Triple-Fact Retriever: An explainable reasoning retrieval model for multi-hop QA problem

Chengmin Wu *, Enrui Hu †, Ke Zhan†, Lan Luo†, Xinyu Zhang†, Hao Jiang†, Qirui Wang†, Zhao Cao†, Fan Yu†, and Lei Chen *
*Department of Computer Science and Engineering, Hong Kong University of Science and Technology
†Distributed and Parallel Software Lab, Huawei
Email: {cwuap, leichen}@cse.ust.hk,
{huenrui1, zhanke2, luolan13, zhangxinyu35, jianghao66, wangqirui1, fan.yu, caozhao1}@huawei.com

Abstract*—Nowadays, multi-hop question answer (QA) problem is challenging and not well solved in the QA community. The dominant bottleneck of the multi-hop QA problem is the need for a reasoning retriever to fetch a document path from an open-domain corpus (*e.g.*, Wikipedia). A reasoning retriever needs to collect an evidence document from large corpora at one hop retrieval and aggregate the evidence for subsequent hop retrieval, which yields a document path after multi-hop retrieval. There exist two challenges, (1) to fetch the evidence document in an efficient and explainable way at one hop retrieval and (2) to update the question information by aggregating the evidence from the retrieved document after each hop retrieval. To address these two challenges, we propose a triple-fact-based retrieval model to effectively retrieve a related document path in an explainable way for each question. We extract a structured representation from the unstructured document and utilize the knowledge of pre-trained language model (PLM) to do the semantic-level matching between the question and document. We evaluate Triple-fact Retriever on the recently proposed open-domain multi-hop QA dataset, HotpotQA, and a cross-document multi-step Reading Comprehension dataset, Wikihop. The results[1] demonstrate that the Triple-fact retriever outperforms the existing baseline retrieval works.

***Index Terms*—Multi-hop QA, a reasoning retriever, triple-fact based model**

I. Introduction

Multi-hop question answering (QA) problem is answering a human-language multi-hop question in an open-domain scenario. The problem contains two major tasks, (1) Given a human-language multi-hop question, retrieve a document path in an open-domain setting, *i.e.*, fetch multiple scatted evidence documents from Wikipedia corpus [1], [2]. (2) Given the document path, gather the evidence pieces to extract the answer to the question. Recently, the multi-hop QA problem has attracted tremendous attention on both natural language processing (NLP) and information retriever community [1]–[7].

We briefly introduce the two major tasks as follows. **Firstly**, given a natural language multi-hop question, which contains complex logic facts, we need to find a document path composed of a sequence of documents when a single document is not enough to find the correct answer. The retrieval of the document path needs to conduct reasoning retrieval across multiple documents. Take Fig 1 as an illustration example, the question q is a multi-hop question. Unlike the single question answer problem [8], we need a reasoning retriever, which finds a document path *"document 1 - document 2"* from the open-domain corpus. Two pieces of knowledge are required to answer the question, "the football club which Walter Otto Davis played at" and "the foundation time for the football club". From the corpus, document 1 satisfies the first knowledge need, which contains the exact football club name information. Then with the name information, document 2 is retrieved to satisfy the second need of knowledge. **Secondly**, a reader model is applied to extract a set of tokens from the retrieved documents as the answer to the question [3].

Though many QA works investigate the QA problem [1], [8], they work on the second task as they focus on extracting the answer from a small document set. However, the bottleneck of the multi-hop QA problem is the first task, which retrieves the related document path from large-scale open-domain corpus in an explainable and efficient way. Specifically, we locate the concrete semantic matching information in the document to a question in document retrieval. There are two challenges for the document retriever task.

Question-Document matching. The core idea of document retriever task is to find the semantic-level matching information between the question and document. In Fig 1, we retrieve document 1 by matching the semantic information it describes, "Walter Otto Davis played at centre for Millwall" to the question "the football club which Walter Otto Davis played at centre forward". **The primary challenge** behind the document retrieval task is learning an excellent semantic representation for both the document and question [9].

However, the existing widely-used IR algorithms, *e.g.*, TF-IDF [10] and BM25 [11] which perform literal matching have insufficient representation capacity and inadequate for semantic-level matching. These token-based matching algorithm cannot handle semantic understanding problems [12], *e.g.*, synonymy problem – different texts can express the same concept, polysemy problem – words can have different meanings. Some works [3]–[6], [13], [14] propose to utilize the superior understanding ability of the recent pre-trained language model (PLM) [15] to learn a good semantic represen-

[1]The source code is available on our website: https://github.com/Rebaccamin/triple_retriever.

tation for natural language text, *i.e.*, question and document. They encode the full text of the input document and question to dense representations. Specifically, they compress the whole information in the document into a fixed dimension embedding vector for variable length document. The compression vector always maintains information loss and basis to the full-text source document. Furthermore, most information in the document is not related to the question or not helpful in finding the answer [16]. Moreover, these works lack of explainability which is encouraged by the existing multi-hop QA task [28]. The blackbox encoder models select documents based on the distance score of the dense representations of the document and question and cannot locate the exact matching information.

