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 show state-of-the-art results in three open-domain QA datasets, showcasing the effectiveness and robustness of our method. Notably, our method achieves significant improvement in HotpotQA, outperforming the previous best model by more than 14 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 an answer span is extracted by a neural machine reading comprehension model. Despite the success of these pipeline methods in single-hop QA, whose questions can be answered based on a single paragraph, they often fail to retrieve the required evidence for answering multi-hop questions, e.g., the question in Figure 1. Multi-hop QA (Yang et al., 2018) 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 Wikipedia paragraph graph in advance using Wikipedia hyperlinks and document structures to model the relationships between paragraphs. Our retriever trains a recurrent neural network to score reasoning paths in this graph

[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 TF-IDF retrievers due to little lexical overlap to the given question.

by maximizing the likelihood of 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 full wiki and HotpotQA distractor settings (Yang et al., 2018), outperforming the previous state-of-the-art methods by more than 14 points absolute gain on the full wiki setting. We also evaluate our approach on SQuAD Open (Chen et al., 2017) and Natural Questions Open (Lee et al., 2019) without changing any architectural designs, achieving better or comparable to the state of the art, which suggests that our 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 (Ding et al., 2019; Qi et al., 2019), 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, instead of requiring a reader to read and find reasoning paths. While the previous approach requires a system to encode all possible nodes, our beam search decoding process only encodes the nodes on the reasoning paths, which significantly reduces the computational costs. Concurrent with our work, Entity-centric IR (Godbole et al., 2019) combines retrieval with entity linking for multi-hop retrieval. Unlike our method, this method does not learn to retrieve reasoning paths sequentially and does not study interplay between retriever and reader. We empirically show large 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. Most recently, Qi et al. (2019) introduce GoldEn Retriever, which reads and generates search queries for two steps to search documents for HotpotQA full wiki. These methods do not use the graph structure of the documents during the iterative retrieval process. Moreover, all of these multi-step retrieval methods do not accommodate arbitrary steps of reasoning and the termination condition is hard-coded. In contrast, our method leverages the Wikipedia graph to retrieve documents that are

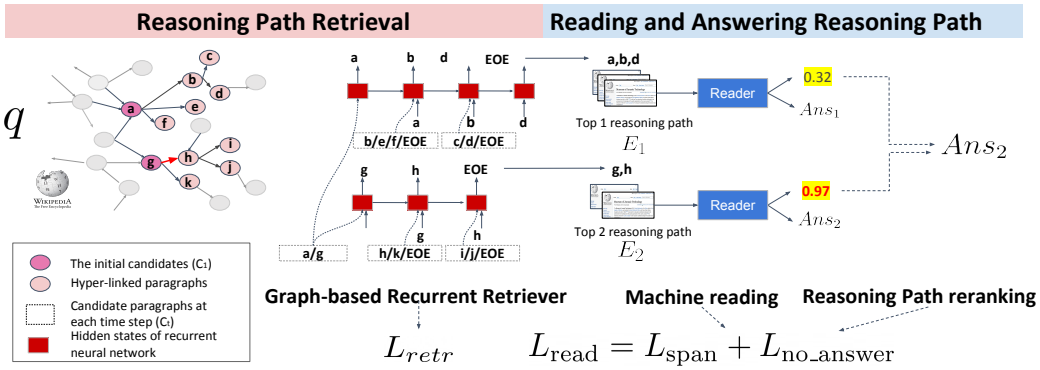

Figure 2: Overview of our framework.

lexically or semantically distant to questions, and is adaptive to any reasoning path lengths, which leads to significant improvement over the previous work in HotpotQA and SQuAD Open.

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 to retrieve a set of reasoning paths to narrow down the search space for the reader, for robust pipeline process. Figure 2 sketches the overview of our QA model.

We are given large-scale documents containing millions of paragraphs p_i (e.g., Wikipedia) in total, a question q, and its answer candidate string a. Let $E = [p_i, \ldots, p_k]$ denote a set of multiple paragraphs necessary to answer the question (we call it a "reasoning path"). We formulate the open-domain 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. However, one of them is likely to be retrieved, and entity mentions in it and the question often entail another paragraph by multi-hop reasoning (e.g., Figure 1). To perform such multi-hop reasoning, we first constructs a graph of Wikipedia documents, covering all of the paragraphs existing in Wikipedia. Each node of the Wikipedia graph $\mathcal{G}$ represents single paragraph p_i.

Constructing the Wikipedia graph Hyperlinks are commonly used to construct relationships between articles on the Web. Hyperlinks are usually maintained by article writers, and are thus useful knowledge resources. Wikipedia consists of its internal hyperlinks to connect articles in Wikipedia by following entity links. We use the hyperlinks to construct the direct edges in $\mathcal{G}$. 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. This graph is constructed offline and is reused throughout training and inference for any question.

