HopRetriever: Retrieve Hops over Wikipedia to Answer Complex Questions

Shaobo Li,[1]* Xiaoguang Li,[2] Lifeng Shang,[2] Xin Jiang,[2]
Qun Liu,[2] Chengjie Sun,[1] Zhenzhou Ji,[1] Bingquan Liu[1]
[1]Harbin Institute of Technology
[2]Huawei Noah's Ark Lab
shli@insun.hit.edu.cn, {lixiaoguang11, shang.lifeng, Jiang.Xin, qun.liu}@huawei.com,
{sunchengjie, jizhenzhou, liubq}@hit.edu.cn

Abstract

Collecting supporting evidence from large corpora of text (e.g., Wikipedia) is of great challenge for open-domain Question Answering (QA). Especially, for multi-hop open-domain QA, scattered evidence pieces are required to be gathered together to support the answer extraction. In this paper, we propose a new retrieval target, **hop**, to collect the hidden reasoning evidence from Wikipedia for complex question answering. Specifically, the hop in this paper is defined as the combination of a hyperlink and the corresponding outbound link document. The hyperlink is encoded as the mention embedding which models the structured knowledge of how the outbound link entity is mentioned in the textual context, and the corresponding outbound link document is encoded as the document embedding representing the unstructured knowledge within it. Accordingly, we build HopRetriever which retrieves hops over Wikipedia to answer complex questions. Experiments on the HotpotQA dataset demonstrate that HopRetriever outperforms previously published evidence retrieval methods by large margins. Moreover, our approach also yields quantifiable interpretations of the evidence collection process.

1 Introduction

Multi-hop QA (Yang et al. 2018) is the Question Answering (QA) task that requires reasoning over multiple supporting documents to extract the final answer. For the open-domain setting, a key part of Multi-hop QA is to retrieve an evidence path from the whole knowledge source (e.g., Wikipedia). Most of the recent works view multi-hop evidence collection as an iterative document retrieval problem (Asai et al. 2020; Feldman and El-Yaniv 2019; Das et al. 2019a), which can be decomposed to several single-step document retrieval. In contrast, some others (Dhingra et al. 2020; Ding et al. 2019) focus on mentioned entities and try to traverse textual data like a virtual structured Knowledge Base (KB). These two methods leverage two different kinds of knowledge for evidence collection respectively: (i) informative but unstructured facts inside the introductory documents of entities. (ii) the structured and implicit relations between entities themselves[1].

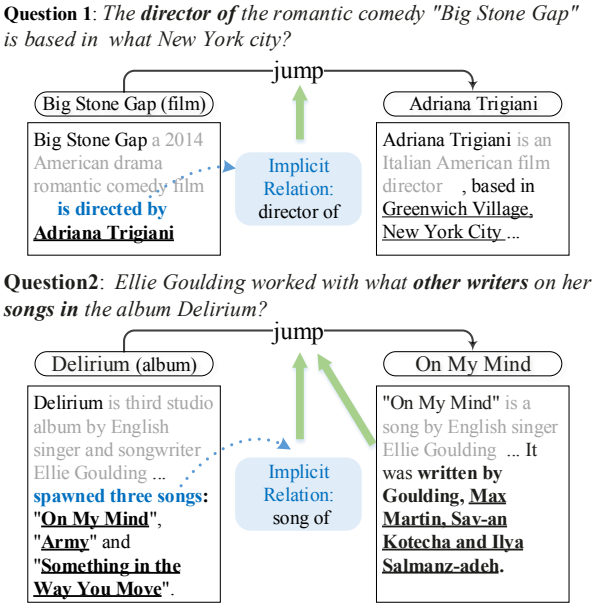


Figure 1: Two examples showing that both structured relation and unstructured fact are needed for complex question answering.

Two examples in Figure 1 show that both of the above knowledge is needed for complex question answering. We consider the problem of based on what evidence can one jump to the second document for further retrieval. For question 1, the structured relation *"directed by"* implied by *"...directed by Adriana Trigiani"* in the first document matches the relation *"director of"* in the question, hence providing sufficient and convincing evidence that one can hop to the introductory document of *Adriana Trigiani* for further retrieval, even without pre-reading it. However, things become complicated for question 2, for three entities share the same relation *"song of"*: *On My Mind*, *Army*, and

*This work was done during an internship at Huawei Noah's Ark Lab.

[1]In this paper, we view the entity relation as structured knowledge is because it directly connects two entities and can be applied to build a structured entity graph.

Something in the Way You Move. In fact, only the entity *On My Mind* satisfies the condition *"works with other writers"* in the question, which makes the relation itself insufficient and indistinctive to make the choice among the three entities. The truth is that only if the unstructured facts about the entity *On My Mind* is browsed through, one can find the conclusive evidence.

As shown above, to collect sufficient supporting evidence within Wikipedia, it's necessary to consider both relational structures between entities and unstructured knowledge hidden inside the introductory document. When the answering process follows the pattern of *"following the vine to get the melon"*, implicit entity-level relation makes retrieval efficient and effective. However, when the relation chain failed, those unstructured facts in the document mount the stage.

In this paper, we study how the structured and unstructured knowledge can be combined together and collaboratively contribute to the evidence collection. Accordingly, We define a **hop** as the combination of a hyperlink and a corresponding outbound link document. A hyperlink in Wikipedia implies how the introductory document of an entity mentions some other, while the outbound link document stores all the unstructured facts and events, which makes a hop contain both relational and factoid evidence for future retrieval.

