Final Report on Multi-hop Reading Comprehension

Woojeong Jin
University of Southern California
woojeong.jin@usc.edu

Abstract

Learning multi-hop reasoning has been a key challenge for reading comprehension models. Ideally, a model should not be able to perform well on a multi-hop question answering task without doing multi-hop reasoning. In this final report, we investigate works on multi-hop reading comprehension including datasets and models. Furthermore, we propose MRCNET, which is composed of two modules: a relevance module and an answering module. The relevance module finds the two most relevant paragraphs among given 10 paragraphs, and the answering module finds answers given the two relevant paragraphs. Our results on this model shows the effectiveness on finding relevant articles and a multi-hop question answering task.

1 Introduction

Multi-hop reading comprehension (RC) or question answering requires the aggregation of evidence across several paragraphs to answer a question. Table 1 shows an example of single-hop and multi-hop questions. A single-hop question "Which player is named 2015 Diamond Head Classic's MVP?" requires finding the player who won MVP from one paragraph. However, a multi-hop question requires further reasoning, which is first finding the player, and then finding the team that player plays for from another paragraph.

In this project, we propose MRCNET, a system for multi-hop RC, which consists two modules: a relevance module and an answering module. The relevance module learns relevance of each paragraph to find the most relevant paragraphs, while the answering module learns how to find answers given the paragraphs. The relevance module provides relevance scores for each paragraphs and the paragraphs are ranked by the score. We assume that answers are in the top-two ranked paragraphs. Given the two paragraphs, the answering module finds the most plausible answers.

Single-hop	Which player is named 2015 Diamond Head Classic's MVP?
Multi-hop	Which team does the player named 2015 Diamond Head Classic's MVP play for?

Table 1: An example of single-hop and multi-hop questions from HOTPOTQA. A multi-hop question requires multi-hop reasoning.

Our experiments show effectiveness of the answering module in MRCNET. Unfortunately, we do not have results of our MRCNET. In this midterm report, we study the datasets and previous work.

2 Dataset

Before diving into previous work, we first examine the datasets (Dua et al., 2019; Yang et al., 2018; Welbl et al., 2018a; Talmor and Berant, 2018). Among them, we study two datasets: WikiHop (Welbl et al., 2018a) and HOTPOTQA (Yang et al., 2018). One key difference is that HotpotQA is span-based (the answer is a span of the passage) while WikiHop is multiple-choice.

WikiHop (Welbl et al., 2018b). Wikihop is English dataset designed for text understanding across multiple documents. The dataset consists of 40k+ questions, answers, and passages, where each passage consists of several documents collected from Wikipedia. Questions are posed as a query of a relation r followed by a head entity h, with the task being to find the tail entity t from a set of entity candidates E. Annotators followed links between documents and were required to use multiple documents to get the answer.

HOTPOTQA (Yang et al., 2018). HOTPOTQA is a dataset with 113k English Wikipedia-based question-answer pairs. The questions are diverse, falling into several categories: inferring the bridge entity, intersection, and comparison. They also introduce a subset of yes/no questions in comparison questions. All require finding and reasoning over multiple supporting documents to answer. Models should choose answers by selecting variable-length spans from the documents. Sentences relevant to finding the answer are annotated in the dataset as "supporting facts" so models can use these at training time as well.

In this project, we focus and experiment on the HotpotQA dataset.

3 Related Work

In this section, we review the related work on HotpotQA (Min et al., 2019b; Xiao et al., 2019; Nishida et al., 2019; Ding et al., 2019; Feldman and El-Yaniv, 2019). Before digging into each method, Multi-hop reading comprehension has two benchmark settings on HotpotQA: distractor and full wiki (open-domain) setting. In the first setting, to challenge the model to find the true supporting facts in the presence of noice, there are 8 paragraphs from Wikipedia as distractors, and 2 gold paragraphs, which contain answers and supporting facts. The second setting truly test the model's ability to locate relevant facts as well as reasoning about them by requiring it to answer the question given the first paragraphs of all Wikipedia articles without gold paragraphs specified.

We divide the previous work into two groups: models for the distractor setting (Xiao et al., 2019; Min et al., 2019b; Nishida et al., 2019), models for the full wiki (open-domain) setting (Min et al., 2019b; Ding et al., 2019; Feldman and El-Yaniv, 2019). Among these work, Xiao et al. (2019); Ding et al. (2019) use entity-level Graph Neural Network (GNN) across multiple paragraphs.

Xiao et al. (2019) proposed the Dynamically Fused Graph Network (DFGN). Their intuition is drawn from the human reasoning process for QA. One starts from an entity of interest in the query, focuses on the words surrounding the start entities, connects to some related entity either found in the neighborhood or linked by the same surface mention, repeats the step to form a reasoning chain, and lands on some entity or snippets likely to be the answer. More specifically, they first find a paragraph and construct an entity graph. From these paragraph and graph, they find an answer.