3.1.1 RECURRENT RETRIEVER

Our method learns to retrieve reasoning paths 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. The initial hidden state h_1 is independent of any questions or paragraphs, and based on a parameterized vector. We use BERT's [CLS] token representation (Devlin et al., 2019) 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 A.1. The next candidate set $\mathbf{C}_{t+1}$ is constructed to include paragraphs that are linked from the selected paragraph p_i in the graph.

Beam search for candidate paragraphs It is computationally expensive to compute Equation (2) over millions of the possible paragraphs. Moreover, a fully trainable retriever often performs poorly for entity-centric questions such as SQuAD, since representing specific entity information in a dense vector space is challenging (Lee et al., 2019). 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. $g_r = [p_r, p_1, \ldots, p_{|g|}]$ 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 prevent our

retriever from being distracted by reasoning paths without correct answer spans. 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., 2019), and (2) *reasoning path 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., 2019), where the input is the concatenation of the question text and the text of the all paragraphs in E, which is shown to be effective in multi-hop QA (Wang et al., 2019a). We share the same model for the re-ranking model, and use the BERT's [CLS] representation to estimate the probability of selecting E to answer the question:

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

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}:

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

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. (2019).

Training examples To train the multi-task reader, we use the ground-truth evidence paragraphs used for training our retriever. It is known to be effective in open-domain QA to use distantly supervised examples, which are not originally associated with the questions but include expected answer strings (Chen et al., 2017; Wang et al., 2018a; Hu et al., 2019; Min et al., 2019a). These distantly supervised examples are also effective to simulate the inference time process. Therefore, we combine distantly supervised examples extracted from our retriever training data with the original supervised examples. Following the procedures in Chen et al. (2017), we add up to one distantly supervised example for each supervised example. We set the answer span as the string that matches a and appears first.

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:

$$L_{\text{read}} = L_{\text{span}} + L_{\text{no_answer}} = (-\log P^{start}_{y^{start}} - \log P^{end}_{y^{end}}) - \log P^r, \tag{8}$$

where y^{start}_q and y^{end}_q 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 three open-domain Wikipedia-sourced datasets: HotpotQA (fullwiki, distractor), SQuAD Open and Natural Questions Open. More details about our experimental setup can be found in Appendix B.

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 *full wiki* setting. Two evidence paragraphs are associated with each question in the train data. Our primary target is the full wiki setting due to its open-domain scenario, and we use the distractor setting to evaluate how well our method works in a closed scenario where the two evidence paragraphs are always included. The dataset also provides annotations to evaluate the prediction of supporting sentences, and we adapt our retriever to the supporting fact prediction. Note that this additional task is specific to HotpotQA. The subtask is performed after finalizing the reasoning path and the answer for each question, and hence this additional task does not affect the QA accuracy. More details are described in Appendix A.3.

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 associated with each question in the training data.

Natural Questions Open Natural Questions Open (Lee et al., 2019; Min et al., 2019a) is composed of questions from the original Natural Questions dataset (Kwiatkowski et al., 2019).[2] The questions are created from Google Search queries independently from the existing articles.

Metrics For HotpotQA and SQuAD Open, 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. We report EM score on Natural Questions Open, following previous work (Lee et al., 2019; Min et al., 2019a). To evaluate the retrieval accuracy for the complex multi-hop reasoning, we use the following metrics: *Answer Recall (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 if at least one of the ground-truth paragraphs is included among the retrieved paragraphs, and *Paragraph Exact Match (P EM)*, which evaluates if both of the ground-truth paragraphs for multi-hop reasoning are included among the retrieved paragraphs

Evidence Corpus and the Wikipedia graph In this work, we use Wikipedia as the evidence corpus and do not use other data such as Google search snippets or external structured knowledge bases. We use the several versions of Wikipedia dumps for the three datasets (See Appendix B.4). 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, following previous studies (Feldman & El-Yaniv, 2019; Ding et al., 2019; Nie et al., 2019).

[2]We use train/dev/test splits provided by Min et al. (2019a), which can be downloaded from `https://drive.google.com/file/d/1qsN5Oyi_OtT2LyaFZFH26vT8Sqjb89-s/view`.

