Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko and **Jaewoo Kang***
Department of Computer Science and Engineering, Korea University
{jinhyuk_lee, ysj5419, hyunjae-kim}@korea.ac.kr
{gomi1503, kangj}@korea.ac.kr

Abstract

Recently, open-domain question answering (QA) has been combined with machine comprehension models to find answers in a large knowledge source. As open-domain QA requires retrieving relevant documents from text corpora to answer questions, its performance largely depends on the performance of document retrievers. However, since traditional information retrieval systems are not effective in obtaining documents with a high probability of containing answers, they lower the performance of QA systems. Simply extracting more documents increases the number of irrelevant documents, which also degrades the performance of QA systems. In this paper, we introduce Paragraph Ranker which ranks paragraphs of retrieved documents for a higher answer recall with less noise. We show that ranking paragraphs and aggregating answers using Paragraph Ranker improves performance of open-domain QA pipeline on the four open-domain QA datasets by 7.8% on average.

1 Introduction

With the introduction of large scale machine comprehension datasets, machine comprehension models that are highly accurate and efficient in answering questions given raw texts have been proposed recently (Seo et al., 2016; Xiong et al., 2016; Wang et al., 2017c). While conventional machine comprehension models were given a paragraph that always contains an answer to a question, some researchers have extended the models to an open-domain setting where relevant documents have to be searched from an extremely large knowledge source such as Wikipedia (Chen et al., 2017; Wang et al., 2017a). However, most of the open-domain QA pipelines depend on traditional information retrieval systems which use TF-IDF rankings (Chen et al., 2017; Wang et al., 2017b). Despite the efficiency of the traditional retrieval systems, the documents retrieved and ranked at the top by such systems often do not contain answers to questions. However, simply increasing the number of top ranked documents to find answers also increases the number of irrelevant documents. The tradeoff between reading more documents and minimizing noise is frequently observed in previous works that defined the N number of top documents as a hyperparameter to find (Wang et al., 2017a).

In this paper, we tackle the problem of ranking the paragraphs of retrieved documents for improving the answer recall of the paragraphs while filtering irrelevant paragraphs. By using our simple but efficient Paragraph Ranker, our QA pipeline considers more documents for a high answer recall, and ranks paragraphs to read only the most relevant ones. The work closest to ours is that of Wang et al. (2017a). However, whereas their main focus is on re-ranking retrieved sentences to maximize the rewards of correctly answering the questions, our focus is to increase the answer recall of paragraphs with less noise. Thus, our work is complementary to the work of Wang et al. (2017a).

Our work is largely inspired by the field of information retrieval called *Learning to Rank* (Liu et al., 2009; Severyn and Moschitti, 2015). Most learning to rank models consist of two parts: encoding networks and ranking functions. We use bidirectional long short term memory (Bi-LSTM) as our encoding network, and apply various ranking functions proposed by previous works (Severyn and Moschitti, 2015; Tu et al., 2017). Also, as the time and space complexities of ranking paragraphs are much larger than those of ranking sentences (Severyn and Moschitti, 2015), we resort to negative sampling (Mikolov et al., 2013) for an efficient training of our Paragraph Ranker.

*Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 565–569
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