Min et al. (2019b) proposed DecompRC that learns to break compositional multi-hop questions into simpler, single-hop sub-questions using spans from the original question. First, DecompRC decomposes the original, multi-hop question into several single-hop sub-questions according to a few reasoning types in parallel, based on span predictions. Then, for every reasoning types DecompRC leverages a single-hop reading comprehension model to answer each sub-question, and combines the answers according to the reasoning type. Finally, it leverages a decomposition scorer to judge which decomposition is the most suitable, and outputs the answer from that decomposition as the final answer.

Nishida et al. (2019) proposed Query Focused Extractor (QFE) model for evidence extraction. So they focus on the evidence extraction based on the previous work (Yang et al., 2018). QFE is inspired by extractive summarization models; compared with the existing method, which extracts each evidence sentence independently, it sequentially extracts evidence sentences by using an RNN with an attention mechanism on the question sentence. It enables QFE to consider the dependency among the evidence sentences and cover important information in the question sentence.

The following work is for the full wiki setting. Thus, finding relevant paragraphs is crucial for their performances.

Ding et al. (2019) proposed Cognitive Graph QA (CogQA), which comprises System 1 and 2 modules. System 1 extracts question-relevant entities and answer candidates from paragraphs and encodes their semantic information. Extracted entities are organized as a cognitive graph. System 2 conducts the reasoning procedure over the graph, and collects clues to guide System 1 to better extract next-hop entities. The above process is iterated until all possible answers are found, and then the final answer is chosen based on reasoning results from System 2.

Feldman and El-Yaniv (2019) proposed MUPPET (multi-hop paragraph retrieval) which relies on the following basic scheme consisting of two main components: (a) a paragraph and question encoder, and (b) a paragraph reader. The encoder is trained to encode paragraphs into d-dimensional vectors, and to encode questions into search vec-

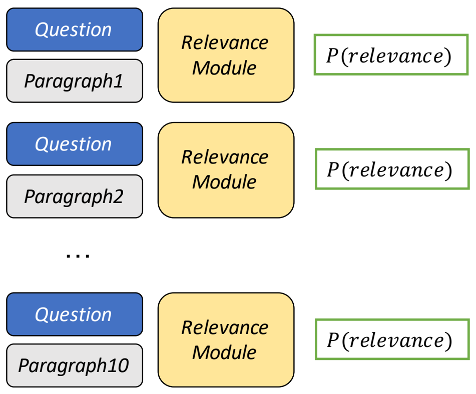

Figure 1: The relevance module learns to find how relevant each paragraph to the question. From the probabilities of relevance we can pick the two most relevant paragraphs.

tors in the same vector space. Then a maximum inner product search algorithm is applied to find the most similar paragraphs to a given question. The most similar paragraphs are then passed to the paragraph reader, and extracts the most probable answer to the question.

The above work showed good performances, but they are out of date. Readers can find the better models on the leaderboard of HotpotQA (https://hotpotqa.github.io).

Temporal Question Answering. We review related work on temporal question answering. A temporal question is any question, which contains a temporal expression, a temporal signal, or whose answer is of temporal nature. For example, ”Who won the state of texas in 2008?” and ”Who was the president after John F. Kennedy died?” are temporal questions. There are only a few work on the temporal question answering task: (Sun et al., 2018; Jia et al., 2018). To deal with temporal and causal relations, Sun et al. (2018) makes an event graph which is a directed graph where vertices represent events which are connected by edges. With generated graphs, they seek to find ordering of events and answer the question. On the other hand, Jia et al. (2018) studies temporal question answering on Knowledge Bases. They first decompose and rewrite the question into non-temporal sub-questions and temporal constraints. Then, they obtain answers and dates for temporal constraints from Knowledge Bases.

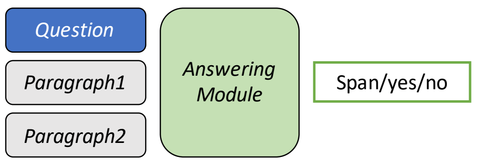

Figure 2: The answering module seeks to find answers for the question. The answer can be answer span, “yes”, or “no.”

4 Proposed Method

4.1 Overview

In multi-hop reading comprehension, a system answers a question over a collection of paragraphs by combining evidence from multiple paragraphs. HotpotQA dataset has two *gold* paragraphs which have supporting evidences and answers for the given questions among 10 given paragraphs. The key idea of our method is that we need to find the two *gold* paragraphs first and then concatenate the paragraphs. Answering module will find answers from the paragraphs.