Models	full wiki				distractor			
	QA		SP		QA		SP	
	F1	EM	F1	EM	F1	EM	F1	EM
Semantic Retrieval (Nie et al., 2019)	58.8	46.5	71.5	39.9	–	–	–	–
GoldEn Retriever (Qi et al., 2019)	49.8	–	64.6	–	–	–	–	–
Cognitive Graph (Ding et al., 2019)	49.4	37.6	58.5	23.1	–	–	–	–
DecompRC (Min et al., 2019c)	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
Transformer-XH (Anonymous, 2020)	62.3	49.8	71.6	42.2	–	–	–	–
Ours (Reader: BERT wwm)	**73.3**	**60.5**	**76.1**	**49.3**	**81.2**	**68.0**	**85.2**	58.6
Ours (Reader: BERT base)	65.8	52.7	75.0	47.9	73.3	59.4	84.6	57.4

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

Models (*: anonymous)	QA		SP	
	F1	EM	F1	EM
Semantic Retrieval	57.3	45.3	70.8	38.7
GoldEn Retriever	48.6	37.9	64.2	30.7
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
HGN*♣	69.2	56.7	**76.4**	**50.0**
MIR+EPS+BERT*♣	64.8	52.9	72.0	42.8
Transformer-XH*	60.8	49.0	70.0	41.7
Ours	**73.0**	**60.0**	**76.4**	49.1

Table 2: **HopotQA full wiki test set results**: results on the HotpotQA full wiki setting on the hidden test set on the official leaderboard on November 6, 2019. ♣ appear after September 25.

Models	F1	EM
DrQA (Chen et al., 2017)	–	29.3
R^3 (Wang et al., 2018a)	37.5	29.1
Paragraph Ranker (Lee et al., 2018)	–	30.2
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
Weaver (Raison et al., 2018)	–	42.3
RE^3 (Hu et al., 2019)	50.2	41.9
multi-passaege (Wang et al., 2019b)	60.9	53.0
BM25+BERT (Lee et al., 2019)	–	33.2
ORQA (Lee et al., 2019)	–	20.2
Ours	**63.8**	**56.5**

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

Implementation details We use the pre-trained BERT models (Devlin et al., 2019) 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. We tuned the most important hyper-parameters, F, the number of the initial TF-IDF-based paragraphs, and B, the beam size, by mainly using the HotpotQA development set (the effects of increasing F are shown in Figure 5 in Appendix C.3 along with the results with $B = 1$). If not specified, we set $B = 8$ for all the datasets, $F = 500$ for HotpotQA full wiki and SQuAD Open, and $F = 100$ for Natural Questions Open.

4.2 Overall Results

Table 1 compares our method with previous published methods on the HotpotQA development set. Our method significantly outperforms all the previous results across the evaluation metrics under both the full wiki and distractor settings. Notably, our method achieves 14.5 F1 and 14.0 EM gains compared to state-of-the-art Semantic Retrieval (Nie et al., 2019) and more than 11.0 F1 gains over the concurrent submission, Transformer-XH (Anonymous, 2020) on full wiki. We can see that our method, even with the BERT base configuration for our reader, significantly outperforms all the previous QA scores. Moreover, our method shows significant improvement in predicting supporting facts in the full wiki setting. We compare the performance of our approach to other models on

Models	EM	
	Dev	Test
ORQA (Lee et al., 2019)	31.3	**33.3**
Hard EM (Min et al., 2019a)	28.8	28.1
BERT + BM 25 (Lee et al., 2019)	24.8	26.5
Ours	**31.7**	32.6

Table 4: **Natural Questions Open results**: we report EM scores on the test and development set of Natural Questions Open, following previous work.

Models	AR	PR	P EM	EM
Ours ($F = 20$)	87.0	93.3	72.7	56.8
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
Semantic Retrieval	77.9	93.2	63.9	46.5

Table 5: **Retrieval evaluation**: Comparing our retrieval method with other methods across Answer Recall, Paragraph Recall, Paragraph EM, and QA EM metrics.

the HotpotQA fullwiki official hidden test set in Table 2. We outperform all the published and unpublished models including work later than ours (HGN and MIR+EPS+BERT) by large margins.

Table 3 and Table 4 show that our framework achieves state-of-the-art performance on the other two QA datasets as well. On SQuAD Open, our model outperforms the concurrent state-of-the-art model (Wang et al., 2019b) by 2.9 F1 and 3.5 EM scores. Due to the fewer lexical overlap between questions and paragraphs on Natural Questions, pipelined approaches using term-based retrievers often face difficulties finding associated articles. Actually, our Natural Questions scores did not change by further increasing F from 100. Nevertheless, our approach matches the performance of the best end-to-end retriever (ORQA), outperforming all the other pipelined approaches (BERT+BM25 and Hard EM) by a large margin as shown in Table 4. In addition to its competitive performance, our retriever can be handled on a single GPU machine, while a fully end-to-end retriever such as ORQA requires industry-scale computational resources for training (Seo et al., 2019).

4.3 Performance of Reasoning Path Retrieval

We compare our retriever with competitive retrieval methods for open-domain QA. We set F to 20.

