Answering Any-hop Open-domain Questions with Iterative Document Reranking

Ping Nie*
Peking University
ping.nie@pku.edu.cn

Yuyu Zhang*
Georgia Institute of Technology
yuyu@gatech.edu

Arun Ramamurthy
Siemens Corporate Technology
arun.ramamurthy@siemens.com

Le Song
Georgia Institute of Technology
lsong@cc.gatech.edu

ABSTRACT

Existing approaches for open-domain question answering (QA) are typically designed for questions that require either single-hop or multi-hop reasoning, which make strong assumptions of the complexity of questions to be answered. Also, multi-step document retrieval often incurs higher number of relevant but non-supporting documents, which dampens the downstream noise-sensitive reader module for answer extraction. To address these challenges, we propose a unified QA framework to answer any-hop open-domain questions, which iteratively retrieves, reranks and filters documents, and adaptively determines when to stop the retrieval process. To improve the retrieval accuracy, we propose a graph-based reranking model that perform multi-document interaction as the core of our iterative reranking framework. Our method consistently achieves performance comparable to or better than the state-of-the-art on both single-hop and multi-hop open-domain QA datasets, including Natural Questions Open, SQuAD Open, and HotpotQA.

CCS CONCEPTS

• **Information systems** → **Question answering**.

KEYWORDS

open-domain question answering; iterative document reranking; multi-document interaction

ACM Reference Format:
Ping Nie, Yuyu Zhang, Arun Ramamurthy, and Le Song. 2021. Answering Any-hop Open-domain Questions with Iterative Document Reranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462853

*Both authors contributed equally to this work.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462853

1 INTRODUCTION

Open-domain question answering (QA) requires a system to answer factoid questions using a large text corpus (e.g., Wikipedia or the Web) without any pre-defined knowledge schema. Most state-of-the-art approaches for open-domain QA follow the retrieve-and-read pipeline initiated by Chen et al. [3], using a retriever module to retrieve relevant documents, and then a reader module to extract answer from the retrieved documents. These approaches achieve prominent results on single-hop QA datasets such as SQuAD [27], whose questions can be answered using a single supporting document. However, they are inherently limited to answering simple questions and not able to handle multi-hop questions, which require the system to retrieve and reason over evidence scattered among multiple documents. In the task of open-domain multi-hop QA [41], the documents with the answer can have little lexical overlap with the question and thus are not directly retrievable. Take the question in Figure 1 as an example, the last paragraph contains the correct answer but cannot be directly retrieved using TF-IDF. In this example, the single-hop TF-IDF retriever is not able to retrieve the last supporting paragraph since it has no lexical overlap with the question, but this paragraph contains the answer and plays a critical role in the reasoning chain.

Recent studies on multi-hop QA attempt to perform iterative retrievals to improve the answer recall of the retrieved documents. However, several challenges are not solved yet by existing multi-hop QA methods: 1) The iterative retrieval rapidly increases the total number of retrieved documents and introduces much noise to the downstream reader module for answer extraction. Typically, the downstream reader module is noise-sensitive, which works poorly when taking noisy documents as input or missing critical supporting documents with the answer [23]. This requires the QA system to reduce relevant but non-supporting documents fed into the reader module. However, to answer open-domain multi-hop questions, it is necessary to iteratively retrieve documents to increase the overall recall of supporting documents. This dilemma poses a challenge for the retrieval phase of open-domain QA systems; 2) Existing multi-hop QA methods such as MUPPET [11] and Multi-step Reasoner [4] perform a fixed number of retrieval steps, which make strong assumptions on the complexity of open-domain questions and perform fixed number of retrieval steps. In real-world scenarios, open-domain questions may require different number of reasoning steps; 3) The relevance of each retrieved document to the question is independently considered. As exemplified in Figure 1, ABR stands for ALBERT-base reranker, which serves as a reference

In multihop setting, different number of retrieval steps may be required by different approaches (opportunity) or even for different multi hop questions.

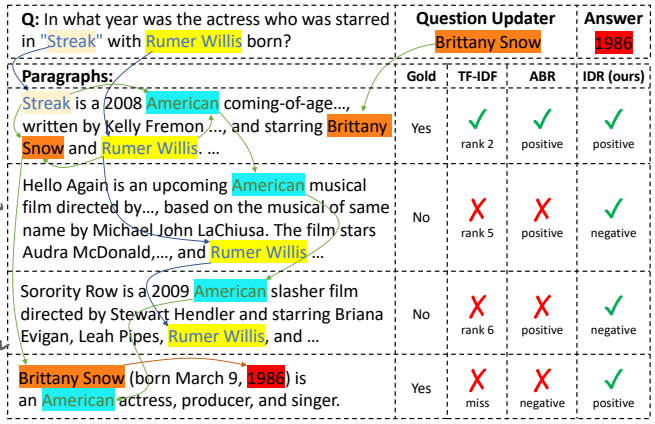

Figure 1: An example open-domain multi-hop question from the HotpotQA dev set, where the question has only partial clues to retrieve supporting documents. The first and fourth paragraphs are gold supporting documents, and the remaining two paragraphs are relevant but non-supporting documents. ABR stands for ALBERT-base reranker, which serves as a reference of the retrieval performance of existing multi-hop QA methods that independently consider the relevance of each document to the question. The ✓ and ✗ symbols mark whether the retriever correctly identifies the document as a supporting / non-supporting one. Below each symbol, we annotate the output of the corresponding retriever with regard to the document, where "positive" ("negative") means that the retriever classifies it as a supporting (non-supporting) document.

of the retrieval performance of existing multi-hop QA methods that independently consider the relevance of each document to the question. Without considering multiple retrieved documents as a whole, these methods can be easily biased to the lexical overlap between each document and the question, and incorrectly classify non-supporting documents as supporting evidence (such as the middle two non-supporting paragraphs in Figure 1, which have decent lexical overlap with the question) and vice versa (such as the bottom paragraph in Figure 1, which has no lexical overlap with the question but is a critical supporting document that contains the answer).

To address the challenges above, we introduce a unified QA framework for answering any-hop open-domain questions named Iterative Document Reranking (IDR). Our framework learns to iteratively retrieve documents with updated question, rerank and filter documents, and adaptively determine when to stop the retrieval process. In this way, our method can significantly reduce the noise introduced by multi-round retrievals and handle open-domain questions that require different number of reasoning steps. To avoid the bias of lexical overlap in identifying supporting documents, our framework considers the question and retrieved documents as a whole and models the multi-document interactions to improve the accuracy of classifying supporting documents.