We propose MRCNet for multi-hop reading comprehension via two modules: a relevance module and an answering module. MRCNet answers questions through a two step process:

1. First, a relevance module in MRCNet finds the two most relevant paragraphs. The module takes the question and each paragraph as input, and provides a relevance score which is a probability of containing supporting facts or an answer in the paragraph.

2. An answering module assumes that the two relevant paragraphs have answers. The question and the two paragraphs will be given to the module and the module finds an answer for the question.

Details of each module will be described in Sections 4.2 (the relevance module) and 4.3 (the answering module).

4.2 Relevance Module

The relevance module scores each paragraph for the given question. In Figure 1, the question and each paragraph are given to the module. We adopt BERT (Devlin et al., 2018) and another fully connected layer followed by sigmoid function to obtain probabilities of relevance.

The model receives a question $Q = [q_1, ..., q_m]$ and a single paragraph $P = [p_1, ..., p_n]$ as input. Following (Devlin et al., 2018), $S = [[\text{CLS}], q_1, ..., q_m, [\text{SEP}], p_1, ..., p_n]$ where [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token to separate questions and answers, is fed into BERT:

$$S' = f(S) \in \mathbb{R}^{h \times (m+n+2)}, \tag{1}$$

where f is the BERT model and h is the hidden dimension of BERT. Next, a fully connected layer with a sigmoid function is applied on the output of the [CLS] token to generate probabilities:

$$P(y_{\text{relevance}}) = \sigma(W_1 \cdot S'_{[\text{CLS}]}), \tag{2}$$

where $y_{\text{relevance}}$ indicates the paragraph is relevant, σ is a sigmoid function, and $W_1 \in \mathbb{R}^{h \times 1}$.

The relevance module yields probability of relevance and from the probabilities we can pick top-2 paragraphs which will be used in the answering module. This module is optimized by binary cross entropy loss.

4.3 Answering Module

The answering module finds answers given two paragraphs. In Figure 2, the answering module receives the question and two paragraphs as input, and it yields the answer (`span`, `yes`, or `no`).

Similarly to the relevance module, the answering module uses BERT to get hidden representations of each tokens S'. On top of these representations, a classifiers uses max-pooling and fully connected layer to generate three scalars:

$$[y_{span}, y_{yes}, y_{no}] = \text{softmax}(W_2 \cdot \text{maxpool}(S')), \tag{3}$$

where y_{span}, y_{yes}, and y_{no} indicate the answer is either a `span`, `yes`, or `no`, and $W_2 \in \mathbb{R}^{h \times 3}$. If the answer is `span`, then the module needs to find an answer span. The answer span is obtained separately following (Devlin et al., 2018):

$$P(T_i = \text{start}) = \text{softmax}(W_3 \cdot S'_i) \tag{4}$$

$$P(T_i = \text{end}) = \text{softmax}(W_4 \cdot S'_i), \tag{5}$$

where $W_3, W_4 \in \mathbb{R}^{h \times 1}$.

4.4 Loss functions

To train the relevance and answering modules, we define loss functions. We first define a loss function for the relevance module which is binary cross entropy loss.

Model	EM	F1
BiDAF	-	58.28
Single-paragraph BERT	-	67.08
DecompRC	55.20	69.63
DFGN	56.31	69.69
MRCNET	62.45	75.25

Table 2: Exact Match and F1 scores on HOTPOTQA. Our MRCNET outperforms other competitors.

Loss for relevance module. Given relevant (positive) paragraphs and irrelevant (negative) paragraphs, we can define a binary cross entropy function as follows:

$$L_1 = -\sum_{y \in Pos} logP(y) - \sum_{y \in Neg} (1 - logP(y)), \tag{6}$$

where $P(y)$ is the probability of whether the paragraph is relevant or not.

Loss for answering module. The answering module has two loss functions: one is for finding answer type (`span`, `yes`, `no`) (L_2), the other one is for finding answer spans (L_3). If the answer type is `span`, then the module needs to find answer spans.

Loss function for finding answer type as follows:

$$L_2 = -\sum_{y \in span} logP^{span}(y) - \sum_{y \in yes} logP^{yes}(y) - \sum_{y \in no} logP^{no}(y),$$

where P^{span} is the probability of *span*, P^{yes} is the probability of *yes*, and P^{no} is the probability of *no*.