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 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 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 linked to each other.
Cognitive Graph (Ding et al., 2019) that uses the provided prediction results on the HotpotQA development dataset.
Semantic Retrieval (Nie 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 5 shows that our recurrent retriever yields 8.8 P EM and 9.1 AR, leading to 10.3 QA EM improvements over Semantic Retrieval. The significant improvement from Re-rank2hop to Entity-centric IR demonstrates that exploring entity-links from the initially retrieved documents helps to retrieve the paragraphs with fewer lexical overlaps. On the other hand, comparing our retriever with Entity-centric IR and Semantic Retrieval shows the importance of learning to sequentially retrieve reasoning paths in the Wikipedia graph. It should be noted that our method with $F = 20$ outperforms all the QA EM scores in Table 1.

4.4 Analysis

Here, we conduct detailed analysis of our framework on the HotpotQA full wiki development set.

Settings ($F = 100$)	F1	EM
full	72.4	59.5
retriever, no recurrent module	52.5	42.1
retriever, no beam search	68.7	56.2
retriever, no link-based negatives	64.1	52.6
reader, no reasoning path re-ranking	70.1	57.4
reader, no negative examples	53.7	43.3

Table 6: **Ablation study**: evaluating different variants of our model on HotpotQA full wiki.

Settings ($F = 100$)	F1	EM
with hyperlinks	72.4	59.5
with entity linking system	70.1	57.3

Table 7: **Performance with different link structures**: comparing our results on the Hotpot QA full wiki development set when we use an off-the-shelf entity linking system instead of the Wikipedia hyperlinks.

Settings ($F = 100$)		F1	EM
Adaptive retrieval		72.4	59.5
L-step retrieval	$L = 1$	45.8	35.5
	$L = 2$	71.4	58.5
	$L = 3$	70.1	57.7
	$L = 4$	66.3	53.9

Table 8: **Performance with different reasoning path length**: comparing the performance with different path length on HotpotQA full wiki. L-step retrieval sets the number of the reasoning steps to a fixed number.

($F = 100$)	Retriever	Reader	EM
Avg. # of L	1.96	2.21	with L
1	539	403	31.2
2	6,639	5,655	60.0
3	227	1,347	63.0

Table 9: **Statistics of the reasoning paths**: the average length and the distribution of length of the reasoning paths selected by our retriever and reader for HotpotQA full wiki. Avg. EM represents QA EM performance.

Ablation study of our framework To study the effectiveness of our modeling choices, we compare the performance of variants of our framework. We ablate the retriever with: 1) *No recurrent module*, that removes the recurrence from our retriever. 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 reasoning path re-ranking*, that only takes the best reasoning path from the retriever and extracts the answer span from this path, and 2) *No negative examples*, that trains the model only with gold paragraphs, removing $L_{\text{no_answer}}$ from L_{read}. During inference, our system reads all the reasoning paths and chooses an answer with the highest answer probability.

Ablation results Table 6 shows that removing any of the listed components gives notable performance drop. The most critical component is our recurrent module, dropping the EM by 17.4 points. As shown in Figure 1, multi-step retrieval often relies on the information 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 and the importance of training models with negative sampling. Replacing beam search with greedy search gives about 4 points EM performance drops, which demonstrates that being aware of the graph structure is helpful to find the best reasoning path.

Performance drop by removing the reasoning path re-ranking indicates the importance of verifying the reasoning paths in our reader. Not using negative examples to train the reader degrades EM more than 16 points, due to the over-confident predictions as discussed in Clark & Gardner (2018).

The performance with an off-the-shelf entity linking system Although the existence of the hyperlinks is not special on the Web, one question is how well our method works without the Wikipedia hyperlinks. Godbole et al. (2019) propose an approach with entity linking system to build entity graphs, and we also evaluate our method on the development set of HotpotQA full wiki with an off-the-shelf entity linking system (Ferragina & Scaiella, 2011) to construct the document graph in our method. More details about our experimental setup can be found in Appendix B.6. Table 7 shows that our approach with the entity linking system shows only 2.3 F1 and 2.2 EM lower scores than those with the hyperlinks, still achieving the state of the art. This suggests that our approach is not restricted to the existence of the hyperlink information.

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.

Figure 4: Reasoning examples by our retriever (the bottom paragraph) and our reader (the red line). Highlighted text denotes a bridge entity, and blue-underlined text represents links.

The performance of different path length The existing iterative retrieval methods fix the number of reasoning steps (Qi et al., 2019; Das et al., 2019; Godbole et al., 2019; Feldman & El-Yaniv, 2019), while our approach accommodates arbitrary steps of reasoning. We also evaluate our method by fixing the length of the reasoning path ($L = \{1, 2, 3, 4\}$). Table 8 shows that out adaptive retrieval performs the best, although the length of all the annotated reasoning paths in HotpotQA is two.