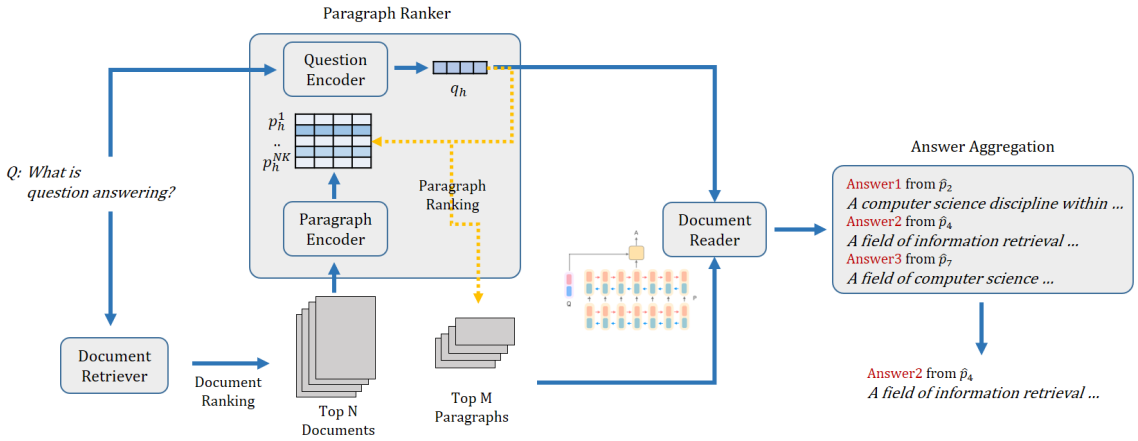

Figure 1: Our proposed open-domain QA pipeline with Paragraph Ranker

Our pipeline with Paragraph Ranker improves the exact match scores on the four open-domain QA datasets by 7.8% on average. Even though we did not further customize Document Reader of DrQA (Chen et al., 2017), the large improvement in the exact match scores shows that future researches would benefit from ranking and reading the more relevant paragraphs. By a qualitative analysis of ranked paragraphs, we provide additional evidence supporting our findings.

2 Open-Domain QA Pipeline

Most open-domain QA systems are constructed as pipelines that include a retrieval system and a reader model. We additionally built Paragraph Ranker that assists our QA pipeline for a better paragraph selection. For the retrieval system and the reader model, we used Document Retriever and Document Reader of Chen et al. (2017).[1] The overview of our pipeline is illustrated in Figure 1.

2.1 Paragraph Ranker

Given N number of documents retrieved from Document Retriever, we assume that each document contains K number of paragraphs on average. Instead of feeding all NK number of paragraphs to Document Reader, we select only M number of paragraphs using Paragraph Ranker. Utilizing Paragraph Ranker, we safely increase N for a higher answer recall, and reduce the number of paragraphs to read by selecting only top ranked paragraphs.

Given the retrieved paragraphs P_i where i ranges from 1 to NK, and a question Q, we encode each paragraph and the question using two separate RNNs such as Bi-LSTM. Representations of each paragraph and the question are calculated as follows:

$$p_h^i = \text{BiLSTM}_p(\text{E}(P_i)) \quad q_h = \text{BiLSTM}_q(\text{E}(Q))$$

where BiLSTM$(\cdot)$ returns the concatenation of the last hidden state of forward LSTM and the first hidden state of backward LSTM. E$(\cdot)$ converts tokens in a paragraph or a question into pretrained word embeddings. We use GloVe (Pennington et al., 2014) for the pretrained word embeddings.

Once each paragraph and the question are represented as p_h^i and q_h, we calculate the probability of each paragraph to contain an answer of the question as follows:

$$p(P_i|Q) = \frac{1}{1 + e^{-s(p_h^i, q_h)}}$$

where we have used similarity function $s(\cdot, \cdot)$ to measure the probability of containing answer to the question Q in the paragraph P_i. While Wang and Jiang (2015) adopted high capacity models such as Match-LSTM for measuring the similarity between paragraphs and questions, we use much simpler scoring functions to calculate the similarity more efficiently. We tested three different scoring functions: 1) the dot product of p_h^i and q_h, 2) the bilinear form ${p_h^i}^T W q_h$, and 3) a multilayer perceptron (MLP) (Severyn and Moschitti, 2015). While utilizing MLP takes much more time than the other two functions, recall of MLP was similar to that of the dot product. Also, as recall of the bilinear form was worse than that of the dot product, we use the dot product as our scoring function.

[1] https://github.com/facebookresearch/DrQA

Due to the large size of NK, it is difficult to train Paragraph Ranker on all the retrieved paragraphs.[2] To efficiently train our model, we use a negative sampling of irrelevant paragraphs (Mikolov et al., 2013). Hence, the loss function of our model is as follows:

$$J(\Theta) = -\log p(P_i|Q) - E_{k\sim p_n}[\log(1 - p(P_k|Q))]$$

where k indicates indexes of negative samples that do not contain the answer, and Θ denotes trainable parameters of Paragraph Ranker. The distribution of negative samples are defined as p_n. We use the distribution of all the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) training paragraphs as p_n .