As illustrated in Figure 2, our method constructs a document graph linked by shared entities to propagate information using a graph attention network (GAT). By leveraging the multi-document information, our reranking model has more knowledge to differentiate supporting documents from irrelevant documents. After initial retrieval, our method updates the question at every retrieval step with a text span extracted from the retrieved documents, and then use the updated question as query to retrieve complementary documents, which are added to the document graph for a new round of interaction. The reranking model is reused to score the documents again and filter the most irrelevant ones. The maintained high-quality shortlist of remaining documents are then fed into the Reader Module to determine whether the answer exists in them. If so, the retrieval process ends and the QA system delivers the answer span extracted by the Reader Module as the predicted answer. Otherwise, the retrieval process continues to the next hop.

Our contributions are summarized as follows:

- *Noise control for iterative retrieval*: We propose a novel QA method to iteratively retrieve, rerank and filter documents, and adaptively determine when to stop the retrieval process. Our method maintains a high-quality shortlist of remaining documents, which significantly reduces the noise introduced to the downstream reader module for answer extraction. Thus, the downstream reader module can extract the answer span with higher accuracy.
- *Unified framework for any-hop open-domain QA*: We propose a unified framework that does not require to pre-determine the complexity of input questions. Different from existing QA methods that are specifically designed for either single-hop or fixed-hop questions, our method can adaptively determine the termination of retrieval and answer any-hop open-domain questions.
- *Multi-document interaction*: We construct entity-linked document graph and employ graph attention network for multi-document interaction, which boosts up the reranking performance. To the best of our knowledge, we are the first to propose graph-based document reranking method for open-domain multi-hop QA.

2 OVERVIEW

2.1 Problem Definition

Given a factoid question, the task of open-domain question answering (QA) is to answer it using a large corpus which can have millions of documents (e.g., Wikipedia) or even billions (e.g., the Web). Let the corpus $C = \{d_1, d_2, \ldots, d_{|C|}\}$ consist of $|C|$ documents as the basic retrieval units[1]. Each document d_i can be viewed as a sequence of tokens $t_1^{(i)}, t_2^{(i)}, \ldots, t_{|d_i|}^{(i)}$. Formally, given a question q, the task is to find a text span $t_s^{(j)}, t_{s+1}^{(j)}, \ldots, t_e^{(j)}$ from one of the documents d_j that can answer the question[2]. For open-domain multi-hop QA, the final documents with the answer are typically multiple hops away from the question, i.e., the system is required to find seed documents and subsequent supporting documents in order of a chain or directed graph to locate the final documents. The retrieved documents are usually connected via shared entities or semantic similarities, and the formed chain or directed graph of documents can be viewed as the reasoning process for answering the question.

[1] We use the natural paragraphs as the basic retrieval units.
[2] In this work, we focus on the extractive or span-based QA setting, but the problem definition and our proposed method can be generalized to other QA settings as well.

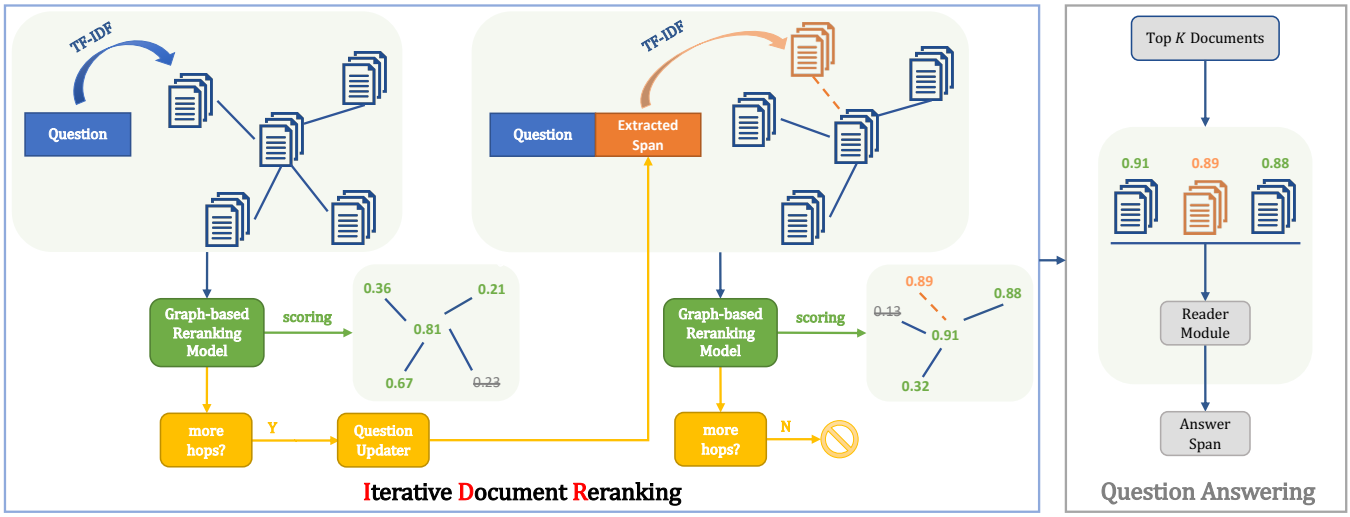

Figure 2: An overview of the IDRQA system, which consists of an Iterative Document Reranking (IDR) phase and a question answering phase. Given an open-domain question, IDR iteratively retrieves, reranks and filters documents, and adaptively determines when to stop the retrieval process. After the initial retrieval, IDR updates the question with an extracted text span as a new query to retrieve more documents at every iteration. Once the retrieval is done, the final highest-scoring documents are fed into the downstream reader module for answer extraction.

Note that the task of *open-domain multi-hop QA* that we describe above is much different from the *few-document setting of multi-hop QA* [25], where the QA system is provided with a tiny set of documents that consists of all the gold supporting documents together with several irrelevant "distractor" documents. The *few-document setting* is designed to test the system's capability of multi-hop reasoning given all of the gold supporting documents, but this is far from being realistic. A real-world open-domain QA system has to locate the necessary supporting documents from a large corpus on its own, which is especially challenging for multi-hop questions since the indirect supporting documents are not easily retrievable given the question itself.

