNN or a LSTM) is applied to question and answer sequences individually, so that both of the two sequences are encoded into latent vectors in the same embedding space. Then, a matching decision is made solely based on the two feature vectors [5, 9]. Under the second architecture, smaller units (such as words or contextual vectors) of the question and answer sequences are firstly matched, and then the matching results are aggregated (by a CNN or a LSTM) into a vector to make the final decision [11, 12].

In this work, we adopt these architectures to our Answer Ranker component and evaluate their performances on two different non-factoid answer selection datasets, i.e. InsuranceQA and FiQA. To evaluate our overall system as well as participate in WWW 2018 Challenge¹, we report the system performance on the FiQA dataset.

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '18 Companion, April 23-27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5640-4/18/04.

<https://doi.org/10.1145/3184558.3191830>

¹<https://sites.google.com/view/fiqa/home>

2 OUR SYSTEM

In the following, we describe our system for non-factoid QA which consists of two components as shown in Figure 1: (1) Answer Retriever module for finding candidate answers and (2) an answer selection model, Answer Ranker, for selecting the most suitable answer.

2.1 Answer Retriever

Following classical QA systems, we use an efficient (non-machine learning) answer retrieval system to first narrow our search space and focus on ranking only answers that are likely to be relevant. A simple inverted index lookup followed by term vector model scoring performs quite well on this task. In particular, our Answer Retriever is based on popular information retrieval models, i.e., *query-likelihood language model* and *bm25*.

2.2 Answer Ranker

In this paper, we pay more attention to the second component that enables choosing suitable answer from a list of candidate answers in regard to the input question. Our Answer Ranker model is inspired by the recent success of neural network models on this task. Typically, previous works such as [5, 9, 12] are based on two types of deep learning frameworks. One is the Siamese architecture [4] and the other is the Compare-Aggregate framework [11]. We adopt both types of frameworks to our Answer Ranker creating SRanker and CARanker as described below. For both answer rankers, given a question q consisting of m tokens and a candidate answer a of l tokens, the word embeddings (WEs) of both q and a are first retrieved, that are sequences of vectors $q = \{q_1, \dots, q_m\}$ and $a = \{a_1, \dots, a_l\}$ where each q_i (a_j) is a d -dimensional vector.

SRanker SRanker is based on the siamese architecture [4]. Firstly, a bidirectional LSTM is applied separately over the two sequences of WEs creating hidden vectors for the question and answer, i.e. $h_q(t) = \text{LSTM}(\vec{h}_q(t-1), q_t) \parallel \text{LSTM}(\overleftarrow{h}_q(t+1), q_t)$ and $h_a(t) = \text{LSTM}(\vec{h}_a(t-1), a_t) \parallel \text{LSTM}(\overleftarrow{h}_a(t+1), a_t)$.

Let o_q denote the final representation of the question by taking the last hidden vector $h_q(m)$ or max/mean pooling over all hidden vectors. Let $H_a = \{h_a(1), h_a(2), \dots, h_a(l)\}$ denote the hidden states when modeling the answer. Instead of using the last hidden state or average or max over all hidden states, an attention mechanism is applied:

$$\begin{aligned} \alpha_t &\propto f_{\text{attention}}(o_q, h_a(t)) \\ o_a &= \sum_t \alpha_t h_a(t) \end{aligned} \quad (1)$$

where $h_a(t)$ is hidden state of the answer at time t . Basically, the attention mechanism gives more weight to words that have more influence on the resulting representation. In SRanker, we adopt the two most commonly used implementations of the attention function: *MLP Attention* [9] and *Bilinear Attention* [3].

MLP Attention: In [9], Tan et al. proposed to compute $f_{\text{attention}}$ by a multi-layer perceptron as follows:

$$\begin{aligned} m(t) &= \tanh(W_a h_a(t) + W_q o_q) \\ f_{\text{attention}} &= \text{softmax}(w^T m(t)) \end{aligned} \quad (2)$$

where W_a, W_q are attentive weight matrices and w is an attentive weight vector.

Bilinear Attention: In [3], Chen et al. presented another attention model which uses a bilinear term instead of a tanh layer, and showed its effectiveness in machine reading comprehension task. We adapt this model to estimate the attention function $f_{\text{attention}}$:

$$f_{\text{attention}} = \text{softmax}_t(o_q^T W_s h_a(t)) \quad (3)$$

where W_s is the attention parameter.