One challenge is how to transform the binary (link or not) hyperlink in Wikipedia to distributed representations implying the implicit and complicated entity relation. One step towards this is the recent work on distributed relation learning (Soares et al. 2019), in which the relation representations are learned solely from the entity-linked text in an unsupervised way. With the powerful ability of BERT (Devlin et al. 2019) for text encoding, (Ding et al. 2019) and (Dhingra et al. 2020) encodes entity spans into node representations to conduct relation-based reasoning. In this paper, we represent each hyperlink with the corresponding entity mention, with the currently described entity as the mention subject and the outbound link entity as the mention object.

Our contributions. To be more specific, this paper introduces HopRetriever, a method to automatically and adaptively leverage both the structured entity relation and unstructured introductory facts for evidence collection. For each entity mention within Wikipedia documents, we encode the textual context around it into mention embedding to represent the implicit structured knowledge. As for the representation of unstructured knowledge in documents, we use BERT to encode document text conditioned on the original question, as previous works do. For each step retrieval, the hop from one document(entity) to another one can gather evidence from two perspectives: (i) How the current document mentions the other one. (ii) What facts are hidden in the introductory document of the other entity. Experiments conclude that our retrieval method outperforms both entity-centric retrieval methods and document-wise retrieval ones.

Our prime contributions are as follows:

- We propose to retrieve hops over Wikipedia to answer complex questions, which adaptively and selectively collects evidence from both structured entity relation and unstructured facts within documents.
- We propose to represent hyperlinks in Wikipedia with mention embeddings, which we show can precisely capture the implicit relation between entities.
- Evaluated on HotpotQA (Yang et al. 2018), the proposed approach significantly outperforms previously published evidence retrieval methods. Additionally, we conduct further experimental analysis and demonstrate the good interpretability of our method.

2 Related Works

Document-wise reasoning. Most current open-domain QA methods directly retrieve documents for evidence collection. Chen et al. (2017), Lee et al. (2019), Karpukhin et al. (2020), and Nie et al. (2019) leverage sparse methods like BM25 or fully trainable models to retrieve candidates from the whole Wikipedia collection. Such approaches, however, find evidence documents independently without knowing the previous retrieved ones, which may cause retrieval failures when one of the evidence documents has a little semantic relationship with the original question. Avoiding this, Feldman and El-Yaniv (2019) and Das et al. (2019a) introduce multi-step retrievers to explore multiple evidence documents iteratively. Most recently, Asai et al. (2020) proposes the PathRetriever that retrieves documents paths along the outbound-link of text graph. With the graph structure of the documents, PathRetriever reduces the search space of documents during each step retrieval, which is much smaller than that of previous iterative retrievers. The biggest difference between PathRetriever and our method is that we additionally consider the structured and multi-valued relation between entities, while PathRetriever uses hyperlinks in a binary way: link or not link.

Entity-centric reasoning. Considering that most factoid QA problems are entity-centric, some other works focus on the entity mention to collect reasoning evidence. Cognitive Graph (Ding et al. 2019) trains a reading comprehension model to predict the next-hop spans, aiming to find the most evidential mentioned entity. Similarly, DrKIT (Dhingra et al. 2020) constructs large mounts of entity mentions from the corpus and proposes a method to reason over these mentions, softly following paths of latent relations. We've shown in Figure 1 that when the question is not the case of *"following the vine to get the melon"*, the mention itself fails to provide sufficient reasoning evidence for which entity to hop. Inspired by the idea of pseudo-relevance feedback (Xu and Croft 2017), Das et al. (2019b) also leverages entity-link to find more supporting evidence. However, this method is still document-level, for the entity links are used not for relation representation, but document expansion. We empirically show significant improvement over the above methods.

Question decomposition. Wolfson et al. (2020), Perez et al. (2020), and Min et al. (2019) propose to decompose a complicated question into several simpler sub-questions and conduct single-hop QA at each step. The challenge for question decomposition is to ensure each sub-question collects the truly necessary evidence. As we know from example 2 in Figure 1, when the structured relation fails, one can not ask reasonable sub-question without exploring enough in-