The nature of multi-hop questions poses significant challenge for retrieving supporting documents, which is crucial to the downstream QA performance. To address this challenge, we argue that it is necessary to iteratively retrieve, rerank and filter documents, so that we can maintain a high-quality shortlist of documents. To this end, we propose the Iterative Document Reranking (IDR) method, which is introduced in the next section.

2.2 System Overview

We illustrate the overview of our IDRQA system in Figure 2. The IDRQA system first uses a given question as query to retrieve top $|\mathcal{D}|$ documents using TF-IDF. To construct the document graph, IDR extracts the entities from the question and retrieved documents using an off-the-shelf Named Entity Recognition (NER) system and connects two documents if they have shared entities. The graph-based reranking model takes the document graph as input to score each document, filter the lowest-scoring documents, and adaptively determines whether to continue the retrieval process. At every future retrieval step, IDR updates the question with an extracted text span from the retrieved documents as a new query to retrieve more documents. Once the retrieval is done, the final highest-scoring documents are concatenated to feed into the downstream reader module [7, 16] for answer extraction.

Algorithm 1: Iterative Document Reranking (inference)

input : A textual question q; the maximum number of retrieval hops $\mathcal{H}$; the number of retrieved documents at each hop N; the number of remaining documents after each reranking process K; the graph-based reranking model $\mathcal{M}$; the reader module $\mathcal{R}$; the question updater model $\mathcal{U}$

output: Predicted answer a

$h \leftarrow 0$
$\mathcal{D} \leftarrow \emptyset$
while $h <= \mathcal{H}$ **do**
  $h \leftarrow h + 1$
  $\mathcal{D}_{\text{new}} \leftarrow$ Retrieve top N documents according to q
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{\text{new}}$
  $\mathcal{D} \leftarrow$ Rerank and get top K documents using $\mathcal{M}(q, \mathcal{D})$
  $a \leftarrow$ Predict the answer using $\mathcal{R}(q, \mathcal{D})$
  if *a is not None* **then** /* No more hop needed */
    return a
  else /* More hop with question updater */
    $q \leftarrow \mathcal{U}(q, \mathcal{D})$
return a

To describe the pipeline of our IDRQA system more precisely, we provide a concise algorithm that summarizes the inference procedure of the iterative document reranking (IDR) phase, as in Algorithm 1. We maintain a set of retrieved documents $\mathcal{D}$. For each

retrieval step, we retrieve the top N documents according to the question q and then append all the newly retrieved documents to the current set of retrieved documents. Then we use the graph-based reranking model $\mathcal{M}$ to rerank those documents, get top K of them, and filter the other ones. This shortlist of retrieved documents together with the question q are then fed in to the downstream reader module $\mathcal{R}$ for answer extraction. Note that the reader module has the option to predict that there is no answer to be extracted from the provided documents. If the reader module does predict no answer, we use the question updater model $\mathcal{U}$ to update the question with an extracted span and move to the next hop of retrieval. The while loop continues until a valid answer is extracted by the reader module or the maximum number of retrieval hops $\mathcal{H}$ is reached.

3 IDRQA SYSTEM

3.1 Graph-based Reranking Model

The graph-based reranking model (Figure 3) is designed to precisely identify the supporting documents in the document graph. We present the components of this reranking model as follows.

Contextual Encoding. Given a question q and $|\mathcal{D}|$ documents retrieved by TF-IDF, we concatenate the tokens of the question and each document to feed into the pre-trained language model as:

$$\mathcal{I}_{q,d_k} = [\text{CLS}]\ q_1 \dots q_{|q|}\ [\text{SEP}]\ t_1^{(k)} \dots t_{|d_k|}^{(k)}\ [\text{SEP}],$$

where $|q|$ and $|d_k|$ denote the number of tokens in the question q and the document d_k, respectively. [CLS] and [SEP] are special tokens used in pre-trained language models such as BERT [7] and ALBERT [16]. Thus we independently encode each document d_k along with the question q to obtain the contextual representation vector $\mathbf{v}_{q,d_k} \in \mathbb{R}^{L\times h}$, where L is the maximum length of the input tokens $\mathcal{I}$, and h is the embedding size. For efficient batch computation, we pad or truncate the input tokens to the length of L. We then concatenate all documents' contextual representation vectors as $\mathbf{v} \in \mathbb{R}^{L|\mathcal{D}|\times h}$.

Graph Attention. After we obtain the question-dependent encoding of each document, we employ a Graph Attention Network (GAT; Veličković et al. [31]) to propagate information on the document graph, where two documents are connected if they have shared entities. To be more specific, for each shared entity E_i in the document graph, we perform pooling over its token embeddings from $\mathbf{v}$ to produce the entity embedding as $\mathbf{e}_i = \text{Pooling}(\mathbf{t}_1^{(i)}, \mathbf{t}_2^{(i)}, \dots, \mathbf{t}_{|E_i|}^{(i)})$ where $\mathbf{t}_j^{(i)}$ is the embedding of the j-th token in E_i, and $|E_i|$ is the number of tokens in E_i. We use both mean- and max-pooling, thus we have $\mathbf{e}_i \in \mathbb{R}^{2h}$. Inspired by Qiu et al. [26], we apply a dynamic soft mask on the entities, serving as the information "gatekeeper" which assigns more weights to entities pertaining to the question. The soft mask applied on each entity E_i is computed as

$$m_i = \sigma\left(\frac{\mathbf{q}V\mathbf{e}_i}{\sqrt{2h}}\right) \quad (1)$$

$$\mathbf{g}_i = m_i\mathbf{e}_i, \quad (2)$$

where $\mathbf{q} \in \mathbb{R}^{2h}$ is the concatenated mean- and max-pooling of the question token embeddings, and $V \in \mathbb{R}^{2h\times 2h}$ is a linear projection matrix, $\sigma(\cdot)$ is the sigmoid function, and $\mathbf{g}_i$ is the masked entity embedding. We then use GAT to disseminate information between entities. Starting from $\mathbf{g}_i^{(0)} = \mathbf{g}_i$, GAT iteratively updates the embedding of each entity with the information from its neighbors as

