Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering

Anonymous authors
Paper under double-blind review

Abstract

Answering questions that require multi-hop reasoning at web-scale requires retrieving multiple evidence documents, one of which often has little lexical or semantic relationship to the question. This paper introduces a new graph-based recurrent retrieval approach that learns to retrieve reasoning paths over the Wikipedia graph to answer multi-hop open-domain questions. Our retriever trains a recurrent neural network that learns to sequentially retrieve evidence documents in the reasoning path by conditioning on the previously retrieved documents. Our reader ranks the reasoning paths and extracts the answer span included in the best reasoning path. Experimental results demonstrate state-of-the-art results in two open-domain QA datasets showcasing the robustness of our method. Notably, our method achieves significant improvement in HotpotQA fullwiki and distractor settings, outperforming the previous best model by more than 10 points.[1]

1 Introduction

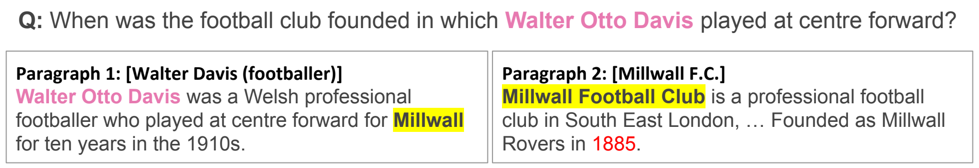
Open-domain Question Answering (QA) is the task of answering a question given a large collection of text documents (e.g., Wikipedia). Most state-of-the-art approaches for open-domain QA (Chen et al., 2017; Wang et al., 2018a; Lee et al., 2018; Yang et al., 2019) leverage non-parameterized models (e.g., TF-IDF or BM25) to retrieve a fixed set of documents, where the answer span can be extracted by a neural machine reading comprehension model. Despite the success of these pipeline methods in single-hop QA, they often fail to retrieve the required evidence for answering multi-hop questions e.g., the question in Figure 1. Multi-hop QA usually requires finding more than one evidence document, one of which often consists of little lexical overlap or semantic relationship to the original question. However, retrieving a fixed list of documents independently does not capture relationships between evidence documents through *bridge entities* that are required for multi-hop reasoning.

Recent open-domain QA methods learn end-to-end models to jointly retrieve and read documents (Seo et al., 2019; Lee et al., 2019). These methods, however, face challenges for entity-centric questions since compressing the necessary information into an embedding space does not capture lexical information in entities. Cognitive Graph (Ding et al., 2019) incorporates entity links between documents for multi-hop QA to extend the list of retrieved documents. This method, however, compiles a fixed list of documents independently and expects the reader to find the reasoning paths.

In this paper, we introduce a new recurrent graph-based retrieval method that learns to retrieve evidence documents as reasoning paths for answering complex questions. Our method sequentially retrieves each evidence document given the history of previously retrieved documents to form several reasoning paths in a graph of entities. Our method then leverages an existing reading comprehension model to answer questions by ranking the retrieved reasoning paths. The strong interplay between the retriever and reader enables our entire method to answer complex questions by exploring more accurate reasoning paths compared to other methods.

More specifically, our method (sketched in Figure 2) constructs the entity graph offline using Wikipedia hyperlinks to model the relationships between entities in documents. Our retriever trains a recurrent neural network to score reasoning paths in this graph by maximizing the likelihood of

[1]Our code will be publicly available.


Q: When was the football club founded in which Walter Otto Davis played at centre forward?

Paragraph 1: [Walter Davis (footballer)]
Walter Otto Davis was a Welsh professional footballer who played at centre forward for Millwall for ten years in the 1910s.

Paragraph 2: [Millwall F.C.]
Millwall Football Club is a professional football club in South East London, … Founded as Millwall Rovers in 1885.

Figure 1: An example of open-domain multi-hop question from HotpotQA. Paragraph 2 is unlikely to be retrieved using TFIDF retrievers due to little lexical overlap to the given question.

selecting a correct evidence paragraph at each step and fine-tuning paragraph BERT encodings. Our reader model is a multi-task learner to score each reasoning path according to its likelihood of containing and extracting the correct answer phrase. We leverage data augmentation and negative example mining for robust training of both models.

Our experimental results show that our method achieves the state-of-the-art results on HotpotQA fullwiki and HotpotQA distractor settings (Yang et al., 2018), outperforming the previous state-of-the-art methods by more than 10 points absolute gain on the full wiki setting. We also evaluate our approach on SQuAD Open without changing any architectural designs or hyperparameters, achieving performance comparable to the state of the art, which suggests that our retrieval method is robust across different datasets. Additionally, our framework provides interpretable insights into the underlying entity relationships used for multi-hop reasoning.

2 Related Work

Neural open-domain question answering Most current open-domain QA methods use a pipeline approach that includes a *retriever* and *reader*. Chen et al. (2017) incorporate a TF-IDF-based retriever with a state-of-the-art neural machine reading comprehension model. The subsequent work improves the heuristic retriever by re-ranking retrieved documents (Wang et al., 2018a;b; Lee et al., 2018; Lin et al., 2018). The performance of these methods is still bounded by the performance of the initial retrieval process. In multi-hop QA, non-parameterized retrievers face the challenge of retrieving the all relevant documents, one or some of which are lexically distant from the question. Recently, Lee et al. (2019) and Seo et al. (2019) introduce fully trainable models that retrieve a few candidates directly from large-scale Wikipedia collections. All these methods find evidence documents independently without the knowledge of previously selected paragraphs or relationships between documents. This would result in failing to conduct multi-hop retrieval, and therefore, similar but irrelevant retrieved evidence documents introduce noisy evidence input for the subsequent machine reading comprehension model.

Retrievers guided by entity links Most relevant to our work are recent studies that attempt to use entity links for multi-hop open-domain QA. Cognitive Graph (Ding et al., 2019) retrieves evidence documents offline using entities in questions and a TF-IDF-based retriever, and trains a machine reading comprehension model to jointly predict possible answer spans and next hop spans to extend the reasoning chain. We train our retriever to find reasoning paths directly, limiting the scope of retrieved documents processed by our reader. Most recently, Entity-centric IR (Godbole et al., 2019) combines retrieval with entity linking to retrieve paragraphs for multi-hop QA. Unlike our method, this method does not learn to retrieve reasoning paths and does not study the interplay between retriever and reader. Moreover, we empirically show significant improvement over both methods.