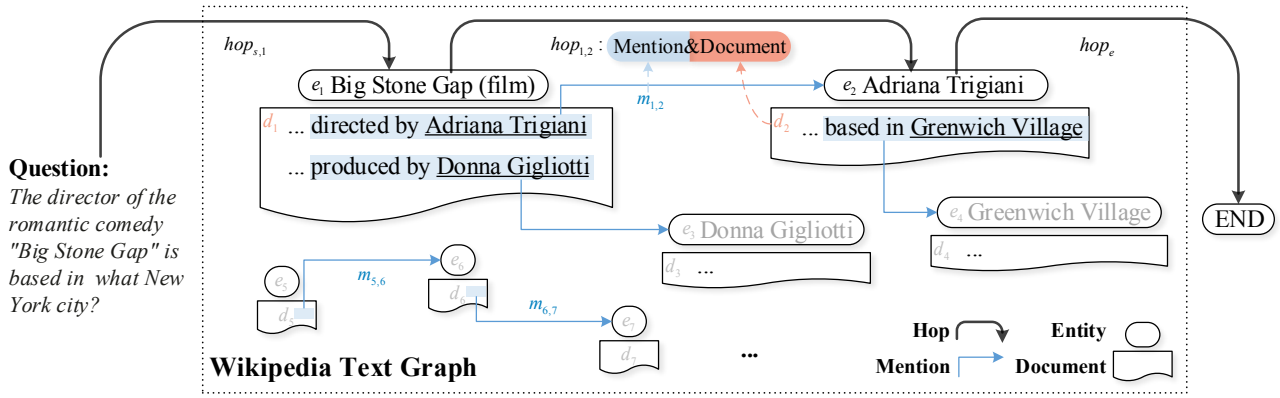


Figure 2: Retrieving Hops over Wikipedia text graph. Documents are retrieved by selecting hops over them iteratively. Each directed arrow implies a mention $m_{i,j}$, which reveals how e_i mentions e_j in the document d_i. Hops between entities are indicated by curved arrows. If the mention $m_{i,j}$ exists between e_i and e_j, the hop $hop_{i,j}$ is represented based on both $m_{i,j}$ and the introductory document d_j for retrieval or based on the d_j solely if no mentions exist.

troductory documents at the next step.

3 Method

3.1 Overview

Task definition. Our task is to obtain the answer a for an open-domain multi-hop question q. A retriever model Retriever is used to collect the multiple evidence pieces over a large-scale knowledge source K:

$$D_q = \text{Retriever}(q, K). \quad (1)$$

D_q should contain multiple documents that are necessary for answering the multi-hop question. All textual facts in D_q and q are concatenated together and fed into a answer extraction model Reader to obtain the answer a:

$$a = \text{Reader}(q, D_q). \quad (2)$$

Our approach. In this paper, we propose HopRetriever to take the place of the retriever model Retriever while keeping the answer extraction model Read as standard (Devlin et al. 2019). The knowledge source K is constructed from Wikipedia[2]. Each Wikipedia page corresponds to an entity e_i, accompanied by an introductory document d_i. Moreover, if there exists an anchor text in d_i linked to e_j, we denote it as a mention $m_{i,j} = e_i \xrightarrow{d_i} e_j$, which means e_j is mentioned by e_i via d_i. Accordingly, the knowledge source is formulated as $K = \{D, E, M\}$ that consists of an entity set $E = \{e_i\}$, an introductory document set $D = \{d_i\}$, and a mention set $M = \{m_{i,j}\}$.

D_q is retrieved iteratively. At each retrieval step, a document is fetched by examining not only the unstructured facts contained in but also the mention of it in the latest selected document. To achieve that, we encode the unstructured textual facts and the mention respectively and then represented them together within a hop. HopRetriever uses hops as the matching objects when retrieving over Wikipedia. The overview of a retrieval process is shown in Figure 2. The details about the hop encoding and the iterative retrieval procedure are discussed in the following two sections.

[2]https://en.wikipedia.org

3.2 Hop Encoding

HopRetriever considers retrieving a new document d_j conditioning on the retrieval history as finding the proper hop from the current foothold entity e_i to the entity e_j. The representation of each hop consists of mention embedding $\mathbf{m}_{i,j}$ that implies the entity relation from e_i to e_j, and the document embedding $\mathbf{u}_j$ of the introductory document of entity e_j.

Mention embedding. We consider the problem of how to encode a hop $hop_{i,j}$ into hop encoding $\mathbf{hop}_{i,j}$. The structured entity relation revealed by $m_{i,j}$ is encoded as mention embedding $\mathbf{m}_{i,j}$, based on the context around it. Inspired by Soares et al. (2019), two entity markers clipping the anchor text of each mentioned entity are introduced to obtain the mention embedding. An example is shown in Figure 3 (from the second example in Figure 1), the document that contains the mention of *On My Mind* is fed into BERT with two additional [MARKER] tokens, and the output representation of the first [MARKER] token is used as the mention embedding vector.

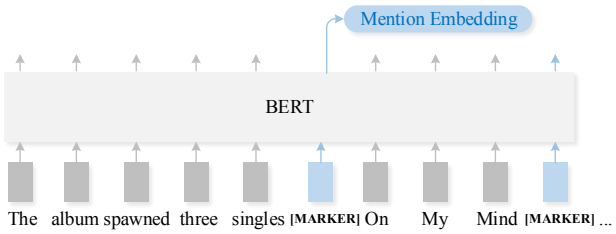


Figure 3: Encoding the mention using entity markers.

If e_j is not mentioned directly in the introductory document of e_i, we represent the relation between them with a trainable uniformed vector $\mathbf{m}_{\text{P}}$, as shown below:

$$\mathbf{m}_{i,j} = \begin{cases} \text{BERT}_{[\text{M-}j]}(q; d_i), & \text{if } m_{i,j} \in M \\ \mathbf{m}_{\text{P}}, & \text{otherwise} \end{cases} \quad (3)$$

where the $\text{BERT}_{[\text{M-}j]}$ is the representation of the entity marker corresponding to entity e_j.