$$\mathbf{h}_i^{(t)} = W_i\mathbf{g}_i^{(t-1)} + b_i \quad (3)$$

$$\mathbf{g}_i^{(t)} = \text{ReLU}\Big(\sum_{j\in N(i)} \alpha_{i,j}^{(t)}\mathbf{h}_j^{(t)}\Big), \quad (4)$$

where $\mathbf{h}_i^{(t)}$ denotes the hidden states of E_i on the t-th GAT layer, $W_1 \in \mathbb{R}^{h\times 2h}$ is a linear projection matrix, b_i is a bias term, $N(i)$ is the set of neighbor entities of E_i, and the entity-entity attention $\alpha_{i,j}^{(t)}$ is computed as follows:

$$s_{i,j}^{(t)} = \text{LeakyReLU}(W_2[\mathbf{h}_i^{(t)}; \mathbf{h}_j^{(t)}]) \quad (5)$$

$$\alpha_{i,j}^{(t)} = \frac{\exp(s_{i,j}^{(t)})}{\sum_k \exp(s_{i,k}^{(t)})}, \quad (6)$$

where $W_2 \in \mathbb{R}^{2h}$ is a linear projection matrix. We finally obtain the GAT updated entity embeddings $\mathbf{g}_i^{(T)}$, where T is the number of GAT layers.

Multi-document Fusion. To further propagate the information to non-entity tokens, we firs use the embedding of each entity token as

$$\hat{\mathbf{t}}_j^{(i)} = W_3[\mathbf{t}_j^{(i)}; \mathbf{g}_i^{(T)}], \quad (7)$$

where $W_3 \in \mathbb{R}^{h\times 3h}$ is a linear projection matrix. Then we replace the corresponding vectors in $\mathbf{v}$ with $\hat{\mathbf{t}}_j^{(i)}$ to obtain $\hat{\mathbf{v}} \in \mathbb{R}^{L|\mathcal{D}|\times h}$. Finally, $\hat{\mathbf{v}}$ is fed into a Transformer [30] layer for multi-document fusion, which updates the representations of all the tokens and outputs the fused representation vectors $\tilde{\mathbf{v}} \in \mathbb{R}^{L|\mathcal{D}|\times h}$.

Document Filter. For each document, we use the [CLS] token embedding from $\tilde{\mathbf{v}}$ as the document representation, which is fed into a binary classifier to score the document's supporting level. In each retrieval hop, the top K documents with the highest scores are selected and the rest are filtered.

3.2 Question Updater

In open-domain multi-hop QA, the question seldom contains all the retrievable clues and one has to identify the missing information to proceed with further reasoning [41]. To increase the recall of indirect supporting documents, we integrate the query generator of GoldEn Retriever [25] into our system, which serves as the question updater at every retrieval step after the initial one.

Question Updater comes after Reader module if no answer was found in top K documents. It aims to generate the *clue span* other than the answer span from the reranked top K documents given current question q. Thus conceptually, this process is similar to the Reader Module which extracts answer span from retrieved documents. More specifically, following GoldEn Retriever [25], we use the same method for data construction to train a span extractor model, which extracts the *clue span* from retrieved documents by predicting its start and end tokens. Formally, given the question q and the reranked top K documents $\mathcal{D}_K = \{d_1, d_2, \dots, d_K\}$, the