Multi-step (iterative) retrievers Similar to our recurrent retriever, multi-step retrievers explore multiple evidence documents iteratively. Multi-step reasoner (Das et al., 2019) repeats the retrieval process for a fixed number of steps, interacting with a machine reading comprehension model by reformulating the query in a latent space to enhance retrieval performance. Feldman & El-Yaniv (2019) also propose a query reformulation mechanism with a focus on multi-hop open-domain QA. These methods do not use the entity-link information during the iterative retrieval process. In contrast, our method leverages Wikipedia graph to explore and retrieve documents that are lexically or semantically distant to questions and significantly outperforms previous work in HotpotQA.

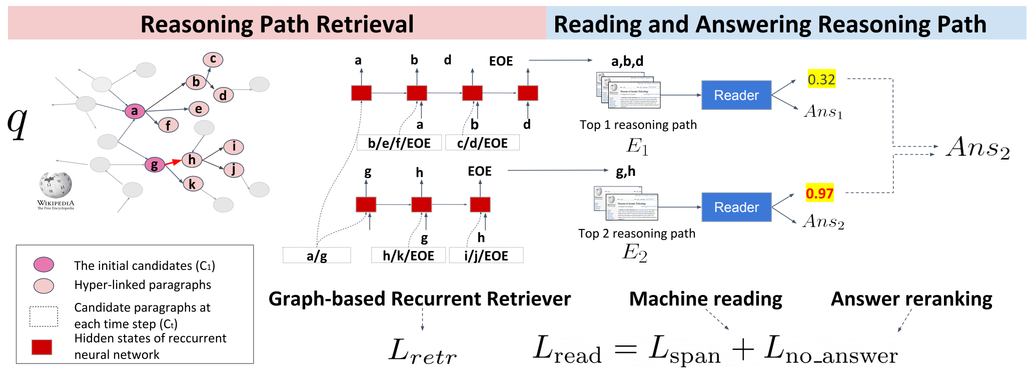

Reasoning Path Retrieval
Reading and Answering Reasoning Path
q
The initial candidates (C1)
Hyper-linked paragraphs
Candidate paragraphs at each time step (Ct)
Hidden states of reccurrent neural network
a/g
b/e/f/EOE
c/d/EOE
h/k/EOE
i/j/EOE
EOE
a,b,d
g,h
Reader
0.32
0.97
Ans_1
Ans_2
Top 1 reasoning path
E_1
Top 2 reasoning path
E_2
Graph-based Recurrent Retriever
Machine reading
Answer reranking
L_{retr}
$L_{\text{read}} = L_{\text{span}} + L_{\text{no\_answer}}$

Figure 2: Overview of our framework.

3 Open-domain Question Answering over Wikipedia Graph

Overview This paper introduces a new graph-based recurrent retrieval method (Section 3.1) that learns to find evidence documents as reasoning paths for answering complex questions. We then extend an existing machine reading model (Section 3.2) to answer questions given a collection of reasoning paths. Our method uses a strong interplay between retrieval and reading steps such that the retrieval method learns reasoning paths to narrow down the search space to likely reasoning paths for the reader, for robust pipeline process. Figure 2 sketches the overview of our QA model.

We formulate the QA task by decomposing the objective function into the retriever objective $S_{\text{retr}}(q, E)$ that sequentially selects evidence documents to form reasoning paths E and the reader objective $S_{\text{read}}(q, E, a)$ that finds the answer a in E.

$$\arg\max_{E,a} S(q, E, a) \quad \text{s.t.} \quad S(q, E, a) = S_{\text{retr}}(q, E) + S_{\text{read}}(q, E, a), \tag{1}$$

3.1 Learning to Retrieve Reasoning Paths

Our method learns to retrieve reasoning paths across a graph of Wikipedia paragraphs. Evidence paragraphs for answering a complex question do not necessarily have lexical overlaps with the question, but are likely to be entailed by entities mentioned in the question or first-hop paragraphs. To capture such multi-hop reasoning paths, our method constructs a graph of Wikipedia documents using hyperlinks and learns to retrieve the reasoning paths across this graph (Section 3.1.1).

Constructing the Wikipedia graph We use Wikipedia hyperlinks to construct the Wikipedia graph. Wikipedia hyperlinks connect articles in Wikipedia by following entity links. The nodes in the graph are the paragraphs in Wikipedia articles, and the directed edges correspond to the hyperlinks. We also consider symmetric within-document links, allowing a paragraph to hop to other paragraphs in the same article. The Wikipedia graph $\mathcal{G}$ is densely connected and covers a wide range of topics that provide useful evidence for open-domain questions.

3.1.1 Recurrent Retriever

Our method learns to retrieve reasoning paths E given the Wikipedia graph by iteratively selecting a paragraph in the reasoning chain.

General formulation with a recurrent retriever We use a Recurrent Neural Network (RNN) to model the reasoning paths for the question q. At the t-th time step ($t \geq 1$) our model selects a paragraph p_i among candidate paragraphs $\mathbf{C}_t$ given the current hidden state h_t of the RNN. We use BERT's (Devlin et al., 2018) [CLS] token representation to independently encode each candidate paragraph p_i along with q. We then compute the probability $P(p_i|h_t)$ that p_i is selected. The RNN selection procedure captures relationships between paragraphs in the reasoning path by conditioning on the selection history. The process is terminated when [EOE], the end-of-evidence symbol, is selected. This allows capturing reasoning paths with arbitrary length given each question. More specifically, the process of selecting p_i at the t-th step is formulated as follows:

$$w_i = \mathrm{BERT}_{[\mathrm{CLS}]}(q, p_i) \in \mathbb{R}^d, \tag{2}$$

$$P(p_i|h_t) = \sigma(w_i \cdot h_t + b), \tag{3}$$

$$h_{t+1} = \mathrm{RNN}(h_t, w_i) \in \mathbb{R}^d, \tag{4}$$

where $b \in \mathbb{R}^1$ is a bias term. Motivated by Salimans & Kingma (2016), we normalize the RNN states to control the scale of logits in Equation (3) and allow the model to learn multiple reasoning paths. The details of Equation (4) are described in Appendix. The next candidate set $\mathbf{C}_{t+1}$ is constructed to include paragraphs that are linked from the selected paragraph p_i in the Wikipedia graph.