Document embedding. The unstructured knowledge about the entity e_j is encoded as document embedding $\mathbf{u}_j$ by feeding the textual facts in d_j (concatenated with q) into BERT, and the output representation of the [CLS] token is taken as the document embedding vector:

$$\mathbf{u}_j = \text{BERT}_{[\text{CLS}]}(q; d_j). \tag{4}$$

Knowledge fusion. The mention embedding $\mathbf{m}_{i,j}$ and the document embedding $\mathbf{u}_j$ are fused together as hop encoding $\mathbf{hop}_{i,j}$ by the attention mechanism proposed in Sukhbaatar et al. (2015). The following fusion procedure allows HopRetriever to adaptively and selectively manage the two kinds of knowledge according to which truly matters:

$$\begin{aligned} a_m &= \mathbf{h}\mathbf{W}_k\mathbf{m}_{i,j} \\ a_u &= \mathbf{h}\mathbf{W}_k\mathbf{u}_j \\ \{w_m, w_u\} &= \text{softmax}(\{a_m, a_u\}) \\ \mathbf{hop}_{i,j} &= w_m \cdot \mathbf{W}_v\mathbf{m}_{i,j} + w_u \cdot \mathbf{W}_v\mathbf{u}_j, \end{aligned} \tag{5}$$

where $\mathbf{h}$ is the vector that encodes the corresponding retrieval history, the $\mathbf{W}_k$ projects the two embedding vectors (*i.e.* $\mathbf{m}_{i,j}$ and $\mathbf{u}_j$) into *key* vectors. The $\mathbf{h}$ acts as *query* vector that interacts with the *key* vectors to calculate the importance weight w_m for the mention embedding $\mathbf{m}_{i,j}$ and w_u for the document embedding $\mathbf{u}_j$, then $\mathbf{m}_{i,j}$ and $\mathbf{u}_j$ are projected into *value* vectors by $\mathbf{W}_v$ and fused as hop encoding with important weights.

3.3 Iterative Retrieval of Hops

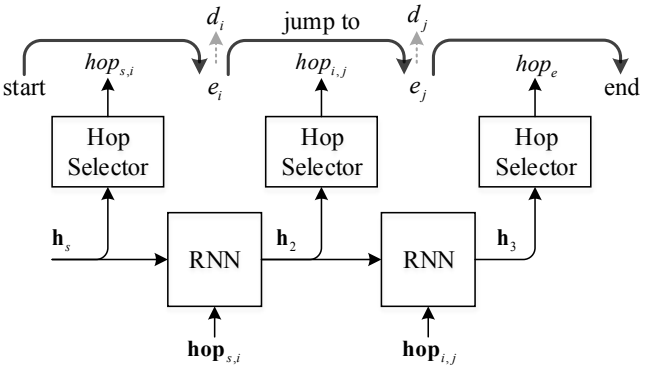


Figure 4: The retrieval process of HopRetriever for three hops. $hop_{s,i}$ indicates a beginning jump from the start to e_i is selected based on the initial hidden state $\mathbf{h}_s$. The selection of hop $hop_{i,j}$ retrieves the supporting document d_j at the second step. hop_e ends the retrieval process finally.

Figure 4 illustrates a three-step recurrent hop retrieval process. Generally, let e_i denote the foothold entity selected at the previous $t-1$ step, the probability of retrieving the document d_j at t step is calculated by the dot product of $\mathbf{h}_t$ and hop encoding $\mathbf{hop}_{i,j}$ (*i.e.* the Hop Selector in Figure 4),- as formulated in the following equation:

$$p(d_j) = \text{sigmoid}(\mathbf{h}_t^\top \mathbf{hop}_{i,j}), \tag{6}$$

where $\mathbf{h}_t$ is the hidden state vector that encodes all the previously selected hops by a Recurrent Neural Network (RNN):

$$\mathbf{h}_t = \begin{cases} \mathbf{h}_s, & t = 1 \\ \text{RNN}(\mathbf{h}_{t-1}, \mathbf{hop}_{k,i}), & t \geq 2 \end{cases} \tag{7}$$

where $\mathbf{h}_s$ is the initial hidden state vector and $\mathbf{hop}_{k,i}$ is the encoding of the hop selected at $t-1$ step. Specially, for $t=1$, the hop $hop_{s,j}$ indicating jumping from the retrieving start to e_j is introduced. Similarly, a special end hop hop_e is used to mark the end of the retrieval process and it is encoded by $\mathbf{m}_p$ and a virtual end document encoding $\mathbf{u}_e$. Let f denote the fusion function formulated as Equation (5), the encodings of different hops are summarized in Table 1.

Notation	Encoding	Explanation
$hop_{i,j}$	$f(\mathbf{m}_{\text{P}}, \mathbf{u}_j)$	e_j is not mentioned in d_i
	$f(\mathbf{m}_{i,j}, \mathbf{u}_j)$	e_j is mentioned in d_i
$hop_{s,j}$	$f(\mathbf{m}_{\text{P}}, \mathbf{u}_j)$	Select d_j at the beginning
hop_e	$f(\mathbf{m}_{\text{P}}, \mathbf{u}_e)$	Retrieval finish

Table 1: Types of hop encoding.