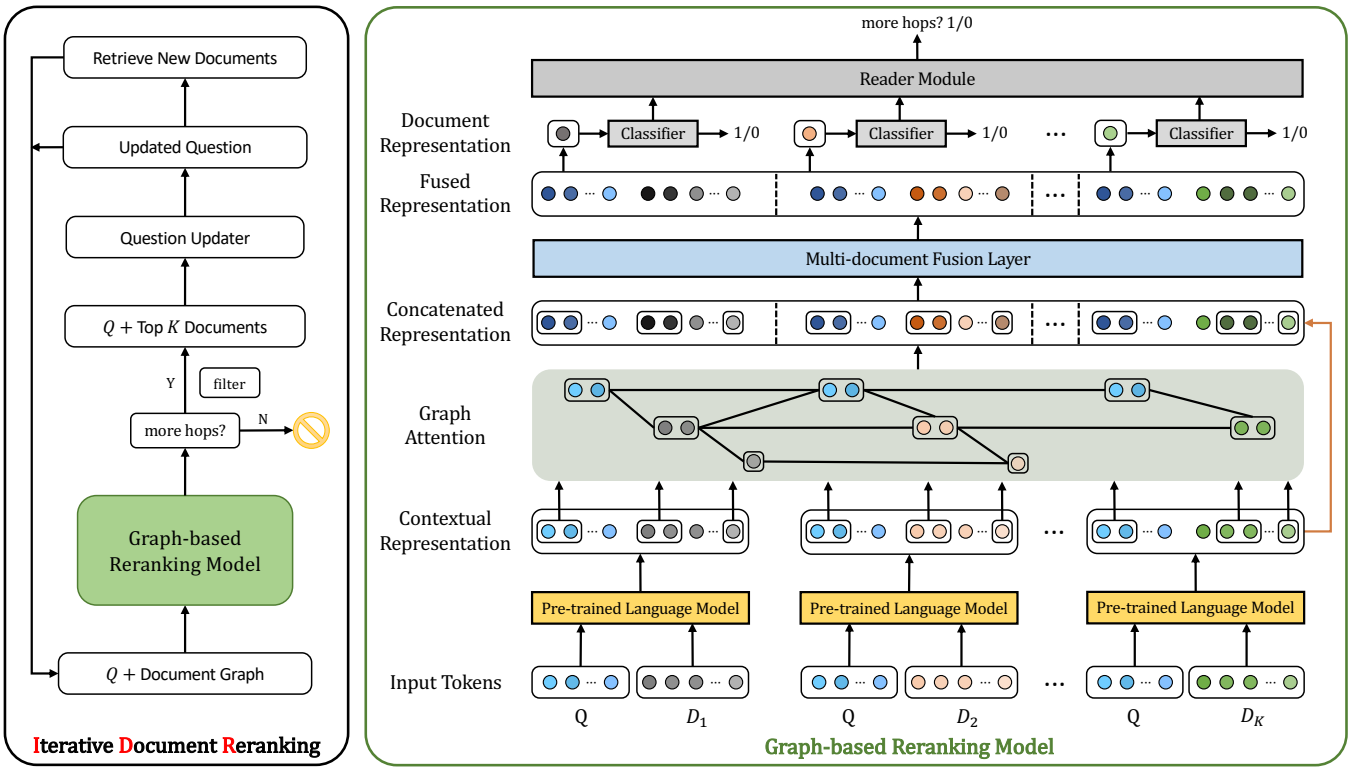

Figure 3: As the core of our IDR framework, the Graph-based Reranking Model first encodes the question and each retrieved document with pre-trained language model to generate contextual representations, and uses the shared entities to propagate information using a Graph Attention Network (GAT). After the entity-entity interaction across multiple documents, the updated entity representations with the original contextual encodings are fed into the fusion layer for further interaction. Finally, the reranking model takes pooled document representations to score each document and filter low-scoring documents. The maintained high-quality shortlist of remaining documents are then fed into the Reader Module to determine whether the answer exists in them. If so, the retrieval process ends and the QA system delivers the answer span extracted by the Reader Module as the predicted answer. Otherwise, the retrieval process continues to the next hop.

Question Updater generates the *clue span* C and concatenate C with the original question q as

$$C = \mathcal{U}(q, \mathcal{D}_K) \tag{8}$$

$$q = \text{concatenate}(q, [\text{SEP}], C), \tag{9}$$

where [SEP] is a special separator token. In order to train the Question Updater, we construct each training sample as a triple of $< q, \mathcal{D}_K, C >$, where q is the question, $\mathcal{D}_K = \{d_1, d_2, \ldots, d_K\}$ is the set of top K retrieved documents reranked by our graph-based reranking model.

3.3 Reader Module

In this work, we mainly focus on the retrieval phase, and use a standard span-based reader module as in BERT [7] and ALBERT [16]. We concatenate the tokens of the question q and the final top K reranked documents to feed into the reader module. At inference time, the reader finds the best candidate answer span by

$$\arg\max_{i,j,\, i \le j} P_i^{\text{start}} P_j^{\text{end}}, \tag{10}$$

where P_i^{start}, P_j^{end} denote the probability that the i-th and j-th tokens are the start and end positions in the concatenated text, respectively, of the answer span. During inference, there is no guarantee that the answer span exists in the reader's input text. To handle the no-answer cases, the reader predicts P_{na} as the probability of having no answer span, and compares P_{na} with $\max_{i,j,\, i \le j} P_i^{\text{start}} P_j^{\text{end}}$ to determine the output between a special no-answer prediction and the best candidate answer span.

4 EXPERIMENTS

4.1 Experimental Settings

We conduct all the experiments on a GPU-enabled (Nvidia RTX 6000) Linux machine powered by Intel Xeon Gold 5125 CPU at 2.50GHz with 384 GB RAM. For the graph-based reranking model, we set the maximum token sequence length $L = 250$, the number of retrieved documents $|\mathcal{D}| = 8$, the maximum number of entities $|\mathcal{E}| = 120$. The embedding size h is 768 and 4,096 for the ALBERT-base and ALBERT-xxlarge model, respectively. The graph attention module has $T = 2$ GAT layers. The maximum number of retrieval

hops $\mathcal{H} = 4$. The top $K = 4$ reranked paragraphs are sent into the downstream reader module.

We implement our system based on HuggingFace's Transformers [37]. Following the previous state-of-the-art method [10], we use ALBERT-xxlarge [16] as the pre-trained language model. We use AdamW [37] as the optimizer and tune the initial learning rate between 1.5e−5 and 2.5e−5.

4.2 Datasets

We evaluate our method on three open-domain QA benchmark datasets: HotpotQA [41], Natural Questions Open [17] and SQuAD Open [3]. On all the three datasets, we focus on the *full wiki* open-domain QA setting, which requires the system to retrieve evidence paragraphs from the entire Wikipedia and extract the answer span from the retrieved paragraphs.

Following the train / dev / test splits of Natural Questions Open and SQuAD Open in previous works [14, 17], we use the original validation set as our test set and keep 10% training set as our dev set. Natural Questions Open and SQuAD open consist of single-hop questions, while HotpotQA consists of 113K crowd-sourced multi-hop questions that require Wikipedia introduction paragraphs to answer. In the train and dev splits of HotpotQA, each question for training comes with two gold supporting paragraphs annotated by the crowd workers. Thus we can evaluate the retrieval performance on the dev set of HotpotQA dataset.

limitation that dataset answers are located in intro Wikipedia paragraphs

Are we combining this with Natural Question dataset ??

4.3 Data Preprocessing

Here we describe the details of data preprocessing methods that we develop for the HotpotQA dataset.

Training data construction for the graph-based reranking model. Graph-based reranking model aims to precisely score the retrieved documents by considering multiple documents as a whole instead of independent instances. In order to make our model reusable and robust to new test cases, we carefully design the training data construction method. To keep the distribution of training and test data consistent, we add negative samples to the training data for our graph-based reranking model. Formally, We pair each question q with a set of documents $\mathcal{D}_{\text{train}}$ to form a training sample. We design each training sample with the following strategies: 1) For each training sample which contains all supporting documents, we pair the question q with $\mathcal{D}_{\text{train}}$ where $\mathcal{D}_{\text{train}}$ includes all NP_s supporting documents necessary for multi-hop QA and $|\mathcal{D}_{\text{train}}| - NP_s$ noisy documents. $\mathcal{D}_{\text{train}}$ is a set of documents in training. NN_s, NP_s are the number of noisy documents and the number of supporting documents; 2) For each training sample which contains partial supporting documents, we paired question q with $\mathcal{D}_{\text{train}}$ where $\mathcal{D}_{\text{train}}$ contains NP_s supporting documents and $|\mathcal{D}_{\text{train}}| - NP_s$ noisy documents. NP_s supporting documents are randomly sampled from all supporting documents; 3) For noisy documents, we sample them according to their score given by TF-IDF; 4) For multi-step reranking, we also randomly sample 30% questions and use Question Updater to generate *the clue span* and update questions. These new questions and their original paired documents are also used as training samples; 5) We concatenate all documents in $\mathcal{D}_{\text{train}}$ in random order, rather than the reranked order. We use $|\mathcal{D}_{\text{train}}| = 6$ and $NP_s = 2$ in our experiments.

due to HotpotQA(??)

? why(??)