If the answer type is `span`, we train the following loss function:

$$L_3 = -\sum_{y \in start} logP^{start}(y) - \sum_{y \in end} logP^{end}(y). \tag{7}$$

Then, total loss will be:

$$L_{tot} = L_1 + L_2 + L_3 \tag{8}$$

5 Experiments

In this section, we evaluate our proposed method on HOTPOTQA (Yang et al., 2018), a recently introduced multi-hop RC dataset over Wikipedia articles. We evaluate our method on a distractor setting which contains the question and a collection of 10 paragraphs: 2 paragraphs (gold paragraphs) are provided to crowd workers to write a multi-hop questions, and 8 distractor paragraphs are collected separately via TF-IDF between the question and the paragraph. The gold paragraphs contain supporting evidences and answers. In this experiment, we seek to find answers and test on the dev set.

5.1 Experimental Setup

Evaluation Metrics. We use two different metrics to evaluate model accuracy We use *Exact Match* and *(Macro-averaged) F1 score*. The exact match metric measures the percentage of predictions that match the ground truth answer exactly. The F1 scores measure the average overlap between the prediction and ground truth answer. The prediction and ground truth are treated as bags of tokens. With these bags of tokens, we compute their F1 scores.

Implementation Details. We use the cased version of BERT Tokenizer (Devlin et al., 2018) to tokenize all passages and questions. In the encoding stage, we also use a pre-trained BERT model as the encoder, and all the hidden state dimensions are set to 768. For optimization, we use Adam Optimizer (Kingma and Ba, 2014) with an initial learning rate of $1e^{-4}$.

Baselines. We compare our approach to published baselines: BiDAF (Seo et al., 2016), Single-paragraph BERT (Min et al., 2019a), DecompRC (Min et al., 2019b), and DFGN (Xiao et al., 2019). We test our method and baselines on a validation set since we couldn't get the test dataset.

5.2 Results

Table 2 shows the performances of our proposed method, MRCNET. Our method significantly outperforms other baselines. MRCNET shows 6% point improvement (EM) and 5% point improvement over DFGN. However, there are more methods on the leaderboard[1]. Some methods shows much better performances over the baselines, but we didn't include the results since they do not release results on a validation set.

5.3 Ablation Study

In this section, we evaluate the performances of the answering module and the relevance module in MRCNET.

[1]https://hotpotqa.github.io

Model	**EM**	**F1**
DFGN with gold paragraphs	55.67	69.15
MRCNET with gold paragraphs	62.57	76.51
MRCNET with ten paragraphs	54.58	67.22
MRCNET	62.45	75.25

Table 3: Exact Match and F1 scores on HOTPOTQA. Our MRCNET with gold paragraphs outperforms DFGN with gold paragraphs. With ten paragraphs, MRCNET significantly degrades the performance. MRCNET (with our relevance module) shows the similar performance to MRCNET with gold paragraphs.

Model	**accuracy**
Relevance module	96.44
Random predictor	48.32

Table 4: Accuracy of finding relevant articles. Two paragraphs out of ten paragraphs are relevant.

Answering Module. Given two gold paragraphs, we study the performance of the module. In Table 3, our propose method with two gold paragraphs shows higher performances than DFGN with two gold paragraphs. It shows 7% performances gains over the baseline, which shows the effectiveness of our method. This suggests that our answering module effectively finds answer spans, yes, or no. Also, MRCNET with ten paragraphs underperforms MRCNET with gold paragraphs. This means that finding most relevant paragraphs is important in performances.

Relevance Module. Our relevance module in MRCNET finds most relevant paragraphs given ten paragraphs. It is crucial to find appropriate paragraphs as shown in Table 3. Compared to MRCNET with ten paragraphs, MRCNET shows 8% performances gains in EM and F1. Also, MRCNET shows comparable results to MRCNET with gold paragraphs. Table 4 shows the performance of the relevance module in finding relevant articles. It achieves 96.44% accuracy, while the random predict shows 48.32% accurcy, which is huge improvement.

6 Conclusion

In this final report, we proposed MRCNET, which has two modules: the relevance module and the answering module. Our proposed method outperforms other baselines. However, we didn't include state of the arts since they don't release results on a validation set. Each module shows effectiveness on each experiment. Future work includes finding the most relevant sentences to give finer-grained evidences.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. *arXiv preprint arXiv:1905.05460*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. *arXiv preprint arXiv:1906.06606*.

Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018. Tequila: Temporal question answering over knowledge bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1807–1810. ACM.

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

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. Compositional questions do not necessitate multi-hop reasoning. *arXiv preprint arXiv:1906.02900*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Multi-hop reading comprehension through question decomposition and rescoring. *arXiv preprint arXiv:1906.02916*.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. 2019. Answering while summarizing: Multi-task learning for multi-hop qa with evidence extraction. *arXiv preprint arXiv:1905.08511*.

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

Yawei Sun, Gong Cheng, and Yuzhong Qu. 2018. Reading comprehension with graph-based temporal-casual reasoning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 806–817.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018a. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018b. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. *arXiv preprint arXiv:1905.06933*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.