3.4 Fine-Grained Sentence-Level Retrieval

A single supporting document can be split into multiple sentences and may not all these sentences are essential for answering the question. Pointing out the indispensable supporting sentences can illuminate the reasons why a document is required. In HopRetriever, the supporting sentence prediction is added as an auxiliary task along with the primary hop retrieval task. At step t, the probability $p(s_{i,l})$ that indicates the l-th sentence in the latest retrieved document d_i is a supporting sentence is calculated by the following equations:

$$\mathbf{s}_{i,l} = \text{BERT}_{[\text{SM-}l]}(q; d_i) \tag{8}$$

$$p(s_{i,l}) = \text{sigmoid}(\mathbf{h}_t\mathbf{W}_s\mathbf{s}_{i,l}), \tag{9}$$

where $\mathbf{s}_{i,l}$ is the sentence embedding vector obtained by inserting a sentence marker [SM-l] at the end of the l-th sentence in d_i, which is similar to how the mention embedding is obtained. If $p(s_{i,l}) > 0.5$, then the l-th sentence in document d_i is identified as a supporting sentence.

3.5 Objective Functions of HopRetriever

HopRetriever is a sequence prediction model with binary cross-entropy objective functions at each step. At the retrieval step t, the objective function of the primary hop retrieval task is

$$\log p(d_j) + \sum_{\bar{d}_j \in D, \bar{d}_j \neq d_j} \log(1 - p(\bar{d}_j)), \tag{10}$$

where d_j is the ground-truth document. For the auxiliary supporting sentence prediction task, the object function at step t is

$$\sum_{l \in L_i} \log p(s_{i,l}) + \sum_{l \notin L_i} \log(1 - p(s_{i,l})), \tag{11}$$

where $s_{i,l}$ is the l-th sentence in d_i, L_i is the set of indices of the ground-truth supporting sentences in d_i. The above two objective functions are maximized together in training.

Model	Ans exists			Sent exists			All docs exist		
	top-1	top-5	top-8	top-1	top-5	top-8	top-1	top-5	top-8
Cognitive Graph QA (Ding et al. 2019)	72.21	-	-	70.25	-	-	57.80	-	-
Semantic Retrieval* (Nie et al. 2019)	77.84	85.96	86.39	81.68	88.35	88.50	69.35	81.73	82.07
PathRetriever (Asai et al. 2020)	80.96	89.09	89.98	82.05	88.69	89.48	73.91	86.12	87.39
HopRetriever	**86.89**	**91.11**	**91.80**	**88.41**	**92.78**	**93.20**	**82.54**	**88.60**	**89.09**

Table 2: Evidence collection result on the HotpotQA fullwiki development set. We compare the top-1, top-5, and top-8 output document sequences from different retrievers using respectively. Cognitive Graph QA produces one document sequence for each question. HopRetriever and PathRetriever output the top-8 document sequences by the adoption of beam search. Semantic Retrieval* ranks documents instead of document sequences, so we assemble the top-2, top-10, and top-16 output documents into the top-1, top-5, and top-8 document sequences respectively for better fairness.

4 Experiments

4.1 Setup

Dataset. HopRetriever is evaluated on the multi-hop question answering dataset HotpotQA (Yang et al. 2018), which includes 90,564 question-answer pairs with annotated supporting documents and sentences for training, 7,405 question-answer pairs for development, and 7,405 questions for testing. All the testing questions correspond to multiple supporting documents. We focus on the *fullwiki* setting of HotpotQA, in which the supporting documents for each question are scattered among almost 5M Wikipedia pages. In the official evaluation, the participant model is required to predict both the exact supporting sentences and the answer text.

Pipeline. The whole procedure follows a coarse-to-fine pipeline that contains three stages:

1. Preliminary retrieval: Only the top-500 documents are used to construct the initial candidate hops of HopRetriever, according to the TF-IDF scores of documents w.r.t. the input question.
2. Supporting documents retrieval and supporting sentence prediction: HopRetriever retrieves the supporting documents iteratively starting from the initial candidate hops, and also predicts supporting sentences from the retrieved documents.
3. Answer extraction: The answer within the retrieved supporting documents is extracted using BERT (large, whole word mask), following the conventional answer boundary prediction approach (Devlin et al. 2019; Seo et al. 2017), which is the same as PathRetriever (Asai et al. 2020).

Implementation details. The negative hop sequences used to train the proposed model are constructed by traversing through the entities in Wikipedia. And the top-40 TD-IDF scored documents w.r.t. the question and top-10 scored documents w.r.t. the ground-truth documents are used as the start points of the traverse.The length of negative hop sequences is fixed to 3. We restrict the maximum input sequence length of BERT to 384. In training, the batch size is set to 16, the learning rate is 3×10^{-5}, and the number of training epochs is 3. We use beam search with beam size set to 8 at the inference time.

4.2 Results

Evidence collection. The HopRetriever is first evaluated by measuring the coverage of ground-truth answers, supporting sentences, and supporting documents in the retrieved supporting documents, as shown in Table 2. The metric Ans exists measures the percentage of the questions whose answers are extractable from the retrieved document sequence. Sent exists is the percentage of the supporting sentences that can be found. The percentage of the questions that have all ground-truth documents retrieved are showed as the All docs exist.

Three models that mainly focus on evidence collection over Wikipedia are evaluated as baselines on the development set:

- **Cognitive Graph QA** (Ding et al. 2019) explicitly utilizes the structured knowledge in Wikipedia with a graph whose nodes are entities or answer spans. The representations of nodes are maintained by Graph Neural Network (GNN) (Battaglia et al. 2018; Kipf et al. 2017).
- **Semantic Retrieval** (Nie et al. 2019) is a multi-grained retrieval baseline that retrieves supporting documents and sentences together, focuses on the unstructured knowledge in Wikipedia.
- **PathRetriever** (Asai et al. 2020) introduces a similar iterative retrieval framework, but only focuses on the unstructured knowledge provided in the introductory document at each retrieval step.