Beam search for candidate paragraphs To navigate our retriever in the large-scale graph, we initialize candidate paragraphs with a TF-IDF-based retrieval and guide the search over the Wikipedia graph. More specifically, the initial candidate set $\mathbf{C}_1$ includes F paragraphs with the highest TF-IDF scores with respect to the question. We expand $\mathbf{C}_t$ ($t \geq 2$) by appending the [EOE] symbol.

We additionally use a beam search to explore paths in the directed graph. We define the score of a reasoning path $E = [p_i, \ldots, p_k]$ by multiplying the probabilities of selecting the paragraphs: $P(p_i|h_1) \ldots P(p_k|h_{|E|})$. The beam search outputs the top B reasoning paths $\mathbf{E} = \{E_1, \ldots, E_B\}$ with the highest scores to pass to the reader model i.e., $S(q, E, a) = S_{\mathrm{read}}(q, E, a)$ for $E \in \mathbf{E}$.

In terms of the computational cost, the number of the paragraphs processed by Equation (2) is bounded by $\mathcal{O}(|\mathbf{C}_1| + B \sum_{t \geq 2} \overline{|\mathbf{C}_t|})$, where B is the beam size and $\overline{|\mathbf{C}_t|}$ is the average size of $\mathbf{C}_t$ over the B hypothesises.

3.1.2 Training of the graph-based recurrent retriever

Data augmentation We train our retriever in a supervised fashion using evidence paragraphs annotated for each question. For multi-hop QA, we have multiple paragraphs for each question, and single paragraph for single-hop QA. We first derive a ground-truth reasoning path $g = [p_1, \ldots, p_{|g|}]$ using the available annotated data in each dataset. $p_{|g|}$ is set to [EOE] for the termination condition. To relax and stabilize the training process, we augment the training data with additional reasoning paths – not necessarily the shortest paths – that can derive the answer. In particular, we add a new training path $g = [p_r, p_1, \ldots, p_{|g|}]$ by adding a paragraph $p_r \in \mathbf{C_1}$ that has a high TF-IDF score and is linked to the first paragraph p_1 in the ground-truth path g. Adding these new training paths helps at the test time when the first paragraph in the reasoning path does not necessarily appear among the paragraphs that initialize the Wikipedia search using the heuristic TF-IDF retrieval.

Negative examples for robustness Our graph-based recurrent retriever needs to be trained to discriminate between relevant and irrelevant paragraphs at each step. We therefore use negative examples along with the ground-truth paragraphs; more specifically, we use two types of negative examples: (1) TF-IDF-based and (2) hyperlink-based ones. For single-hop QA, we only use the type (1). For multi-hop QA, we use both types, and the type (2) is especially important to find correct reasoning paths over the linked graph. We typically set the number of the negative examples to 50.

Loss function For the sequential prediction task, we estimate $P(p_i|h_t)$ independently in Equation (3) and use the binary cross-entropy loss to maximize probability values of all the possible paths. Note that using the widely-used cross-entropy loss with the softmax normalization over $\mathbf{C}_t$ is not desirable here; maximizing the probabilities of g and g_r contradict with each other. More specifically, the loss function of g at the t-th step is defined as follows:

$$L_{\mathrm{retr}}(p_t, h_t) = -\log P(p_t|h_t) - \sum_{\tilde{p} \in \tilde{\mathbf{C}}_t} \log\left(1 - P(\tilde{p}|h_t)\right), \tag{5}$$

where $\tilde{\mathbf{C}}_t$ is a set of the negative examples described above, and includes [EOE] for $t < |g|$. We exclude p_r from $\tilde{\mathbf{C}}_1$ for the sake of our multi-path learning. The loss is also defined with respect to g_r in the same way. All the model parameters, including those in BERT, are jointly optimized.

3.2 Reading and Answering Given Reasoning Paths

Our *reader* first verifies each reasoning path in $\mathbf{E}$, and finally outputs an answer span a from the most plausible reasoning path. We model the reader as a multi-task learning of (1) *machine reading* *comprehension,* that extracts an answer span from a reasoning path E using a standard approach (Seo et al., 2017; Xiong et al., 2017; Devlin et al., 2018), and (2) *answer re-ranking,* that re-ranks the retrieved reasoning paths by computing the probability that the path includes the answer.

For the span extraction task, we use BERT (Devlin et al., 2018), where the input is the concatenation of the question text and the text in E. We share the same model for the re-ranking model, we use the BERT's [CLS] representation to estimate the probability of selecting E to answer the question:

$$P(E|q) = \sigma(w_n \cdot u_E) \quad \text{s.t.} \quad u_E = \text{BERT}_{\text{[CLS]}}(q, E) \in \mathbb{R}^D, \tag{6}$$

where $w_n \in \mathbb{R}^D$ is a weight vector. At the inference time, we select the best evidence $E_{best} \in \mathbf{E}$ by $P(E|q)$, and output the answer span by S_{read}:

$$E_{best} = \underset{E \in \mathbf{E}}{\arg\max}\, P(E|q), \quad S_{\text{read}} = \underset{i,j,\, i \leq j}{\arg\max}\, P_i^{start} P_j^{end}, \tag{7}$$

where P_i^{start}, P_j^{end} denote the probability that the i-th and j-th tokens in E_{best} are the start and end positions, respectively, of the answer span. These probabilities are calculated in the same manner as in Devlin et al. (2018).

Training examples To train the multi-task reader, we use the ground-truth evidence paragraphs used for training our retriever. The re-ranking task needs to discriminate between relevant and irrelevant reasoning paths, and thus we augment the original training data with additional negative examples to simulate incomplete evidence. In particular, we add paragraphs that appear to be relevant to the given question but actually do not contain the answer to prevent our model from extracting incorrect spans with high confidence (Clark & Gardner, 2018).