Based on the rank of each paragraph from Paragraph Ranker and the rank of source document from Document Retriever, we collect top M paragraphs to read. We combine the ranks by the multiplication of probabilities $p(P_i|Q)$ and $\tilde{p}(D_i|Q)$ to find most relevant paragraphs where $\tilde{p}(D_i|Q)$ denotes TF-IDF score of a source document D_i.

2.2 Answer Aggregation

We feed M paragraphs to Document Reader to extract M answers. While Paragraph Ranker increases the probability of including answers in the top M ranked paragraphs, aggregation step should determine the most probable answer among the M extracted answers. Chen et al. (2017) and Clark et al. (2017) used the unnormalized answer probability from the reader. However, as the unnormalized answer probability is very sensitive to noisy answers, Wang et al. (2017b) proposed a more sophisticated aggregation methods such as coverage-based and strength-based re-rankings.

In our QA pipeline, we incorporate the coverage-based method by Wang et al. (2017b) with paragraph scores from Paragraph Ranker. Although strength-based answer re-ranking showed good performances on some datasets, it is too complex to efficiently re-rank M answers. Given the M candidate answers $[A_1, ..., A_M]$ from each paragraph, we aggregate answers as follows:

$$\begin{aligned}\hat{A} &= \arg\max_{A_i} p(A_i|Q) \\ &= \arg\max_{A_i} \tilde{p}(A_i|P_i, Q)^{\alpha} p(P_i|Q)^{\beta} \tilde{p}(D_i|Q)^{\gamma}\end{aligned} \quad (1)$$

where $\tilde{p}(A_i|P_i, Q)$ denotes the unnormalized answer probability from a reader given the paragraph P_i and the question Q. Importance of each score is determined by the hyperparamters α, β, and γ. Also, we add up all the probabilities of the duplicate candidate answers for the coverage-based aggregation.

3 Experiments

3.1 Datasets

We evaluate our pipeline with Paragraph Ranker on the four open-domain QA datasets. Wang et al. (2017a) termed SQuAD without relevant paragraphs for the open-domain QA as **SQuAD$_{\text{OPEN}}$**, and we use the same term to denote the open-domain setting SQuAD. **CuratedTrec** (Baudiš and Šedivỳ, 2015) was created for TREC open-domain QA tasks. **WebQuestions** (Berant et al., 2013) contains questions from Google Suggest API. **WikiMovies** (Miller et al., 2016) contains questions regarding movies collected from OMDb and the MovieLens database. We pretrain Document Reader and Paragraph Ranker on the SQuAD training set.[3]

3.2 Implementation Details

Paragraph Ranker uses 3-layer Bi-LSTM networks with 128 hidden units. On SQuAD$_{\text{OPEN}}$ and CuratedTrec, we set α, β, and γ of Paragraph Ranker to 1. Due to the different characteristics of questions in WebQuestion and WikiMovies, we find α, β, and γ based on the validation QA pairs of the two datasets. We use $N = 20$ for the number of documents to retrieve and $M = 200$ for the number of paragraphs to read for all the four datasets. We use Adamax (Kingma and Ba, 2014) as the optimization algorithm. Dropout is applied to LSTMs and embeddings with $p = 0.4$.

3.3 Results

In our experiments, **Paragraph Ranker** ranks only paragraphs, and answers are determined by unnormalized scores of the answers. **Paragraph Ranker + Answer Agg.** sums up the unnormalized probabilities of duplicate answers (i.e., $\beta = \gamma = 0$). **Paragraph Ranker + Full Agg.** aggregates answers using Equation 1 with the coverage-based aggregation.

[2] $NK \approx 350$ when $N = 5$ in SQuAD QA pairs.

[3] On SQuAD development set, pretrained Document Reader achieves 69.1% EM, and pretrained Paragraph Ranker achieves 96.7% recall on the top 5 paragraph .