To be fairly compared with PathRetriever, which is the state-of-the-art published model, HopRetriever uses the same initial search space (*i.e.* top-500 documents based on TF-IDF scores) and pre-trained model (*i.e.* BERT-base) with PathRetriever. Notably, HopRetriever outperforms the PathRetriever by 5.93%, 6.36%, and 8.63% on the top-1 evidence collection metrics respectively, and also achieves significant improvement over Semantic Retriever and Cognitive Graph QA, which further demonstrates the effectiveness of HopRetriever.

A more detailed comparison with PathRetriever is shown in Table 3. We can observe that HopRetrieve works more effectively on the bridging questions. In the HotpotQA dataset, the ground-truth supporting documents of comparison questions may not be directly relevant to each other where no structured knowledge is available, which makes HopRetriever perform almost the same as PathRetriever. In con-

Model	Ans exists			Sent exists			All docs exist		
Recall @	top-1	top-5	top-8	top-1	top-5	top-8	top-1	top-5	top-8
PathRetriever (Comparison)	77.00	**81.17**	82.25	88.33	90.36	90.62	**86.42**	**89.58**	**90.38**
HopRetriever (Comparison)	**77.40**	80.97	**82.31**	**91.31**	**92.73**	**92.79**	84.26	85.41	85.41
PathRetriever (Bridging)	81.95	91.08	91.92	80.56	88.29	89.20	70.77	85.25	86.63
HopRetriever (Bridging)	**89.27**	**93.66**	**94.19**	**87.73**	**92.79**	**93.29**	**82.11**	**89.41**	**90.01**

Table 3: Evidence collection results on different types of questions.

	Model	Ans		Sup		Joint	
		EM	F1	EM	F1	EM	F1
dev	Cognitive Graph QA (Ding et al. 2019)	37.55	49.40	23.11	58.52	12.18	35.28
	Semantic Retrieval (Nie et al. 2019)	46.41	58.70	39.86	71.53	26.53	49.00
	PathRetriever (Asai et al. 2020)	60.49	73.30	49.16	76.05	35.82	61.43
	HopRetriever	**62.07**	**75.18**	**52.53**	**78.92**	**37.81**	**64.50**
test	DecompRC (Min et al. 2019)	30.00	40.65	-	-	-	-
	Cognitive Graph QA (Ding et al. 2019)	37.12	48.87	22.82	57.69	12.42	34.92
	DrKIT (Dhingra et al. 2020)	42.13	51.72	37.05	59.84	24.69	42.8
	Semantic Retrieval (Nie et al. 2019)	45.32	57.34	38.67	70.83	25.14	47.60
	Transformer-XH (Zhao et al. 2019)	51.60	64.07	40.91	71.42	26.14	51.29
	PathRetriever (Asai et al. 2020)	60.04	72.96	49.08	76.41	35.35	61.18
	Semantic Retrieval + HGN (Fang et al. 2020)	59.74	71.41	51.03	77.37	37.92	62.26
	HopRetriever	**60.83**	**73.93**	**53.07**	**79.26**	**38.00**	**63.91**

Table 4: Answer extraction and supporting sentence prediction result in the fullwiki setting of HotpotQA.

trast, the ground-truth supporting documents of the bridging questions are stringed with mentions that can provide informative structured knowledge, so HopRetriever performs better by leveraging mentions additionally.

Answer extraction and supporting sentence prediction. Table 4 shows the performance of different methods for the answer and supporting sentence prediction. Naturally, the answer extraction and supporting sentence prediction result benefit from the improvements of document retrieving. By providing more accurate supporting documents, HopRetriever outperforms all the aforementioned baseline models on the development set and also the other published models[3] on the test set.

4.3 Analysis

Detailed analysis of HopRetriever is carried out in this section, especially about how the structured and unstructured knowledge in Wikipedia contribute to evidence retrieval.

Embedding weights on different question types. At the t step in the retrieval procedure of HopRetriever, the decision whether to select a document d_j depends on the hop encoding $\mathbf{hop}_{i,j}$, which contains a mention embedding and a document embedding assigned with learnable weights as formulated in Equation (5). We analyze the weights and find that they provide intuitive explanation about which embedding is more important for different question types. Table 5 shows the average weight of mention embedding and document embedding on different question types.

It can be seen that the mention embedding accounts for a large portion (89.53%) on the bridge questions. The bridge questions always require selecting a proper document along with a hyperlink and the mentions do provide helpful information for bridging the evidence pieces. Conversely, when processing the comparison questions, the weight of mention embedding is relatively small (4.61%) because there are no available mentions between the supporting documents.