For multi-hop QA, we select one ground-truth paragraph including the answer span, and swap it with one of the TF-IDF top ranked paragraphs. For single-hop QA, we also use the TF-IDF-based negative examples, and we simply replace the single ground-truth paragraph with those which do not include the expected answer string. For the distorted evidence $\tilde{E}$, we aim at minimizing $P(\tilde{E}|q)$.

Multi-task loss function The objective is the sum of cross entropy losses for the span prediction and re-ranking tasks. The loss for the question q and its evidence candidate E is as follows:

$$L_{\text{read}} = L_{\text{span}} + L_{\text{no\_answer}} = (-\log P_{y^{start}}^{start} - \log P_{y^{end}}^{end}) - \log P^r, \tag{8}$$

where y_q^{start} and y_q^{end} are the ground-truth start and end indices, respectively. $L_{\text{no\_answer}}$ corresponds to the loss of the re-ranking model, to discriminate the distorted paragraphs with no answers. P^r is $P(E|q)$ if E is the ground-truth evidence; otherwise $P^r = 1 - P(E|q)$. We mask the span losses for negative examples, in order avoid unexpected effects to the span predictions.

4 Experiments

4.1 Experimental Setup

We evaluate our method in two open-domain Wikipedia-sourced datasets: HotpotQA and SQuAD Open. More details about our experimental setup can be found in Appendix.

HotpotQA HotpotQA (Yang et al., 2018) is a human-annotated large-scale multi-hop QA dataset. The answer to a multi-hop question can be extracted from a collection of 10 paragraphs in the *distractor* setting, and from the entire Wikipedia in the *fullwiki* setting. Two evidence paragraphs are accociated with each question in the train data. The dataset also provides annotations to evaluate the prediction of supporting sentences. We adapt our retriever to the supporting fact prediction task.

SQuAD Open SQuAD Open (Chen et al., 2017) is composed of questions from the original SQuAD dataset (Rajpurkar et al., 2016). This is a single-hop QA task, and single paragraph is accosiated with each question in the training data.

Metrics For the two datasets, we report standard F1 and EM scores to evaluate the overall QA accuracy to find the correct answers. For HotpotQA, we also report Supporting Fact F1 (SP F1) and Supporting Fact EM (SP EM) to evaluate the sentence-level retrieval accuracy. To evaluate the retrieval accuracy for the complex multi-hop reasoning, we use the following metrics: *Answer Recall*

	fullwiki				distractor			
	QA		SP		QA		SP	
Models	F1	EM	F1	EM	F1	EM	F1	EM
SemanticRetrievalMRS (Nie et al., 2019)	58.8	46.5	71.5	39.9	–	–	–	–
Cognitive Graph (Ding et al., 2019)	49.4	37.6	58.5	23.1	–	–	–	–
DecompRC (Min et al., 2019b)	43.3	–	–	–	70.6	–	–	–
MUPPET (Feldman & El-Yaniv, 2019)	40.4	31.1	47.7	17.0	–	–	–	–
QFE (Nishida et al., 2019)	–	–	–	–	68.7	53.7	**84.7**	**58.8**
DFGN (Xiao et al., 2019)	–	–	–	–	69.2	55.4	–	–
Baseline (Yang et al., 2018)	34.4	24.7	41.0	5.3	58.3	44.4	66.7	22.0
Ours (Reader: BERT wwm)	**69.6**	**56.4**	**72.3**	**44.2**	**78.0**	**64.2**	83.2	54.0
Ours (Reader: BERT base)	65.2	52.1	72.0	43.0	72.6	58.7	82.2	52.8

Table 1: **HopotQA development set results**: QA and SP (supporting fact prediction) results on HotpotQA's fullwiki and distractor settings. "–" denotes no results are available.

(AR), which evaluates the recall of the answer string among top paragraphs (Wang et al., 2018a; Das et al., 2019), *Paragraph Recall (PR),* which evaluates the recall of the ground-truth paragraph among the top retrieved paragraphs, *Paragraph Exact Match (P EM),* which evaluates if both of the ground truth paragraphs for multi-hop reasoning are included among the retrieved paragraphs, and *Exact Match (EM),* which evaluates if the final answer exactly matches the ground truth answer.

The Wikipedia graph To construct the Wikipedia graph, the hyperlinks are automatically extracted from the raw HTML source files. Directed edges are added between a paragraph p_i and all of the paragraphs included in the target article. The constructed graph consists of 32.7M nodes and 205.4M edges. For HotpotQA we only use the introductory paragraphs in the graph that includes about 5.2M nodes and 23.4M edges.

Implementation details We use the pre-trained BERT models (Devlin et al., 2018) using the uncased base configuration $(d = 768)$ for our retriever and the whole word masking uncased large (wwm) configuration $(d = 1024)$ for our readers. We follow Chen et al. (2017) for the TF-IDF-based retrieval model and use the same hyper-parameters. If not specified, we set the number of the initial TF-IDF-based paragraphs F to 500 and set the beam size $B = 8$.

4.2 Overall Results

Table 1 compares our method with previous published methods on the HotpotQA development set. Our method significantly outperforms all the published results across all of the QA evaluation metrics under both full-wiki and distractor settings. Notably, our method achieves 10.8 F1 and 9.9 EM gains compared to state-of-the-art SemanticRetrievalMRS model (Nie et al., 2019). These results confirm the significance of our recurrent reasoning path retrieval method to avoid independent selection of paragraphs. Moreover, our method shows significant improvement in predicting supporting facts. Table 2 further verifies the effectiveness of our method on the official hidden test set.[2] In particular, we outperform all the published and unpublished results by large margins.

Table 3 shows that our multi-path retrieval and re-ranking framework achieves competitive results on the single-hop QA task, SQuAD. Our model is comparable to the most recent state-of-the-art model (Wang et al., 2019), and outperforms all other published work by a large margin.

4.3 Performance of Reasoning Path Retrieval

We compare our recurrent retriever with competitive retrieval methods for open-domain QA.