The effectiveness of the interplay between retriever and reader Table 6 shows that the interplay between our retriever and reader models is effective. To understand this, we investigate the length of reasoning path selected by our retriever and reader, and their final QA performance. Table 9 shows that the average length selected by our reader is notably longer than that by our retriever. Table 9 also presents the EM scores averaged over the questions with certain length of reasoning paths ($L = \{1, 2, 3\}$). We observe that our framework performs the best when it selects the reasoning paths with the length of 3, showing 76.3 F1 and 63.0 EM scores. Based on these observations, we expect that the retriever favors a shorter path, while the reader tends to select a longer but more convincing multi-hop reasoning path to derive an answer string. As discussed in Min et al. (2019b), we observe that some questions are answerable based on a single paragraph, and our model is likely to terminate after selecting single paragraph for such questions.

Qualitative example of retrieved reasoning paths Finally, we show two examples from HotpotQA full wiki, and more qualitative examples are presented in Appendix C.4. Figure 3 shows an example where our approach successfully retrieves the correct reasoning path and answers correctly, while Re-rank fails. 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 selected by Re-rank. 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. In Figure 4, we compare the reasoning paths ranked highest by our retriever and reader. Although the gold path is included among the top 8 paths selected by the beam search, our retriever selects a wrong paragraph as the best reasoning path. By re-ranking the reasoning paths, the reader eventually selects the correct reasoning path ("2017-18 Wigan Athletic F.C. season" $\rightarrow$ "EFL Cup"). This example shows the effectiveness of the strong interplay of our retriever and reader.

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 14 points absolute gain on the full wiki setting. Our approach also achieves the state-of-the-art performance on SQuAD Open and Natural Questions Open without any architectural changes, demonstrating the robustness of our method. In addition to the competitive performance, our method provides insights into 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

Anonymous. Transformer-XH: Multi-hop question answering with extra hop attention. In *Submitted to ICLR*, 2020. URL `https://openreview.net/forum?id=r1eIiCNYwS`. under review.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv: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*, 2019.

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.

Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *IEEE software*, 29(1):70–75, 2011.

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 *Proceedings of the 2nd 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.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Rhinehart, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, et al. Natural questions: a benchmark for question answering research. *TACL*, 2019.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019.

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

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. A discrete hard em approach for weakly supervised question answering. In *EMNLP*, 2019a.

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

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

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:1901.04085*, 2019.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *WMT*, 2018.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *EACL*, 2017.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. Answering complex open-domain questions through iterative query generation. In *EMNLP*, 2019.

Martin Raison, Pierre-Emmanuel Mazaré, Rajarshi Das, and Antoine Bordes. Weaver: Deep co-encoding of questions and documents for machine reading. *arXiv:1804.10490*, 2018.

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.

Haoyu Wang, Mo Yu, Xiaoxiao Guo, Rajarshi Das, Wenhan Xiong, and Tian Gao. Do multi-hop readers dream of reasoning chains? In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, 2019a.

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*, 2019b.

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.1 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 $w_{[\mathrm{EOE}]} \in \mathbb{R}^d$ 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 $w_{[\mathrm{EOE}]}$ with those of w_i, by applying layer normalization (Ba et al., 2016) of the last layer in BERT because $w_{[\mathrm{EOE}]}$ is used along with the BERT outputs. Without the layer normalization, the $L2$-norms of w_i and $w_{[\mathrm{EOE}]}$ can be quite different, and the model can easily discriminate between them by the difference of the norms.

A.2 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.

A.3 MODELING DETALIS OF OUR READER COMPONENT

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 subtask 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-answer-sentence encoding for the task, where a question string is concatenated with its answer string. The answer string is the ground-truth one during the training time. 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 path and its predicted answer string from our retriever and reader models to finally output the supporting facts for each question.

B DETAILS ABOUT EXPERIMENTS

B.1 DATASET STATISTICS OF HOTPOTQA, SQUAD OPEN AND NATURAL QUESTIONS 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. For Natural Questions Open, we follow dataset splits provided by Min et al. (2019a), and training, development and test datasets contain 79,168, 8,757 and 3,610, respectively. For SQuAD Open and Natural Questions Open, we train our reader on the original examples with augmented additional negative examples and distantly supervised examples.

B.2 DERIVING GROUND-TRUTH REASONING PATHS

Section 3.1.2 describes our training strategy for our recurrent retriever. We apply the data augmentation technique to HotpotQA and Natural Questions to consider multi-hop reasoning. 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 and Natural Questions Open, each training example has only single paragraph p, and thus it is trivial to derive g as $[p, [\mathrm{EOE}]]$. 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 $g = [p_1, p_2, [\mathrm{EOE}]]$.