Training data construction for the reader module. We designed the training data for the Reader Module carefully since the Reader module is not only used to predict the final answer but also to tell that *no answer* found in in current context. The training sample is a triple of $< q, \mathcal{D}, a >$ where q is question, $\mathcal{D} = \{d_1, d_2, d_3, d_4\}$ is 4 documents feed into QA model and a is the final answer. For each question, we construct 5 types of training sample: 1) We concatenate two supporting passages (ordered as originally in the dataset) and two highest TF-IDF scored negative passages (ordered from higher to lower reranking score) as training samples; 2) We construct a random shuffled version of the 1st type in passage level; 3) We randomly replace one of the supporting passages of the 1st type by a negative passage; 4) One passage which has the final answer and is not one of the supporting passage and three negative passages; 5) We construct four negative passages with high TF-IDF score which are not supporting passages and do not contain the final answer.

4.4 Evaluation Metrics

We evaluate both the paragraph reranking performance and the overall QA performance. Following the existing studies [1, 23, 24], we evaluate the paragraph-level retrieval accuracy using the Paragraph Exact Match (EM) metric, which compares the top 2 paragraphs with the gold supporting paragraphs. For the QA performance, we report standard answer Exact Match (EM) and F_1 scores to measure the overlap between the gold answer and the extracted answer span.

4.5 Overall Results

We evaluate our method on both single-hop and multi-hop open-domain QA datasets. Note that our method is hop-agnostic, i.e., we consistently use the same setting of $\mathcal{H} = 4$ as the maximum number of retrieval hops for all the three datasets.

For the multi-hop QA task, we report performance on both the dev and test sets of the HotpotQA dataset. As reported in Table 1, we compare the performance of our system IDRQA with various existing methods on the HotpotQA full wiki dev set. Since the golden supporting paragraphs are only labeled on the train / dev splits of the HotpotQA dataset, we can only report the paragraph EM metric on the dev set. In Table 2, we compare our system with various existing methods on HotpotQA full wiki test set. IDRQA outperforms all published and previous unpublished methods on the HotpotQA dev set and the hidden test set on the official leaderboard (on May 21, 2020). Note that we focus on the QA task in this paper, and for the supporting fact prediction task we simply concatenate each pair of question and supporting fact to train a binary classification model. The joint EM and F_1 scores combine the evaluation of answer spans and supporting facts as detailed in [41]. Thus we are behind state-of-the-art performance on supporting fact and joint scores.

For the single-hop QA task, we evaluate our method on Natural Questions (NQ) Open and SQuAD Open datasets. We summarize the performance in Table 3. For NQ Open, we follow the previous work DPR [14] to use dense retriever instead of TF-IDF for document retrieval. Our method also achieves QA performance comparable to or better than the state-of-the-art methods on both datasets, which

Table 1: Performance on the HotpotQA full wiki dev set.

Method	Answer		Supporting Fact		Paragraph
	EM	F_1	EM	F_1	EM
Cognitive Graph [9]	37.6	49.4	23.1	58.5	57.8
DecompRC [21]	–	43.3	–	–	–
MUPPET [11]	31.1	40.4	17.0	47.7	
GoldEn Retriever [25]	–	49.7	–	–	–
DrKIT [8]	35.7	46.6	–	–	–
Semantic Retrieval MRS [23]	46.5	58.8	39.9	71.5	63.9
Transformer-XH [43]	50.2	62.4	42.2	71.6	–
Recurrent Retriever [1]	60.5	73.3	49.3	76.1	75.7
HopRetriever [19]	62.1	75.2	**52.5**	78.9	**82.5**
IDRQA (ours)	**62.9**	**75.9**	51.3	**79.1**	79.8

Table 2: Performance on the HotpotQA full wiki test set.

Method	Answer		Supporting Fact		Joint	
	EM	F_1	EM	F_1	EM	F_1
DecompRC [21]	30.0	40.6	–	–	–	–
QFE [24]	28.6	38.0	14.2	44.3	8.6	23.1
Cognitive Graph [9]	37.1	48.8	22.8	57.6	12.4	34.9
MUPPET [11]	30.6	40.2	16.6	47.3	10.8	27.0
GoldEn Retriever [25]	37.9	48.5	30.6	64.2	18.0	39.1
DrKIT [8]	42.1	51.7	37.0	59.8	24.6	42.8
Semantic Retrieval MRS [23]	45.3	57.3	38.6	70.8	25.1	47.6
Transformer-XH [43]	51.6	64.0	40.9	71.4	26.1	51.2
Recurrent Retriever [1]	60.0	73.0	49.0	76.4	35.3	61.1
Hierarchical Graph Network [10]	59.7	71.4	51.0	77.4	37.9	62.2
HopRetriever [19]	60.8	73.9	53.1	79.3	38.0	63.9
Multi-hop Dense Retrieval [39]	62.3	75.3	**57.5**	**80.9**	**41.8**	**66.6**
IDRQA (ours)	**62.5**	**75.9**	51.0	78.9	36.0	63.9

Table 3: Answer EM scores on the test set of Natural Questions Open and SQuAD Open.

Method	NQ	SQuAD
DrQA [3]	–	29.8
R^3 [34]	–	29.1
Multi-step Reasoner [4]	–	31.9
BERTserini [40]	–	38.6
MUPPET [11]	–	39.3
Multi-passage BERT [36]	–	53.0
ORQA [17]	33.3	20.2
Recurrent Retriever [1]	31.7	**56.5**
Dense Passage Retriever [14]	41.5	36.7
IDRQA (ours)	**45.5**	**56.6**

shows the robustness of our method across different open-domain QA datasets.

4.6 Detailed Analysis

Ablation study. To investigate the effectiveness of each module in IDRQA, we compare the performance of several variants of our system on HotpotQA full wiki dev set. As shown in Table 4, once we disable the *Iterative Reranking*, *Graph-based Reranking* and *Question Updater* module, both the paragraph reranking and QA performance drop significantly. Notably, there is a 11 points drop in Paragraph EM decrease and a 12 points drop in Answer F_1 when *Graph-based Reranking* is removed from IDRQA. This shows the importance of the graph-based reranking model in our system. To further study the impact of these modules, we decompose the QA performance into question categories *Bridge* and *Comparison*. We find that the QA performance on the bridge questions drops much more significantly than that on the comparison questions. This is because the comparison questions require to compare two entities mentioned in the question [41], thus iterative retrieval and multi-hop reasoning may not be necessary. In contrast, to answer the bridge questions which often have missing entities, our iterative graph-based document reranking method is of crucial importance.