TF-IDF (Chen et al., 2017), the widely used retrieval method that scores paragraphs according to the TF-IDF scores of the question-paragraph pairs. We simply select the top-2 paragraphs.
Re-rank (Nogueira & Cho, 2019) that learns to retrieve paragraphs by fine-tuning BERT to re-rank the top F (=20) TF-IDF paragraphs. We select the top-2 paragraphs after re-ranking.
Re-rank 2hop which extends Re-rank to accommodate two-hop reasoning. It first adds paragraphs

[2]The leaderboard results are at `https://hotpotqa.github.io/` (on September 24th, 2019).

Models (*: unpublished)	QA		SP	
	F1	EM	F1	EM
SemanticRetrievalMRS	57.3	45.3	70.8	38.7
Cognitive Graph	48.9	37.1	57.7	22.8
Entity-centric IR	46.3	35.4	43.2	0.06
MUPPET	40.3	30.6	47.3	16.7
DecompRC	40.7	30.0	–	–
QFE	38.1	28.7	44.4	14.2
Baseline	32.9	24.0	37.7	3.9
Transformer-XH*	60.8	49.0	70.0	41.7
Entity-centric BERT*	53.1	41.8	57.3	26.3
GoldEn Retriever*	48.6	37.9	64.2	30.7
Ours	**68.9**	**56.0**	**73.0**	**44.1**

Table 2: **HopotQA fullwiki test set results**: results on the HotpotQA fullwiki setting on the hidden test set on the official leaderboard at the time of submission (September 24, 2019).

Models	F1	EM
DrQA (Chen et al., 2017)	–	29.8
R^3 (Wang et al., 2018a)	37.5	29.1
multi-step Reasoner (Das et al., 2019)	39.2	31.9
MINIMAL (Min et al., 2018)	42.5	34.7
DENSPI-hybrid (Seo et al., 2019)	44.4	36.2
MUPPET (Feldman & El-Yaniv, 2019)	46.2	39.3
BERTserini (Yang et al., 2018)	46.1	38.6
RE^3 (Hu et al., 2019)	50.2	41.9
multi-passaege (Wang et al., 2019)	**60.9**	**53.0**
BM25+BERT (Lee et al., 2019)	–	33.2
ORQA (Lee et al., 2019)	–	20.2
Ours	56.9	49.0

Table 3: **SQuAD Open results**: we report F1 and EM scores on the test set of SQuAD Open, following previous work.

Models	AR	PR	P EM	EM
Ours	86.2	92.6	71.6	51.1
TF-IDF	39.7	66.9	10.0	18.2
Re-rank	55.1	85.9	29.6	35.7
Re-rank 2hop	56.0	70.1	26.1	38.8
Entity-centric IR	63.4	87.3	34.9	42.0
Cognitive Graph	76.0	87.6	57.8	37.6

Table 4: **Retrieval evaluation**: Comparing our retrieval method with other methods across Answer Recall, Paragraph Recall, Paragraph EM, and QA EM metrics.

Settings	F1	EM
full	63.9	51.1
retriever, no recurrent module	52.5	42.0
retriever, no beam search ($B = 1$)	62.1	49.7
retriever, no link-based negatives	57.1	45.7
reader, no answer re-ranking	61.0	48.6
reader, no negative examples	51.0	40.4

Table 5: **Ablation study**: evaluating different variants of our model across F1 and EM on HotpotQA fullwiki.

linked from the top TF-IDF paragraphs. It then uses the same BERT model to select the paragraphs.
Entity-centric IR is our re-implementation of Godbole et al. (2019) that is related to Re-rank 2hop, but instead of simply selecting the top two paragraphs, they re-rank the possible combinations of the paragraphs that are liked to each other.
Cognitive Graph (Ding et al., 2019) that uses the provided prediction results of the state-of-the-art retrieval model on the HotpotQA development dataset.

Retrieval results Table 4 shows that our recurrent retriever yields significantly better retrieval scores, leading to almost 14 points P EM and 13.5 QA EM gain over the strongest method in Cognitive Graph. Exploring entity links from the initially retrieved documents help to retrieve the paragraphs with fewer lexical overlaps. Particularly, Entity-centric IR improves Re-rank 2hop across both P EM and AR by 8 points. On the other hand, comparing our method with Entity-centric IR shows the importance of learning to sequentially retrieve reasoning paths in the Wikipedia graph. Also note that our method even with $F = 20$ outperforms all the published QA EM scores in Table 1.

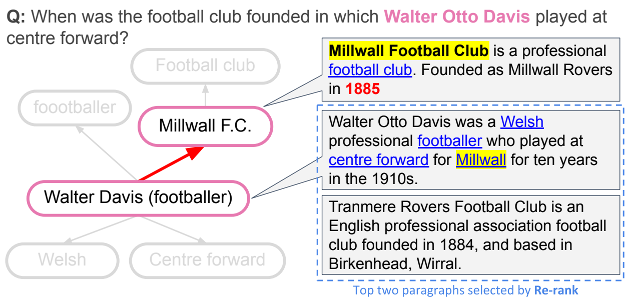
4.4 Analysis

Ablation study of our framework To study the effectiveness of our modeling choices, we compare the performance of variants of our framework on the HotpotQA fullwiki setting. We ablate the retriever with: 1) *No recurrent module*, removes the recurrence from our graph-based recurrent retriever, equivalent to the Entity-centric IR setting in Section 4.3. This method selects the path with the most highest probability path on the graph. 2) *No beam search*, that uses greedy search ($B = 1$) in our recurrent retriever. 3) *No link-based negative examples,* that trains the retriever model without adding hyperlink-based negative examples besides TF-IDF-based negative examples.

We ablate the reader model with: 1) *No answer re-ranking*, that removes $L_{\text{no\_answer}}$ from L_{read} and only takes the best reasoning path to extract the answer span. 2) *No negative examples*, that trains the model only with gold paragraphs in the HotpotQA distractor and does not use negative examples.

	Retriever	Reader
Avg. # of L	1.96	2.21
$L = 1$	539	403
$L = 2$	6639	5655
$L = 3$	227	1347

Table 6: The average length of the reasoning path selected by our retriever and reader for HotpotQA fullwiki, and the distribution of length of the reasoning paths selected by the two components.