Model	SQuAD$_{\text{OPEN}}$		CuratedTrec		WebQuestions		WikiMovies	
	EM	Recall	EM	Recall	EM	Recall	EM	Recall
DrQA (Chen et al., 2017)	27.1	77.8	19.7	86.0	11.8	74.4	24.5	70.3
DrQA + Fine-tune	28.4	-	25.7	-	19.5	-	34.3	-
DrQA + Multitask	29.8	-	25.4	-	20.7	-	36.5	-
R^3 (Wang et al., 2017a)	29.1	-	28.4	-	17.1	-	38.8	-
Par. Ranker	28.5	83.1	26.8	91.4	18.0	70.7	33.4	79.7
Par. Ranker + Answer Agg.	28.9	-	28.2	-	18.4	-	33.9	-
Par. Ranker + Full Agg.	**30.2**	-	**35.4**	-	**19.9**	-	**39.1**	-

Table 1: Open-domain QA results on four QA datasets. Best scores including those of the Multitask model are underlined. Bold texts denote best scores excluding those of the Multitask model.

Question #1	*What position does Von Miller play?* (SQuAD$_{\text{OPEN}}$)
Answer	*linebacker*, *linebacker*, *linebacker*
Doc. Retriever	(Top-1 document) *Ferdinand Miller, from 1875 von Miller ... was an ore caster, ... Miller was born and died in Munich. He was the son of the artisan and First ... Ferdinand was simultaneously ennobled. Ferdinand's younger brother was the ...*
Par. Ranker	(Top-1 paragraph) *The two teams exchanged field goals ... with a 48-yarder by ...*
	(Top-2 paragraph) *Luck was strip-sacked by Broncos' **linebacker** Von Miller ...*
	(Top-3 paragraph) *Broncos' **linebacker** Von Miller forced a fumble off RGIII ...*

Table 2: Top ranked paragraphs by Paragraph Ranker based on SQuAD$_{\text{OPEN}}$

In Table 1, we summarize the performance and recall of each model on open-domain QA datasets. We define recall as the probability of read paragraphs containing answers. While Reinforced Reader-Ranker (R^3) (Wang et al., 2017a) performs better than DrQA on the three datasets (SQuAD$_{\text{OPEN}}$, CuratedTrec, WikiMovies), Paragraph Ranker + Full Agg. outperforms both DrQA and R^3. Paragraph Ranker + Full Agg. achieved 3.78%, 24.65%, 2.05%, 0.77% relative improvements in terms of EM on SQuAD$_{\text{OPEN}}$, CuratedTrec, WebQuestion, and WikiMovies, respectively (7.8% on average). It is noticeable that our pipeline with Paragraph Ranker + Full Agg. greatly outperforms DrQA + Multitask in SQuAD$_{\text{OPEN}}$ and CuratedTrec.

3.4 Analysis

In Table 2, we show 3 random paragraphs of the top document returned by Document Retriever, and the top 3 paragraphs ranked by Paragraph Ranker from the top 40 documents. As Document Retriever largely depends on matching of query tokens with document tokens, the top ranked document is usually the document with most tokens matching the query. However, Question 1 includes the polysemy of the word "play" which makes it more difficult for Document Retriever to perform effectively. Our Paragraph Ranker well understands that the question is about a sports player not a musician. The top 1-3 paragraphs for the second question came from the 30th, 7th, and 6th documents, respectively, ranked by Document Retriever. This shows that increasing number of documents to rank helps Paragraph Ranker find more relevant paragraphs.

4 Conclusion

In this paper, we present an open-domain question answering pipeline and proposed Paragraph Ranker. By using Paragraph Ranker, the QA pipeline benefits from increased answer recall from paragraphs to read, and filters irrelevant documents or paragraphs. With our simple Paragraph Ranker, we achieve state-of-the-art performances on the four open-domain QA datasets with large margins. As future works, we plan to further improve Paragraph Ranker based on the researches on learning to rank.

Acknowledgement

This research was supported by National Research Foundation of Korea (NRF-2017R1A2A1A17069645, NRF-2017M3C4A7065887), and the Korean MSIT (Ministry of Science and ICT) under the National Program for Excellence in SW (2015-0-00936) supervised by the IITP (Institute for Information & communications Technology Promotion)

References

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 222–228. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

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 neural information processing systems*, pages 3111–3119.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Zhucheng Tu, Matt Crane, Royal Sequiera, Junchen Zhang, and Jimmy Lin. 2017. An exploration of approaches to integrating neural reranking models in multi-stage ranking architectures. *arXiv preprint arXiv:1707.08275*.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017a. R3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017b. Evidence aggregation for answer re-ranking in open-domain question answering. *arXiv preprint arXiv:1711.05116*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017c. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.