In SRanker, we adopt a *pairwise* ranking method to define our objective function. First, a cosine similarity between two vectors o_q and o_a is used to score the input pair, i.e. $\text{sim}(q, a)$ and the hinge loss training objective is defined as follows:

$$L = \max\{0, \lambda - \text{sim}(q, a_+) + \text{sim}(q, a_-)\} \quad (4)$$

where a_+ is a ground truth answer, a_- is an incorrect answer randomly chosen from the entire answer space, and λ is a margin. In non-factoid QA datasets, we treat any question with more than one ground truth as multiple training examples. During training, for each question we randomly sample K negative answers, but only use the one with the highest L to update the model.

CARanker CARanker is based on Compare-Aggregate model [11], which is illustrated in Figure 3. In general, the model can be divided into four layers:

Word Representation Layer: Similar to [11], we preprocess questions and answers as follows:

$$\begin{aligned} \bar{q}_i &= \sigma(W_1 q_i + b_1) \odot \tanh(W_2 q_i + b_2) \\ \bar{a}_j &= \sigma(W_1 a_j + b_1) \odot \tanh(W_2 a_j + b_2) \end{aligned} \quad (5)$$

where W_1, W_2 and b_1, b_2 are network parameters.

Attention Layer: As shown in Figure 3, in this layer we compute attention matrix $E = \{e_{ij}\}$ of question vector $\bar{q}_i$ and answer vector $\bar{a}_j$ and obtain soft aligned sub-phrase $H = \{h_j^a\}$ as follows:

$$\begin{aligned} e_{ij} &= \bar{q}_i * \bar{a}_j \\ h_j^a &= \sum_{i=1}^m \text{softmax}(e_{ij}) \bar{q}_i \end{aligned} \quad (6)$$

where h_j^a is the corresponding vector in $\bar{q}$ that is aligned to $\bar{a}_j$.

Comparison Layer: The goal of this comparison layer is to match each $\bar{a}_j$ (the j^{th} word and its context in $\bar{a}$) with h_j^a (a weighted version of $\bar{q}$ that best matches $\bar{a}_j$). Here we adopt a comparison function as follows:

$$t_j^a = \bar{a}_j \odot h_j^a \quad (7)$$

where $\odot$ is element-wise multiplication.

Aggregation Layer: Finally, we aggregate the vectors t_j^a using one-layer CNN [7]:

$$\begin{aligned} r_a &= \text{CNN}([t_1^a, t_2^a, \dots, t_l^a]) \\ s_a &= \tanh(W_s r_a + b_s) \end{aligned} \quad (8)$$

where W_s and b_s are the weights of prediction layer. The final score s_a is then used to rank candidate answers.

As suggested by Bian et al. [2], we adopt a *listwise* ranking method for our CARanker. Concretely, we feed a question Q and a set of candidate answers $A = \{A_1, A_2, \dots, A_n\}$ and target label set

Figure 1: An overview of our system for non-factoid question answering

Figure 2: Overview of SRanker

Figure 3: Overview of CARanker. $\oplus$ represents attention operation, and $\otimes$ represents element-wise multiplication

$Y = \{y_1, y_2, \dots, y_n\}$ into the model, then get the normalized score vector S :

$$S = \text{softmax}([s_{A_1}, s_{A_2}, \dots, s_{A_n}]) \quad (9)$$

Target labels also need to be normalized by $Y = \frac{Y}{\sum_{i=1}^n y_i}$. Lastly, the KL-divergence loss is used to train our CARanker:

$$L = \frac{1}{n} \sum_{i=1}^n KL(S||Y) \quad (10)$$

3 EXPERIMENTAL SETUP

3.1 Dataset

First, to evaluate our Answer Rankers we conduct an empirical evaluation based on two different answer selection datasets: InsuranceQA and FiQA. These datasets contain text of different domains and exhibit different characteristics:

- **InsuranceQA** - This is a recently released large-scale non-factoid QA dataset from the insurance domain created by Feng et al. [6]. In this work we use the first version of the dataset. The dataset is already divided into a training set, a validation set, and two test sets, in which a question may have multiple correct answers and normally the questions are much shorter than the answers. The average length of questions and answers in tokens are 7 and 95, respectively. For each question in the development and test sets, there is a set of 500 candidate answers, which include the ground-truth answers and randomly selected negative answers. More details can be found in [6].
- **FiQA** - This dataset has been recently released for WWW 2018 Challenge. The dataset is built by crawling Stackexchange, Reddit and StockTwits in which part of the questions are opinionated, targeting mined opinions and their respective entities, aspects, sentiment polarity and opinion holder. We minimally preprocess the data only performing tokenization and lowercasing all words. To reduce the size of resulting vocabulary, we remove all rare words which occur less than 5 times. The length of the vocabulary $|V|=22413$ and the average length of questions and answers in tokens are 11 and 135, respectively. We split the dataset into training, development and test sets which consist of 5999, 323 and 324 questions, respectively. To evaluate our Answer Rankers,

Table 2: The statistic of the FiQA dataset

Dataset	Train	Dev	Test	Avg.Q	Avg.A	#CAs
InsuranceQA	12887	1000	1800x2	7	95	500
FiQA	5999	323	324	11	135	500

for each question in the development and test sets, we construct the answer pools by including the correct answer(s) and randomly selected candidates from the complete set of unique answers. Finally we have 500 candidate answers for each question.

Table 2 presents some statistics about the datasets, including the number of questions in each set, average length of questions and answers as well as average number of candidate answers in the development and test sets.

To evaluate our overall system, we utilize the same set of questions in the development and test sets in FiQA dataset. Unlike Answer Ranker evaluation, these questions are first passed through Answer Retriever to retrieve a pool of candidate answers. Then, the Answer Ranker, which is trained on the FiQA answer selection dataset, is applied to select the positive answers.

3.2 Implementation Details

Our Answer Retriever is implemented based on Apache Lucene.² We tried *bm25* and *query-likelihood language model (LM)* as our retrieval models, and finally use *bm25* since it yields slightly better results. For each input question, we used the top 50 and top 100 answers returned by *bm25* model as our candidate answers for the answer selection step.

Our Answer Rankers are trained as follows. The model parameters are optimized using Adam optimizer [1] with a learning rate of 0.001. The parameters are regularized with a per-minibatch L2 regularization strength of 10^{-5} and a dropout of $d = 0.2$ is also applied to prevent overfitting. We initialized the word embeddings with 300-dimensional Glove vectors [8] trained on 840 billion words. Embeddings for words not present in the Glove vectors are randomly initialized with each component sampled from the uniform distribution over $[-0.25, 0.25]$.

SRanker A batch size is tuned amongst {50, 100, 150} and finally set to 100. The hidden size of LSTM layer is set to 512 for FiQA and 141 for InsuranceQA. We tried different margins λ in the hinge loss function and finally fixed the margin to $\lambda = 0.2$. A number of negative answers $K = 50$ is used during training. The word embeddings are also part of the parameters and are optimized during training.

CARanker A batch size is tuned amongst {30, 50, 100} and finally set to 30. The dimensionality of the hidden layers is set to 300 for FiQA and 282 for InsuranceQA. For each positive answer, the number of negative answers is tuned amongst {5, 10, 20, 50} which are randomly chosen from the answer set. The word embeddings are not updated during training in CARanker.

Since sequences within a mini-batch have different lengths, we use a mask matrix to indicate the real length of each sequence. We

²<https://lucene.apache.org/>

Table 3: Experimental results of Answer Rankers on InsuranceQA. The first group reports the results of several baselines on InsuranceQA. The results of Siamese-LSTM are reported in the second group. We report the performance of SRanker and CARanker in the last group.

Model	Test1		Test2	
	MRR	P@1	MRR	P@1
CNN (Feng et al. [6])	–	0.628	–	0.592
CNN-GESD (Feng et al. [6])	–	0.653	–	0.610
AP-LSTM (Tan et al. [9])	–	0.690	–	0.648
IARNN-Gate (Wang et al [10])	–	0.701	–	0.628
Siamese-LSTM	0.743	0.643	0.719	0.617
SRanker _{mlp}	0.790	0.693	0.762	0.648
SRanker _{bilinear}	0.781	0.689	0.755	0.658
CARanker	0.783	0.686	0.756	0.652

Table 4: Experimental results of Answer Rankers on FiQA. The results of Siamese-LSTM are reported in the first group. The results of SRanker and CARanker are shown in the second group.