Q: When was the football club founded in which Walter Otto Davis played at centre forward?
Football club
foootballer
Millwall F.C.
Walter Davis (footballer)
Welsh
Centre forward
Top two paragraphs selected by Re-rank

Millwall Football Club is a professional football club. Founded as Millwall Rovers in 1885

Walter Otto Davis was a Welsh professional footballer who played at centre forward for Millwall for ten years in the 1910s.

Tranmere Rovers Football Club is an English professional association football club founded in 1884, and based in Birkenhead, Wirral.

Figure 3: Reasoning examples by our model (red line) and Re-rank (the bottom two paragraphs). Highlighted text denotes a bridge entity, and blue-underlined text represents links.

Ablation results Table 5 shows that removing any of the listed components gives notable performance drop. The most critical component is our recurrent module, dropping the EM by 9 points. As shown in Figure 1, one or more documents to answer multi-hop open-domain questions relies on the entities mentioned in another paragraph. Therefore, without conditioning on the previous time steps, the model fails to retrieve the complete evidence. Training without hyperlink-based negative examples results in the second largest performance drop, indicating that the model can be easily distracted by reasoning paths without a correct answer. Replacing beam search with greedy search gives almost equivalent performance drops, which demonstrates that being aware of the graph structure is helpful to find the best reasoning path.

Performance drop by removing answer re-ranking indicates the importance of verifying the reasoning paths in our reader. Not using negative examples to train the reader degrades more than 10 points, due to the reader's over-confident predictions as discussed in Clark & Gardner (2018).

Analysis on reasoning path length Table 6 shows the average length of the reasoning paths selected by our retriever and reader, and also the number of the examples with reasoning paths whose length is $L = \{1, 2, 3\}$ on HotpotQA fullwiki. The table shows that our approach is adaptive to collect the required evidence. As discussed in Min et al. (2019a), we observe that some questions can be actually answered based on a single paragraph, and our model is likely too terminate after selecting single paragraph for such questions. In addition, the average length selected by our reader is notably larger than the one by our retriever. We expect that the retriever favors a shorter path, while the reader tends to select longer but more convincing multi-hop path.

Qualitative example of retrieved reasoning paths Figure 3 shows an example where our approach successfully retrieves the correct reasoning path and answers correctly, while Re-rank fails to retrieve the two paragraphs. The reasoning path is represented as a red arrow. The top two paragraphs next to the graph are the introductory paragraphs of the two entities on the reasoning path, and the paragraph at the bottom shows the wrong paragraph that Re-rank chooses. The "Millwall F.C." has fewer lexical overlaps and the bridge entity "Millwall" is not stated in the given question. Thus, Re-rank chooses a wrong paragraph with high lexical overlaps to the given question.

5 Conclusion

This paper introduces a new graph-based recurrent retrieval approach, which retrieves reasoning paths over the Wikipedia graph to answer multi-hop open-domain questions. Our retriever learns to directly retrieve evidence documents in the reasoning path given the history of previously retrieved documents. Subsequently, our reader ranks the reasoning paths, and it determines the final answer as the one extracted from the best reasoning path. Our experimental results significantly advance the state-of-the-art on HotpotQA by more than 10 points absolute gain on the fullwiki setting. Our approach also achieves comparable to the state-of-the-art performance on SQuAD Open, demonstrating the robustness of our method. In addition to the competitive performance, our method provides insights of the underlying entity relationships and is more directly interpretable. Future work involves end-to-end training of our graph-based recurrent retriever and reader for improving our current two-stage training.

REFERENCES

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *ACL*, 2017.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. In *ACL*, 2018.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Multi-step retriever-reader interaction for scalable open-domain question answering. In *ICLR*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2018.

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

Yair Feldman and Ran El-Yaniv. Multi-hop paragraph retrieval for open-domain question answering. In *ACL*, 2019.

Ameya Godbole, Dilip Kavarthapu, Rajarshi Das, Zhiyu Gong, Abhishek Singhal, Xiaoxiao Yu, Mo Guo, Tian Gao, Hamed Zamani, Manzil Zaheer, and Andrew McCallum. Multi-step entity-centric information retrieval for multi-hop question answering. In *EMNLP 2019 Workshop on Machine Reading for Question Answering*, 2019.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *ACL*, 2019.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *ICLR*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Bernhard Kratzwald and Stefan Feuerriegel. Adaptive document retrieval for deep question answering. In *EMNLP*, 2018.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. Ranking paragraphs for improving answer recall in open-domain question answering. In *EMNLP*, 2018.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *ACL*, 2019.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising distantly supervised open-domain question answering. In *ACL*, 2018.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. Efficient and robust question answering from minimal context over documents. In *ACL*, 2018.

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

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*, 2019b.

Yixin Nie, Songhe Wang, and Mohit Bansal. Revealing the importance of semantic retrieval for machine reading at scale. In *EMNLP*, 2019.

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

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *ACL*, pp. 157–163, 2017.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*, 2019.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. R^3: Reinforced ranker-reader for open-domain question answering. In *AAAI*, 2018a.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. Evidence aggregation for answer re-ranking in open-domain question answering. In *ICLR*, 2018b.

Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *EMNLP*, 2019.

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

Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *ICLR*, 2017.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with BERTserini. In *NAACL (Demonstrations)*, 2019.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

APPENDIX

A DETAILS ABOUT MODELING

A Normalized RNN We decompose Equation (4) as follows:

$$a_{t+1} = W_r[h_t; w_i] + b_r, \quad h_{t+1} = \frac{\alpha}{\|a_{t+1}\|} a_{t+1}, \tag{9}$$

where $W_r \in \mathbb{R}^{d \times 2d}$ is a weight matrix, $b_r \in \mathbb{R}^d$ is a bias vector, and $\alpha \in \mathbb{R}^1$ is a scalar parameter (initialized with 1.0). We set the global initial state a_1 to a parameterized vector $s \in \mathbb{R}^d$, and we also parameterize an [EOE] vector $w_{\text{[EOE]}} \in \mathbb{R}^d$ for the [EOE] symbol. The use of w_i for both the input and output layers is inspired by Inan et al. (2017); Press & Wolf (2017). In addition, we align the norm of $w_{\text{[EOE]}}$ with those of w_i, by applying layer normalization (Ba et al., 2016) of the last layer in BERT because $w_{\text{[EOE]}}$ is used along with the BERT outputs. Without the layer normalization, the $L2$-norms of w_i and $w_{\text{[EOE]}}$ can be quite different, and the model can easily discriminate between them by the difference of the norms.