B.3 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 model, we set the learning rate to $3 \cdot 10^{-5}$, and the maximum number of training epochs to two. Empirically we observe better performance with a larger batch size as discussed in previous work (Liu et al., 2019; Ott et al., 2018), and thus we set the mini-batch size to 120. A mini-batch example consists of a question with its evidence paragraphs. We will release our code to follow our experiments.

B.4 THE WIKIPEDIA DUMPS FOR EACH DATASET

For HotpotQA, we use the pre-processed Wikipedia dump from October, 2017, provided by HotpotQA authors[5]. For Natural Questions Open, we use the English Wikipedia dump from December 20, 2018, following previous work (Lee et al., 2019; Min et al., 2019a). For SQuAD Open, we use the Wikipedia dump provided by Chen et al. (2017).

Although using the same dump for different Open-domain QA datasets is a common practice for open-domain QA studies (Chen et al., 2017; Wang et al., 2018a; Lee et al., 2018), this potentially causes performance deterioration due to the temporal inconsistency. Some of the open-domain QA

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

[4] `https://pytorch.org/`.

[5] `https://hotpotqa.github.io/wiki-readme.html`.

benchmarks (e.g., SQuAD Open, Natural Questions Open) are created by discarding originally associated paragraphs from machine reading comprehension datasets. Due to the Wikipedia's constantly evolving nature, however, using Wikipedia dumps from different period may affect the overall results. For instances, a question from SQuAD Open test set, "Where does Kenya rank on the CPI scale?" is originally paired with a paragraph from the article of Kenya, and based on single sentence "Kenya ranks low on Transparency International's Corruption Perception Index (CPI)" from the paragraph, a competitive machine reading comprehension model can easily extract the answer span, "low". However, this sentence has been rewritten as "Kenya has a high degree of corruption according to Transparency International's Corruption Perception Index (CPI)" in the current article[6] and more recent dumps.

Although Wikipedia is a useful knowledge source for open-domain QA research, its rapidly evolving nature should be considered more carefully for the reproducibility. We will make all of the data including pre-processed Wikipedia articles for each experiment available for future research.

B.5 Details about Initial Candidates C_1 selection

To retrieve the initial candidates C_1 for each question, we use a TF-IDF based retriever with the bi-gram hashing (Chen et al., 2017). For HotpotQA full wiki, we retrieve top 500 introductory paragraphs from the all introductory paragraph only corpus. For SQuAD Open and Natural Questions Open, we first retrieve 50 Wikipedia articles through the same TF-IDF retriever, and further run another TF-IDF-based paragraph selections Clark & Gardner (2018) to retrieve 500 paragraphs in total.

B.6 Details about entity linking experiment

We experiment with a variant of our approach, where we incorporate an entity linking system with our framework, in place of the Wikipedia hyperlinks. In this experiment, we first retrieve seed paragraphs using TF-IDF ($F = 100$), and run an off-the-shelf entity linker (TagMe by Ferragina & Scaiella (2011)) over the paragraphs. If the entity linker detects some entities, we retrieve their corresponding Wikipedia articles, and add edges from the seed paragraphs to the entity-linked paragraphs. Once we build the graph, then we re-run all of the experiments while the other components are exactly the same. We use the TagMe official Python wrapper.[7]

C Additional Results

C.1 Upper-bound of our retrieval module

For scalability and computational efficiency, we bootstrap our retrieval module with TF-IDF retrieval; we first retrieval F paragraphs using TF-IDF with the method described in Section B.5 and initialize C_1 with these TF-IDF paragraphs. Although we expand our candidate paragraphs at each time step using the Wikipedia graph, if our method failed to retrieve paragraphs a few-hops away from the answer paragraphs, it is likely to fail to reach the answer paragraphs. To estimate the paragraph EM upper-bound, we have checked if two gold paragraphs are included in the top 20 TF-IDF paragraphs and their hyperlinked paragraphs in the HotpotQA full wiki setting. We found that for 75.4% of the questions, all of the gold paragraphs are included in the collections of the TF-IDF paragraphs and the hyperlinked paragraphs. Also, it should be noted when we only consider the TF-IDF retrieval results, the upper-bound drops to 35.1%, which suggests that the TF-IDF-based retrieval cannot effectively discover the paragraphs multi-hop away due to the few lexical overlap. When we increase the number of F to 100 and 500, the upper-bound reaches 84.1% and 89.2%, respectively.