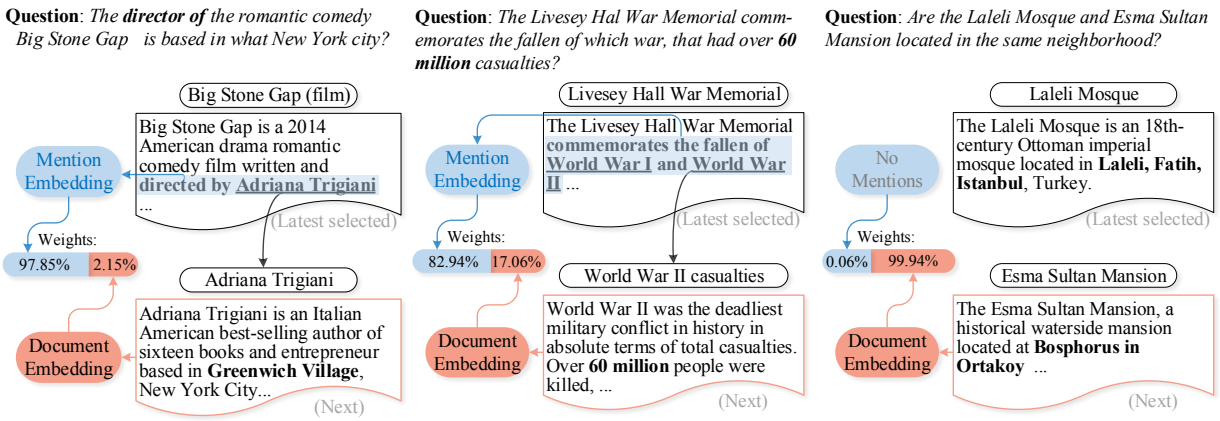
Question Type	Mention	Document
Bridging	89.53%	10.47%
Comparison	4.61%	95.39%

Table 5: Weights of mention embedding and document embedding on bridging questions and comparison questions.

Embedding weights in different cases. Three examples are presented in Figure 5 to further inspect the learned weights in the hop encoding. In case 1, a strong clue that matches with the information *"director of"* in question occurs as the mention *"directed by"*, so the weight of mention embedding is relatively high. In case 2, the entity *"World War I"* and *"World War II"* are mentioned with the same context, which means they cannot be distinguished only based on the mention embedding, so more attention is paid to the document embedding which encodes the important fact *"60 million"*. In case 3, no mentions exist in the latest selected document so the hop encoding almost completely depends on the document embedding. We can see that the embedding weights can bring intuitive interpretation about which embedding, or which types of knowledge, is more important for different questions when selecting a hop.

Probing task for the mention embedding. The structured entity relation is represented by the markers around the mentions, as described in Section 3.2. To explore what

[3]By the submission time of this paper, recently published method on HotpotQA fullwiki leaderboard is PathRetriever.



(a) Case 1. (b) Case 2. (c) Case 3.

Figure 5: The weights of mention embedding and document embedding in different cases.

Model	Ans exists			Sent exists			All docs exist		
Recall @	top-1	top-5	top-8	top-1	top-5	top-8	top-1	top-5	top-8
full	86.89	91.11	91.80	88.41	92.78	93.20	82.54	88.60	89.09
1. w/o structured knowledge	76.35	86.02	88.12	80.91	88.49	89.92	66.20	78.89	81.23
2. w/o weighting	86.21	91.07	91.52	87.73	92.55	93.09	81.38	88.09	88.70
3. w/o sentence prediction	86.58	90.88	91.51	87.98	92.54	92.98	82.03	88.29	88.89

Table 6: Ablation experiments of HopRetriever.

the mention embedding learns, we design a special probing task: *distracted hop selection*. That is, the ground-truth hop for bridging questions is shuffled with other hops that have the same mentioned entity but different mention context, and HopRetriever is required to select the right one from these distracting hops for each question. To make the right selection, one should understand more about how each entity is mentioned, but not the entity itself. The summary of this task is shown in Table 7. The experiment result shows that although the distracting hops are not used as negative samples for training, the HopRetriever can retrieve ground-truth hops just based on learned mention embedding at high accuracy (96.42%), indicating that the mention embedding does learn the implicit relation between entities, but not the entities themselves.

Total questions in development set	7405
Number of the bridging questions	5918
Average number of distracting hops per question	52.20
Accuracy based on mention embedding	96.42%

Table 7: Summary of the mention embedding probing task.

Ablation study. As shown in Table 6, ablation experiments are conducted to corroborate the effectiveness of HopRetriever. In experiment 1, the structured knowledge in hops is removed (*i.e.* set the weight of mention embedding w_m to 0 in Equation 5), the performance dropped significantly, which stresses the importance of structured knowledge in Wikipedia for multi-hop evidence retrieval. The performance also degraded in experiment 2 in which the weighting for the structured and unstructured knowledge in hops is disabled (*i.e.* set $w_m = w_u = 1$ in Equation 5), demonstrating that the fusion function improves the performance while providing interpretations. The auxiliary supporting sentence prediction task is removed in the experiment 3. The result shows that the auxiliary task has no side-effect on the primary hop retrieval task. Additionally, the sentence representations are obtained by the sentence markers contained in the latest retrieved document which has been encoded already at the previous step. So the auxiliary task does not require much additional computation.