Question-Paragraph Encoding Equation (2) shows that we compute each paragraph representation w_i conditioned by the question q. An alternative approach is separately encoding the paragraphs and the question as in Lee et al. (2019); Das et al. (2019); Seo et al. (2019), to directly retrieve paragraphs without any lexical matching retrieval engines; such a neural retriever suffers from compressing the necessary information into fixed-dimensional vectors, resulting in low performance on entity-centric questions (Lee et al., 2019). On the other hand, it has been shown that attention-based paragraph-question interactions improve the retrieval accuracy if the retrieval scale is tractable (Wang et al., 2018a; Lee et al., 2018). There is a trade-off between the scalability and the accuracy, and this work aims at striking the balance by jointly using the lexical matching retrieval and the graphs, followed by the rich question-paragraph encodings.

Yes-No Questions In the HotpotQA dataset, we need to handle yes-no questions as well as extracting anser spans from the paragraphs. We treat the two special types of the answers, yes and no, by extending the re-ranking model in Equation (6). More specifically, we extend the binary classification to a multi-class classification task, where the positive "answerable" class is decomposed into the following three classes: span, yes, and no. If the probability of "yes" or "no" is the largest among the three classes, our reader directly outputs the label as the answer, without any span extraction. Otherwise, our reader uses the span extraction model to output the answer.

Supporting fact prediction We adapt our recurrent retriever to the sub task of the supporting fact prediction in HotpotQA (Yang et al., 2018). The task is outputting sentences which support to anser the question. More specifically, such supporting sentences are annotated for the two ground-truth paragraphs in the training data. Since our framework outputs the most plausible reasoning path E along with the answer, we can add an additional step to select supporting facts (sentences) from the paragraphs in E. We train our recurrent retriever by using the training examples for the supporting fact prediction task, where the model parameters are not shared with those of our paragraph retriever. We replace the question-paragraph encoding in Equation (2) with question-sentence encoding for the task. We then maximize the probability of selecting the ground-truth sequence of the supporting fact sentences, while setting the other sentences as negative examples. At test time, we use the best reasoning paths from our paragraph retriever to finally output the supporting facts.

B Details about experiments

Detailed Dataset Statistics of HotptoQA fullwiki, distractor and SQuAD Open The HotpotQA training, development, and test datasets contain 90,564, 7,405 and 7,405 questions, respectively. We train our retriever and reader on the HotpotQA distractor training data, with augmented additional references and negative examples described in Section 3.1.1 and 3.2. For SQuAD Open, we use the original training dataset (78,713 questions) as our training data, and the original developmental dataset (10,570 questions) as our test dataset.

Deriving ground-truth reasoning paths Section 3.1.2 describes our training strategy for our recurrent retriever. To derive the ground-truth reasoning path g, we use the ground-truth evidence paragraphs associated with the questions in the training data for each dataset. For SQuAD, each training example has only single paragraph p, and thus it is trivial to derive g as $[p$, [EOE]$]$. For the multi-hop case, HotpotQA, we have two ground-truth paragraphs p_1, p_2 for each question. Assuming that p_2 includes the answer string, we set $g = [p_1, p_2,$ [EOE]$]$.

Training settings To use the pre-trained BERT models, we used the public code base, pytorch-transformers,[3] written in PyTorch.[4] For optimization, we used the code base's implementation of the Adam optimizer (Kingma & Ba, 2015), with a weight-decay coefficient of 0.01 for non-bias parameters. A warm-up strategy in the code base was also used, with a warm-up rate of 0.1. Most of the settings follow the default settings. To train our recurrent retriever, we set the learning rate to $3 \cdot 10^{-5}$, and the maximum number of the training epochs to three. The mini-batch size is four; a mini-batch example consists of a question with its corresponding paragraphs. To train our reader

[3]`https://github.com/huggingface/pytorch-transformers`.
[4]`https://pytorch.org/`.

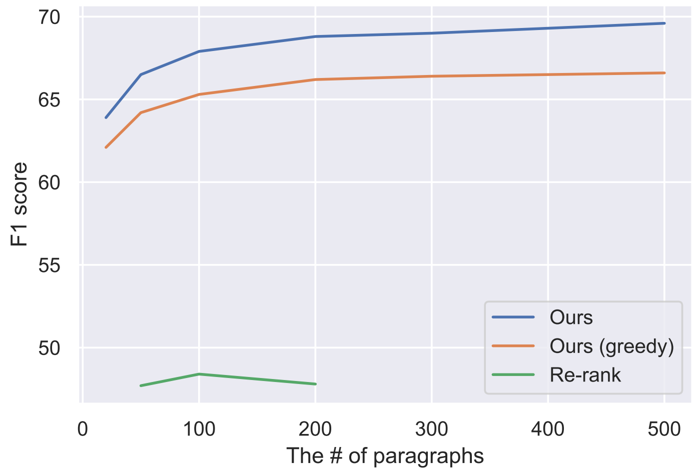


Figure 4: The effects of our graph-based recurrent retriever.

model, we set the learning rate to $3 \cdot 10^{-5}$, and the maximum number of training epochs to two. The mini-batch size is six; a mini-batch example consists of a question with its evidence paragraphs.

C ADDITIONAL RESULTS

C.1 ON THE ROBUSTNESS TO THE INCREASE OF THE PARAGRAPHS

As we discussed in 3.1.1, we could significantly reduce the searching space and thus could scale the number of initial retrieved candidates. Increasing the number of the initial retrieval often improves the recall among selected paragraphs. On the other hand, a large number of candidates paragraphs introduces additional noises and may distract models and eventually hurt the performance (Kratzwald & Feuerriegel, 2018). We compare the performance of three different approaches ((i) *ours*, (ii) *ours (greedy, without answer re-ranking)*, and (iii) *Re-rank*), increasing the number of the initial retrieval from 10 to 500 (For Re-rank, we compare the performance up to 200 paragraphs). The experimental results clearly show that our approach (graph-based recurrent retriever with answer re-ranking) has robustness towards the increase of the initial candidates paragraphs, and thus can constantly yield performance gains by increasing the number of the initial candidates. Ours (greedy) also shows performance improvements by increasing the initial candidates number from 10 to 300. However, after a certain number, the greedy approach stops improving the performance. Re-rank starts suffering from the noises caused by many distracting paragraphs included in the initial candidate paragraphs at $F = 200$.