C.2 Per-Category Question Answering and Retrieval Performance

In HotpotQA, there are two types of questions, *bridge* and *comparison*. While *comparison* type questions explicitly mention the two entities related to the given questions, in bridge-type questions,

[6] `https://en.wikipedia.org/wiki/Kenya` on October 25, 2019

[7] `https://github.com/marcocor/tagme-python`

Models	Total (7,405)			Bridge (5,918)			Comp (1,487)		
	PR	P EM	EM	PR	P EM	EM	PR	P EM	EM
Ours	**94.3**	**75.7**	**60.5**	**93.9**	**73.7**	**57.8**	98.7	83.5	**70.5**
Cognitive Graph (Ding et al., 2019)	87.6	57.8	37.5	84.8	51.8	36.1	98.6	81.6	53.7
Semantic Retrieval (Nie et al., 2019)	93.2	63.9	46.5	91.6	57.2	42.7	**99.7**	**90.6**	61.7

Table 10: **Retrieval evaluation**: Comparing our retrieval method with other methods across Answer Recall, Paragraph Recall, Paragraph EM, and QA EM metrics.

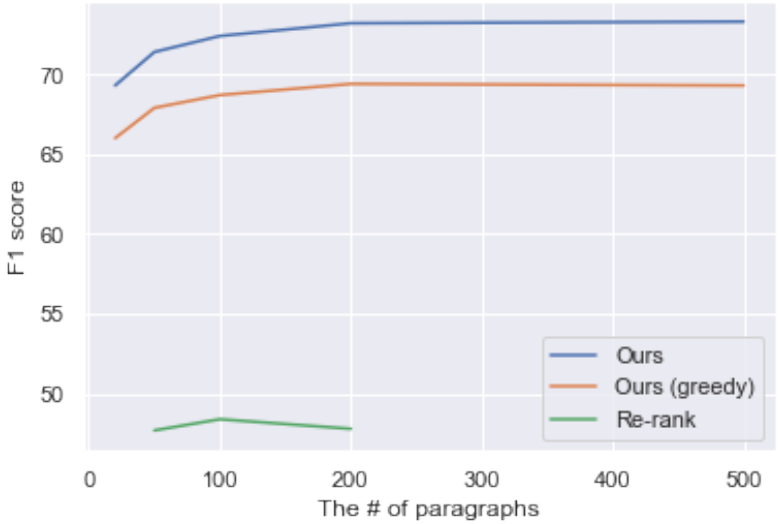

Figure 5: **The robustness to the increase of the F.** We compare the F1 scores of our model, our model without a beam search and Re-rank with different number of F.

the bridge entities are rarely explicitly stated. This makes it hard for a retrieval system to discover the paragraphs entailed by the bridge entities.

We evaluate the question answering and paragraph retrieval performance for each of two question types. We compare the PR, P EM and QA EM for each category with two state-of-the-art models, Cognitive Graph (Ding et al., 2019) and Semantic Retrieval (Nie et al., 2019). Here, we set our initial TF-IDF number to 500. Table 10 shows that our retriever yields 16.5 P EM gain and 15.1 EM gain over Semantic Retrieval for the challenging bridge type questions. For comparison type questions, our method achieve almost 10 point higher QA EM than Semantic Retrieval. We observed that some of the comparison type questions can be answered based on single paragraph, and thus our model select only one paragraphs for some of these comparison type questions, resulting in lower P EM scores on comparison type questions. We show several examples of the questions where we can answer based on single paragraph in Section C.3.

C.3 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 reasoning path 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). Figure 5 clearly shows that our approach (graph-based recurrent retriever with reasoning path 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 per-

formance improvements by increasing the initial candidates number from 10 to 200. 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.4 More qualitative analysis on the reasoning path on HotpotQA full wiki

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 9, our model flexibly selects more than two or only one paragraph for each question. To understand these behaviors, we conduct qualitative analysis on these examples whose reasoning paths are shorter or longer than the original gold reasoning paths.

Reasoning path only with single paragraph First, we show two examples (one is a bridge type question and the other is a comparison type question), where our retriever selects single paragraph and terminates without selecting any additional paragraphs.

The bridge type question in Table 11 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.

The comparison type question in Table 11 also shows that even comparison type questions do not always require two paragraphs to answer the questions, and our model only selects one paragraph necessary to answer the given question. In this example, the question has large lexical overlap with one of the ground-truth paragraph (**The Bears and I**), resulting in allowing a model to answer the question based on the single paragraph.

Min et al. (2019b) 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.

Reasoning path with three paragraphs All of the HotpotQA examples are authored by annotators who are shown two relevant paragraphs, and thus, originally the length of ground-truth reasoning paths is always two. On the other hand, as our model accommodate arbitrary steps of reasoning, it often selects reasoning paths longer than the original annotations as shown in Table 9. *When a model selects longer reasoning path for a HotpotQA question, does it contain paragraphs that provide additional evidence?* We show an example in Table 12, so as to answer this question. Our model selects additional paragraph, **Blue Jeans (Lana Del Rey song)** at the first step, and then selects the two annotated gold paragraphs. This first paragraph is strongly relevant to the given question, but does not contain an answer. This additional evidence might help the reader to find the correct bridge entity ("Back to December").