5 Conclusion

In this paper, we propose the HopRetriever to collect reasoning evidence over Wikipedia for multi-hop question answering. Both the structured knowledge indicated by hyperlinks and the unstructured knowledge presented as introductory documents in Wikipedia, are involved and leveraged together in HopRetriever to help the evidence collection. The experiment on the HotpotQA dataset shows that the performance of HopRetriever improved observably as a result of combining the structured knowledge with unstructured knowledge, and outperforms all the published models on the leaderboard. Moreover, by inspecting the proportion of the two kinds of knowledge in hops, which kind of knowledge leads the retrieving of each evidence piece can be observed directly, which also provides extra intuitive interpretations for the selection of each evidence.

References

Asai, A.; Hashimoto, K.; Hajishirzi, H.; Socher, R.; and Xiong, C. 2020. Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL https://openreview.net/forum?id=SJgVHkrYDH.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V. F.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gülçehre, Ç.; Song, H. F.; Ballard, A. J.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K. R.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational Inductive Biases, Deep Learning, and Graph Networks. *CoRR* abs/1806.01261. URL http://arxiv.org/abs/1806.01261.

Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017. Reading Wikipedia to Answer Open-Domain Questions. In Barzilay, R.; and Kan, M., eds., *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 1870–1879. Association for Computational Linguistics. doi:10.18653/v1/P17-1171. URL https://doi.org/10.18653/v1/P17-1171.

Das, R.; Dhuliawala, S.; Zaheer, M.; and McCallum, A. 2019a. Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL https://openreview.net/forum?id=HkfPSh05K7.

Das, R.; Godbole, A.; Kavarthapu, D.; Gong, Z.; Singhal, A.; Yu, M.; Guo, X.; Gao, T.; Zamani, H.; Zaheer, M.; and McCallum, A. 2019b. Multi-step Entity-centric Information Retrieval for Multi-Hop Question Answering. In Fisch, A.; Talmor, A.; Jia, R.; Seo, M.; Choi, E.; and Chen, D., eds., *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China, November 4, 2019*, 113–118. Association for Computational Linguistics. doi:10.18653/v1/D19-5816. URL https://doi.org/10.18653/v1/D19-5816.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186. Association for Computational Linguistics. doi:10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Dhingra, B.; Zaheer, M.; Balachandran, V.; Neubig, G.; Salakhutdinov, R.; and Cohen, W. W. 2020. Differentiable Reasoning over a Virtual Knowledge Base. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL https://openreview.net/forum?id=SJxstlHFPH.

Ding, M.; Zhou, C.; Chen, Q.; Yang, H.; and Tang, J. 2019. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, 2694–2703. Association for Computational Linguistics. doi:10.18653/v1/p19-1259. URL https://doi.org/10.18653/v1/p19-1259.

Fang, Y.; Sun, S.; Gan, Z.; Pillai, R.; Wang, S.; and Liu, J. 2020. Hierarchical Graph Network for Multi-hop Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8823–8838.

Feldman, Y.; and El-Yaniv, R. 2019. Multi-Hop Paragraph Retrieval for Open-Domain Question Answering. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, 2296–2309. Association for Computational Linguistics. doi:10.18653/v1/p19-1222. URL https://doi.org/10.18653/v1/p19-1222.

Karpukhin, V.; Oguz, B.; Min, S.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *CoRR* abs/2004.04906. URL https://arxiv.org/abs/2004.04906.

Kipf, T. N.; et al. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Lee, K.; et al. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, 6086–6096. Association for Computational Linguistics. doi:10.18653/v1/p19-1612. URL https://doi.org/10.18653/v1/p19-1612.

Min, S.; Zhong, V.; Zettlemoyer, L.; and Hajishirzi, H. 2019. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 6097–6109.

Nie, Y.; et al. 2019. Revealing the Importance of Semantic Retrieval for Machine Reading at Scale. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, 2553–2566. Association for Computational Linguistics. doi:10.18653/v1/D19-1258. URL https://doi.org/10.18653/v1/D19-1258.

Perez, E.; Lewis, P.; Yih, W.-t.; Cho, K.; and Kiela, D. 2020. Unsupervised Question Decomposition for Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8864–8880.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional Attention Flow for Machine Comprehension. In *International Conference on Learning Representations*.

Soares, L. B.; FitzGerald, N.; Ling, J.; and Kwiatkowski, T. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, 2895–2905. Association for Computational Linguistics. doi: 10.18653/v1/p19-1279. URL https://doi.org/10.18653/v1/p19-1279.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end Memory Networks. In *Advances in neural information processing systems*, 2440–2448.

Wolfson, T.; Geva, M.; Gupta, A.; Gardner, M.; Goldberg, Y.; Deutch, D.; and Berant, J. 2020. Break It Down: A Question Understanding Benchmark. *Transactions of the Association for Computational Linguistics* 8: 183–198.

Xu, J.; and Croft, W. B. 2017. Quary Expansion Using Local and Global Document Analysis. *SIGIR Forum* 51(2): 168–175. doi:10.1145/3130348.3130364. URL https://doi.org/10.1145/3130348.3130364.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Riloff, E.; Chiang, D.; Hockenmaier, J.; and Tsujii, J., eds., *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, 2369–2380. Association for Computational Linguistics. doi:10.18653/v1/d18-1259. URL https://doi.org/10.18653/v1/d18-1259.

Zhao, C.; Xiong, C.; Rosset, C.; Song, X.; Bennett, P.; and Tiwary, S. 2019. Transformer-XH: Multi-Evidence Reasoning with eXtra Hop Attention. In *International Conference on Learning Representations*.