Bridge: In what year was the actress who was starred in "Streak" with Rumer Willis born?	Supporting
Streak is a 2008 American … film directed by Demi Moore, … and starring Brittany Snow and Rumer Willis…	Yes
Sorority Row is a 2009 American slasher film … starring Briana Evigan, Leah Pipes, Rumer Willis, …	No
Brittany Anne Snow (born March 9, 1986) is an American actress, producer, and singer.	Yes
IDRQA answer: 1986 ✓ **ABR answer:** 2009 ✗	

Bridge: What screenwriter with credits for "Evolution" co-wrote a film starring Nicolas Cage and Téa Leoni?	Supporting
David Weissman is a screenwriter and director. His film credits include "The Family Man" (2000), "Evolution" (2001), and "When in Rome" (2010).	Yes
The Family Man is a 2000 American romantic comedy-drama film directed by Brett Ratner, written by David Diamond and David Weissman, and starring Nicolas Cage and Téa Leoni.	Yes
IDRQA answer: David Weissman ✓ **ABR answer:** David Weissman ✓	

Comparison: Who is older, Annie Morton or Terry Richardson?	Supporting
Annie Morton (born October 8, 1970) is an American model born in Pennsylvania…	Yes
Terrence "Uncle Terry" Richardson (born August 14, 1965) is an American fashion and portrait photographer…	Yes
IDRQA answer: Annie Morton ✗ **ABR answer:** Annie Morton ✗	

Figure 4: Case study of example questions with supporting paragraphs from HotpotQA dev set.

Table 4: Ablation study of our system in different settings on HotpotQA full wiki dev set.

Ablation Setting	Bridge (79.9%)		Comparison (20.1%)		Full Dev (100%)		
	Answer		Answer		Answer		Paragraph
	EM	F_1	EM	F_1	EM	F_1	EM
IDRQA	58.1	73.2	71.4	77.9	60.7	74.2	73.8
w/o Graph-based Reranking	47.4	59.5	70.1	76.4	52.1	62.9	62.6
w/o Iterative Reranking	49.8	61.7	70.5	77.3	54.3	64.5	65.2
w/o Question Updater	56.6	71.4	71.3	77.8	59.8	72.9	70.3

Impact of retrieval steps. IDRQA aggregates the document scores to check whether the collected evidence is enough to answer the question, and adaptively determines when to stop the retrieval process. We investigate the number of retrieval steps selected by IDRQA, and report its distribution with breakdown performance in Table 5. Over 60% questions are answered with 2-step retrieval. About 20% questions are answered with 1-step retrieval, which is close to the ratio of the comparison questions that may not need iterative retrieval. For questions that IDRQA selects to perform over 2-step retrieval, a significant drop on both reranking and QA performance is observed, showing that these questions are the hardest ones in HotpotQA.

Case study and limitations. We showcase several example questions with answers from IDRQA and the baseline ALBERT-base reranker (ABR) in Figure 4. The first case is a hard bridge question where ABR extracts the wrong answer from a relevant but non-supporting paragraph, showing the advantage of iterative reranking in our system. The second question is correctly answered by both IDRQA and ABR, since it provides sufficient clues to retrieve both paragraphs. The final case is a comparison question that requires numerical reasoning, which is not correctly answered by both IDRQA and ABR. This shows the limitation of our system, and we plan to explore the combination of multi-hop and numerical reasoning in future work.

Table 5: Distribution of the selected retrieval steps on HotpotQA dev set, which is adaptively determined by IDRQA.

Retrieval	% of Questions	Paragraph	Answer	
		EM	EM	F_1
1-step	21.8	88.4	70.3	81.9
2-step	63.2	76.9	62.0	75.9
3-step	5.3	39.6	45.2	59.8
4-step (max)	9.7	22.0	37.7	50.4

5 RELATED WORK

Open-domain QA. For text-based QA, the QA system is provided with semi-structured or unstructured text corpora as the knowledge source to find the answer. Text-based QA dates back to the QA track evaluations of the Text REtrieval Conference (TREC) [32]. Traditional approaches for text-based QA typically use a pipeline

of question parsing, answer type classification, document retrieval, answer candidate generation, and answer reranking, such as the famous IBM Watson system which beat human players on the *Jeopardy!* quiz show [12]. However, such pipeline-based QA systems require heavy engineering efforts and only work on specific domains. The open-domain QA task was originally proposed and formalized in Chen et al. [3], which builds a simple pipeline with a TF-IDF retriever module and a RNN-based reader module to produce answers from the top 5 retrieved documents. Different from the machine reading comprehension task (MRC) that provides a single paragraph or document as the evidence [27], open-domain QA is more challenging since the retrieved documents are inevitably noisy. Recent works on open-domain QA largely follow the retrieve-and-read approach, and have made prominent improvement on both the retriever module [17, 23, 34] and the reader module [22, 33, 35]. These approaches simply perform one-shot document retrieval to handle single-hop questions. However, for complicated questions that require multi-hop reasoning, these approaches are not applicable since they fail to collect necessary evidence scattered among multiple documents.

Open-domain multi-hop QA. HotpotQA [41] is crowd-sourced over Wikipedia as the largest free-form dataset for open-domain multi-hop QA to date. Recently, a variety of approaches have been proposed to address the multi-hop challenge. DecompRC [21] decomposes a multi-hop question into simpler sub-questions and leverages single-hop QA models to answer it, which still uses one-shot TF-IDF retrieval to collect relevant documents. BERT Reranker [5], DrKIT [8] and Transformer-XH [43] employ the entities in the question and the retrieved documents to link additional documents, which expands the one-shot retrieval results and improves the evidence coverage. GoldEn Retriever [25] adopts iterative TF-IDF retrieval by generating a new query for each retrieval step. Graph Retriever [20] employs entity linking to iteratively retrieve and construct a graph of documents. These iterative retrieval methods mitigate the recall problem of one-shot retrieval, however, without iterative reranking and filtering process, the expanded documents inevitably introduce noise to the downstream QA model. Multi-step Reasoner [4] and MUPPET [11] read retrieved documents to reformulate the query in latent space for iterative retrieval. However, these embedding-based retrieval methods have difficulties to capture the lexical information in entities due to the compression of information into embedding space. Moreover, these methods perform a fixed number of retrieval steps, which are not able to handle questions that require arbitrary hops of reasoning. Recurrent Retriever [1] supports adaptive retrieval steps, but can only select one document at each step and has no interactions among documents outside of the retrieval chain. In contrast, our method leverages document graph instead of chain to propagate information, reranks and filters documents at each hop of retrieval, and terminates the retrieval process according to the number of positive retrieved documents in the graph.