C.5 Qualitative analysis on the reasoning path on Natural Questions Open

We present one example where our model finds multi-hop reasoning path effectively in Natural Questions Open (development set) in Table 13. The question "who sang the original version of killing me so" has relatively fewer lexical overlap bith the originally annotated paragraph (**Killing Me Softly with His Song (V)** in Table 13). Moreover, there are several entities named with "killing me sogtly" in Wikipedia, because many artists cover the song. To answer this question correctly, our retriever first selects **Roberta Flack (I)**, and then hops to the originally annotated paragraph, **Killing Me Softly with His Song (V)**. Our reader further verifies this reasoning path and extracts the correct answer from **Killing Me Softly with His Song (V)**. This example shows that even without gold reasoning paths annotations, our model trained on augmented examples could learn to retrieve multi-hop reasoning path from the entire Wikipedia.

Q [bridge]: 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**
Q [comparison]: In between The Bears and I and Oceans which was released on July 31, 1974, by Buena Vista Distribution?
The Bears and I: The Bears and I is a 1974 American drama film directed by Bernard McEveety and written by John Whedon. The film stars Patrick Wayne, Chief Dan George, Andrew Duggan, Michael Ansara and Robert Pine. *The film was released on July 31, 1974, by Buena Vista Distribution.*
Oceans (film): Oceans is a 2009 French nature documentary film directed, produced, co-written, and narrated by Jacques Perrin, with Jacques Cluzaud as co-director.
Annotated reasoning path: **The Bears and I**, **Oceans (film)** Predicted reasoning path: **The Bears and I**

Table 11: Two examples of the questions that our model retrieves a reasoning path with only one paragraph. We partly remove sentences irrelevant to the questions.

C.6 Qualitative analysis on the reasoning path on HotpotQA distractor

Although the main focus on this paper in open-domain QA, where we need to retrieve multiple paragraphs from large-scale document collections, we show state-of-the-art performance on HotpotQA distractor setting as well with the exactly same architecture. To understand its behavior in closes multiple paragraph settings, 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 14 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 the Re-rank strategy, **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 14 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: Yoann Lemoine, a French video director, has created music videos for Lana Del Rey, Katy Perry, and an orchestral country pop ballad by which top pop artist?
Yoann Lemoine: Yoann Lemoine (born 16 March 1983) is a French music video director, graphic designer and singer-songwriter. His most notable works include his music video direction for Katy Perry's "Teenage Dream", Taylor Swift's single "Back to December", Lana Del Rey's "Born to Die" and Mystery Jets' "Dreaming of Another World".
Back to December: "Back to December" is a song written and recorded by American singer/songwriter Taylor Swift for her third studio album "Speak Now" (2010). "Back to December" is considered an orchestral country pop ballad and its lyrics are a remorseful plea for forgiveness for breaking up with a former lover.
Blue Jeans (Lana Del Rey song): "Blue Jeans" is a song by American singer-songwriter Lana Del Rey for her second studio album "Born to Die" (2012). Produced by Emile Haynie, the song was written by Del Rey, Haynie, and Dan Heath. Charting across Europe and Asia, "Blue Jeans" reached the top 10 in Belgium, Poland, and Israel. The second was shot and directed by Yoann Lemoine, featuring film noir elements and crocodiles.
Annotated reasoning path: **Yoann Lemoin** → **Back to December** Predicted reasoning path: **Blue Jeans (Lana Del Rey song)** → **Yoann Lemoin** → **Back to December**

Table 12: An examples of the questions where our model predict reasoning paths of the length of three. The model expects that the question is answerable based on the last paragraph of the annotated reasoning path.

Q: who sang the original version of killing me softly
Roberta Flack (I): Roberta Cleopatra Flack (born February 10, 1937) is an American singer. She is known for her No. 1 singles "The First Time Ever I Saw Your Face", "Killing Me Softly with His Song"...
Killing Me Softly with His Song (V), The song was written in collaboration with Lori Lieberman, who recorded the song in late 1971. In 1973 it became a number - one hit in the US and Canada for Roberta Flack, Many artists have covered the song....
Annotated reasoning path: **Killing Me Softly with His Song (V)** Predicted reasoning Path: **Roberta Flack (I)** → **Killing Me Softly with His Song (V)**

Table 13: An example from Natural Questions Open. The bold text represents titles and paragraph indices (e.g., (I) denotes that the paragraph is an introductory paragraph). The highlighted phrase represents a bridge entity and the text in red represents an answer span.

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, "Doní 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 14: An example from HotpotQA distractor development set. Highlighted text show the bridge entities for multi-hop, and also the words in red denotes the predicted answer.