C.2 MORE QUALITATIVE ANALYSIS ON THE REASONING PATH ON HOTPOTQA FULLWIKI

In this section, we conduct more qualitative analysis on the reasoning paths predicted by our model. Directly retrieving plausible reasoning paths and re-ranking the paths provide us interpretable insights into the underlying entity relationships used for multi-hop reasoning.

As shown in Table 6, our model flexibly selects more than two or only one paragraph for each question. To understand these behaviors, we conduct qualitative analysis on these “unexpected” reasoning path cases.

Reasoning path only with single paragraph First, we show an example, where our retriever selects single paragraph and terminates without selecting any additional paragraphs. Table 7 shows that, while originally this questions requires a system to read two paragraphs, **Before I Go to Sleep (film)** and **Nicole Kidman**, our retriever and reader eventually choose **Nicole Kidman** only. The second paragraph has a lot of lexical overlaps to the given question, and thus, a system may not need to read both of the paragraphs to answer. Min et al. (2019a) also observed that some of the questions do not necessarily require multi-hop reasoning, while HotpotQA is designed to require multi-hop reasoning (Yang et al., 2018). In that sense, we can say that our method automatically detects potentially single-hop questions.

Q: Before I Go to Sleep stars an Australian actress, producer and occasional what??

Before I Go to Sleep (film): Before I Go to Sleep is a 2014 mystery psychological thriller film written and directed by Rowan Joff and based on the 2011 novel of the same name by S. J. Watson. An international co-production between the United Kingdom, the United States, France, and Sweden, the film stars Nicole Kidman, Mark Strong, Colin Firth, and Anne-Marie Duff.

Nicole Kidman: Nicole Mary Kidman, is an Australian actress, producer and occasional singer. She is the recipient of several awards, including an Academy Award, two Primetime Emmy Awards, a BAFTA Award, three Golden Globe Awards, and the Silver Bear for Best Actress.

Annotated reasoning path **Before I Go to Sleep (film)** $\rightarrow$ **Nicole Kidman**
Predicted reasoning path: **Nicole Kidman**

Table 7: An example where our model retrieves a reasoning path with only one paragraph. The model expects that the question is answerable based on the last paragraph of the annotated reasoning path.

C.3 QUALITATIVE EXAMPLES OF HOTPOTQA DISTRACTOR

To understand how our retriever, especially the recurrent module, behaves, we conduct qualitative analysis based on the results on HotpotQA distractor. The distractor contains the two ground-truth paragraphs, and thus we can analyze the model behaviors in an oracle setting.

Table 8 is one example from the HotpotQA distractor setting to show the effectiveness of our graph-based recurrent retriever. In this example, **P1** and **P2** are the ground-truth paragraphs. At the first time step, the retriever expects that **P2** is unlikely to be the evidence, as the retriever is not aware of a bridge entity, "Pasek & Paul". If we simply adopt "top-2" strategy as discussed in Section 4.4, **P3** with the second highest probability would be selected, resulting in the wrong paragraph selection. In our framework, our retriever has context from the previous retrieval and thus, at the second time step, it chooses the correct paragraph, **P2**, lowering the probability of **P3**. At the third step, our model stops the prediction by outputting [EOS]. In 588 examples (7.9%) of the entire distractor development dataset, the paragraph selection by our graph-based recurrent retriever differs from the top-2 strategy, which demonstrates the effectiveness of conditioning multi-step retrieval.

We present another example, where only graph-based recurrent retrieval model succeeds finding the correct paragraph pairs. The second question in Table 8 shows that at the first time step our retriever successfully finds the first ground-truth paragraph, but did not pay attention to the other ground-truth paragraph at all, as the retriever is not aware of the bridge entity, "the Russian Civil War". Once it recognizes the first one, it finds the second one (**P2**). Like this, we can see how our model successfully learns to model relationships between paragraphs to answer the question.

Q: Which songwriting duo composed music for ”La La Land”, and created lyrics for ”A Christmas Story: The Musica”?			
P1: A Christmas Story: The Musical is a musical version of the film ”A Christmas Story ... The musical has music and lyrics written by Pasek & Paul and the book by Joseph Robinette.	0.98 ✓	0.00	0.00
P2: Benj Pasek and Justin Paul, known together as Pasek and Paul, are an American songwriting duo and composing team for musical theater, films, and television. ... they won both the Golden Globe and Academy Award for Best Original Song for the song ”City of Stars”.	0.08	0.89 ✓	0.00
P3: La La Land” is a song recorded by American singer Demi Lovato. It was written by Lovato, Joe Jonas, Nick Jonas and Kevin Jonas and produced by the Jonas Brothers alongside John Fields, for Lovato’s debut studio album, ”Dońt Forget” (2008).	0.12	0.00	0.00
Q: Alexander Kerensky was defeated and destroyed by the Bolsheviks in the course of a civil war that ended when ?			
P1: The Socialist Revolutionary Party, or Party of Socialists-Revolutionaries sery”) was a major political party in early 20th century Russia and a key player in the Russian Revolution. ... The anti-Bolshevik faction of this party, known as the Right SRs, which remained loyal to the Provisional Government leader Alexander Kerensky was defeated and destroyed by the Bolsheviks in the course of the Russian Civil War and subsequent persecution.	0.95 ✓	0.00	0.00
P2: The Russian Civil War (November 1917 October 1922) was a multi-party war in the former Russian Empire immediately after the Russian Revolutions of 1917, as many factions vied to determine Russiaś political future.	0.00	0.87 ✓	0.00
P3:Alexander Fyodorovich Kerensky was a Russian lawyer and key political figure in the Russian Revolution of 1917.	0.08	0.09	0.00

Table 8: An example from HotpotQA distractor. Highlighted text show the bridge entities for multi-hop, and also the words in red denotes the predicted answer.