Graph Neural Networks for QA. Encouraged by the success of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), graph neural networks (GNNs) are proposed to generalize both of them and organize the connections of neurons according to the structure of the input data [2, 38]. The main idea is to generate a node's representation by aggregating its own features and neighbors' features. Similar to CNNs, in GNNs, multiple graph convolutional layers can be stacked to produce high-level node representations. Many popular variants of GNNs are proposed, such as GraphSage [13], Graph Convolutional Network (GCN) [15], Graph Attention Network (GAT) [31], etc. GNNs have achieved great success on graph-related tasks such as node classification, node regression and graph classification. Recently, Graph neural networks (GNNs) have been shown effective on knowledge-based QA tasks by reasoning over graphs [6, 28, 42]. Recent studies on text-based QA also leverage GNNs for multi-hop reasoning. HDE-Graph [29] constructs the graph with entity and document nodes to enable rich information interaction. CogQA [9] extracts candidate answer spans and next-hop entities to build the cognitive graph for reasoning. HGN [18] employs a hierarchical graph that consists of paragraph, sentence and entity nodes for reasoning on different granularities. DFGN [26] introduces a fusion layer on top of the entity graph with a mask prediction module. Graph Retriever [20] uses graph convolution network to fuse information on the entity-linked graph of passages. Multi-grained MRC [44] utilizes GATs to obtain different levels of representations of documents for machine reading comprehension. These GNN-based methods serve as the *reader* module to extract answers from a few documents. In contrast, our work employs graph attention network, a popular GNN variant, in the *retriever* module. To the best of our knowledge, we are the first to propose GNN-based document reranking method for open-domain multi-hop QA.

6 CONCLUSION

We present a QA framework that can answer any-hop open-domain questions, which iteratively retrieves, reranks and filters documents with a graph-based reranking model, and adaptively decides how many steps of retrieval and reranking are needed for a multi-hop question. Our method consistently achieves promising performance on both single- and multi-hop open-domain QA datasets.

REFERENCES

[1] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering. In *International Conference on Learning Representations*.

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv* (2018).

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada, 1870–1879.

[4] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering. In *International Conference on Learning Representations*.

[5] Rajarshi Das, Ameya Godbole, Dilip Kavarthapu, Zhiyu Gong, Abhishek Singhal, Mo Yu, Xiaoxiao Guo, Tian Gao, Hamed Zamani, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step Entity-centric Information Retrieval for Multi-Hop Question Answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Hong Kong, China, 113–118.

[6] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question Answering by Reasoning Across Documents with Graph Convolutional Networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, 2306–2317.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota, 4171–4186.

[8] Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable Reasoning over a Virtual Knowledge Base. In *International Conference on Learning Representations*.

[9] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2694–2703.

[10] Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical Graph Network for Multi-hop Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, 8823–8838.

[11] Yair Feldman and Ran El-Yaniv. 2019. Multi-Hop Paragraph Retrieval for Open-Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy, 2296–2309.

[12] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building Watson: An overview of the DeepQA project. *AI magazine* 31, 3 (2010), 59–79.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[14] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2004.04906* (2020).

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations* (Palais des Congrès Neptune, Toulon, France) *(ICLR '17)*. https://openreview.net/forum?id=SJU4ayYgl

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.

[17] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6086–6096.

[18] Chong Li, Kunyang Jia, Dan Shen, C.J. Richard Shi, and Hongxia Yang. 2019. Hierarchical Representation Learning for Bipartite Graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization.

[19] Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2020. HopRetriever: Retrieve Hops over Wikipedia to Answer Complex Questions. *arXiv preprint arXiv:2012.15534* (2020).

[20] Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Knowledge guided text retrieval and reading for open domain QA. *arXiv preprint arXiv:1911.03868* (2019).

[21] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6097–6109.

[22] Jianmo Ni, Chenguang Zhu, Weizhu Chen, and Julian McAuley. 2019. Learning to Attend On Essential Terms: An Enhanced Retriever-Reader Model for Open-domain Question Answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 335–344.

[23] Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the Importance of Semantic Retrieval for Machine Reading at Scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2553–2566.

[24] Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. 2019. Answering while Summarizing: Multi-task Learning for Multi-hop QA with Evidence Extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2335–2345.

[25] Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering Complex Open-domain Questions Through Iterative Query Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2590–2602.

[26] Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically Fused Graph Network for Multi-hop Reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy, 6140–6150.

[27] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 2383–2392.

[28] Daniil Sorokin and Iryna Gurevych. 2018. Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 3306–3317.

[29] Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop Reading Comprehension across Multiple Documents by Reasoning over Heterogeneous Graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2704–2713.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[32] Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. 200–207.

[33] Bingning Wang, Ting Yao, Qi Zhang, Jingfang Xu, Zhixing Tian, Kang Liu, and Jun Zhao. 2019. Document Gated Reader for Open-Domain Question Answering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 85–94.

[34] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced Ranker-Reader for Open-Domain Question Answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[35] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018. Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering. In *International Conference on Learning Representations*.

[36] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 5878–5882.

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

[38] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[39] Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval. In *International Conference on Learning Representations*. https://openreview.net/forum?id=EMHoBG0avc1

[40] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-End Open-Domain Question Answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Association for Computational Linguistics, Minneapolis, Minnesota, 72–77.

[41] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2369–2380.

[42] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[43] Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-XH: Multi-Evidence Reasoning with eXtra Hop Attention. In *International Conference on Learning Representations*.

[44] Bo Zheng, Haoyang Wen, Yaobo Liang, Nan Duan, Wanxiang Che, Daxin Jiang, Ming Zhou, and Ting Liu. 2020. Document Modeling with Graph Attention Networks for Multi-grained Machine Reading Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 6708–6718. https://doi.org/10.18653/v1/2020.acl-main.599