Model	NDCG	MRR	P@1
Siamese-LSTM	0.509	0.566	0.469
SRanker _{mlp}	0.562	0.616	0.509
SRanker _{bilinear}	0.558	0.606	0.506
CARanker	0.605	0.653	0.540

trained all models for a maximum of 20 epochs. In this work, we use Mean Reciprocal Rank (MRR), Precision@1 (P@1) and Normalized Discounted Cumulative Gain (NDCG) at top 10 as our evaluation metrics. We take MRR scores on the development set at every epoch and save the parameters of the network for the top three models. We report the best test score from the saved models. All models were built on a Linux machine with Nvidia GTX Ti 1080 GPU (12GB RAM). The code to reproduce the reported results are publicly available at <https://github.com/namkhanhtran/nn4nqa>.

4 EXPERIMENTAL RESULTS

In this section, we first present the results of our Answer Rankers on InsuranceQA and FiQA datasets, and then describe overall performance of our system on the FiQA dataset.

4.1 Answer Ranker

The results of our Answer Ranker models are shown in Table 3 and Table 4. Overall, our models perform better than the simple Siamese-LSTM model. This is due to the fact that answers in non-factoid QA are often noisy and include extraneous information irrelevant to the question, which cannot be handled by the simple Siamese-LSTM model. This confirms the conclusion from previous works [9, 10] where attention-based models outperform the basic Siamese matching model. In addition, SRanker and CARanker obtain comparable

Table 5: Overall performance of our system on FiQA. The top 100 candidate answers returned by Answer Retriever are considered.

Model	NDCG	MRR	P@1
SRanker _{mlp}	0.248	0.232	0.115
SRanker _{bilinear}	0.252	0.242	0.138
CARanker	0.263	0.243	0.127

Table 6: Overall performance of our system on FiQA. The top 50 candidate answers from Answer Retriever are taken.

Models	NDCG	MRR	P@1
SRanker _{mlp}	0.278	0.242	0.119
SRanker _{bilinear}	0.297	0.268	0.153
CARanker	0.308	0.279	0.157

results on InsuranceQA while CARanker outperforms SRanker on FiQA. Table 3 also shows that our Answer Ranker models obtain similar performances compared to previous baselines.

4.2 Full Non-factoid Question Answering

Table 5 and Table 6 show the overall performance of our system for answering non-factoid questions from FiQA dataset. The results show that by using top 50 candidate answers from Answer Retriever, our system performs slightly better than using top 100 candidate answers. In addition, SRanker_{bilinear} and CARanker show comparable performances and both outperform SRanker_{mlp}. One possible explanation is that CARanker and SRanker_{bilinear} directly model the interaction between questions and answers via multiplication, while SRanker_{mlp} indirectly models this interaction via a tanh layer. In addition, it can be observed that the results here are lower than in Table 4. This is due to the fact that correct answers might not be included in the candidate answers returned from Answer Retriever (bm25). More advanced approaches could be used to improve the answer retrieval step performance.

5 CONCLUSIONS

We have presented a neural network based framework for tackling non-factoid question answering. Our framework contains Answer Retriever which retrieves a pool of candidate answers and Answer Ranker which selects the most relevant answer from the answer pool. For Answer Ranker, we adopted recent deep learning based approaches which aim to learn low-dimensional vectors for questions and answers, perform matching and probably aggregate matching scores for final answer ranking. Experimental results show that our Answer Rankers obtain comparable performances to start-of-the-art approaches and overall our system achieves promising results.

ACKNOWLEDGMENTS

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) for the project eLabour (01UG1512C).

REFERENCES

- [1] Jimmy Ba and Diederik Kingma. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference of Learning Representations*.
- [2] Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A Compare-Aggregate Model with Dynamic-Clip Attention for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2358–2367.
- [3] Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2358–2367.
- [4] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. 539–546.
- [5] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. In *CoRR, abs/1602.03609*.
- [6] Minwei Feng, Bing Xiang, Michael R. Glass, Lidian Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Workshop on Automatic Speech Recognition and Understanding*. 813–820.
- [7] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1746–1751.
- [8] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.
- [9] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 464–473.
- [10] Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1288–1297.
- [11] Shuohang Wang and Jing Jiang. 2017. A Compare-Aggregate Model for Matching Text Sequences. In *Proceedings of 5th the International Conference on Learning Representations*.
- [12] Radu Florian Zhiguo Wang, Wael Hamza. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 4144–4150.