Question update. **The second challenge** for the multi-hop question is to update the question during the multi-hop retrieval efficiently. As the required document share less or even no overlap to the original question, *e.g.*, document 2 has no distinct overlap to the question Q. We need to add the *updater-clue* from document 1, – "Millwall", the football club name, to the question. In this way, document 2 can be retrieved based on this new question $Q^{'}$. It is a challenging problem to select an updater-clue which is the information of some tokens from the retrieved document. Some [3], [17] concatenate the full text information of the retrieved document to update question, which induces large noise information unavoidably. However, the search space of the updater-clue is exponential, $O(2^a)$, while a is the size of document tokens as an updater-clue is a variable token sequence of a document $d = \{w_1, w_2, ..., w_a\}$ [13], [14].

To solve these two challenges, we propose our solution, a triple-fact based retrieval model for the multi-hop question. **To solve the first challenge**, we induce prior knowledge of the input document to reduce the burden of the semantic understanding requirement. We represent the original unstructured document information by structured representation, *i.e.*, a complete-minimized triple fact set T_d of document d. It is challenging to extract a complete-minimized triple fact set for a long-text document. Hierarchical Agglomerative Clustering (HAC) algorithm [18], [19] which is always applied to reduce the size of triple facts needs a high time complexity. HAC algorithm merges pairs of points as one cluster in a "bottom-up" way until the number of clusters satisfies the need and the time complexity is $O(m^3)$ where m is the size of the start triple fact set. Moreover, the information can be loss when selecting a representation point from each cluster. We propose a partition-based triple fact construction algorithm that can generate a complete-minimized triple fact set in $O(m^2)$ with no information loss to address these issues.

We conduct the retrieval on the field of triple fact instead of the full-text of each document, and our retriever is explainable from the triple fact level intuitively. In Fig 1, our retriever selects document 1 by finding the matching information, w.r.t, the blue triple fact t_2 to the question. Compared with existing works [3], [4] which calculate the score of dense document representation, our retriever can alleviate the information encode error of a single dense vector in representing long-text document. Besides, instead of considering the whole information of the document, our retriever depends on the matched triple fact which alleviates the impact of other noise information within the document. Moreover, our retrieval model locates the matching information concretely on some triple facts of a document which can satisfy the explainable need of document retrieval.

To solve the second challenge, we update the question information by choosing a triple fact instead of uncertain-length tokens from the retrieved document to reduce the computation cost. We reduce the updater time complexity from $O(2^a)$ to $O(|T_d|)$. In Fig 1, we select the triple fact t_3 with red color to update the original question and generate a new question Q'.

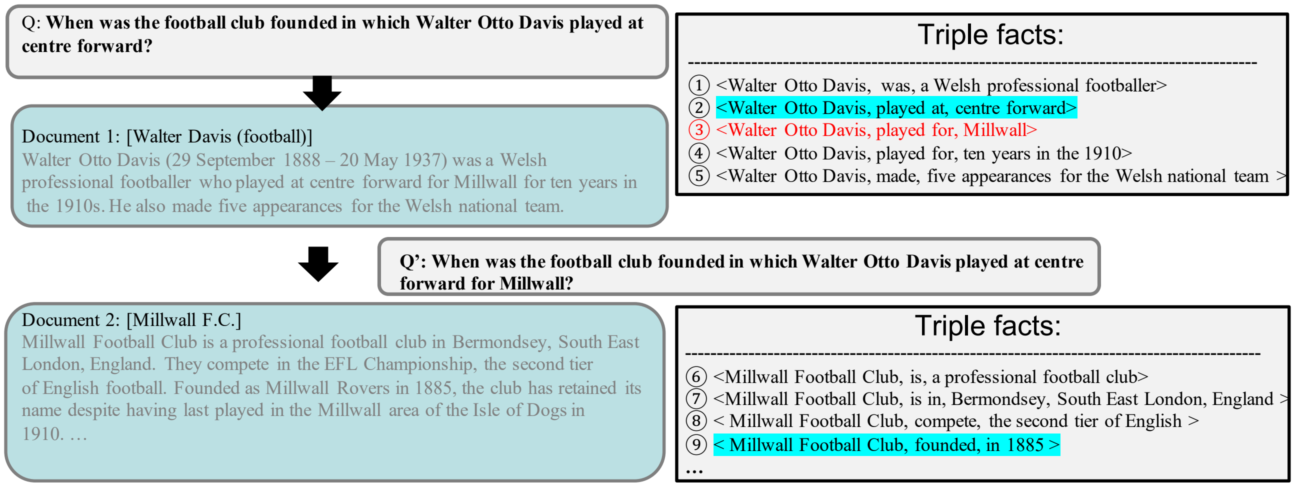
A structured representation is much more readable for machine compared with the unstructured plain text [20]. To the best of our knowledge, this is the first work that leverages the structured knowledge instead of the unstructured plain text to do the document retrieval for the open-domain multi-hop question answer problem. We summarize the novel contributions of this paper as follows.

- To extract a complete-minimized triple fact set T_d representing the document information, we propose an efficient partition-based triple construction algorithm to HAC based method. Our algorithm reduces the time complexity from $O(m^3)$ to $O(m^2)$, where m is the size of the extracted triple fact set by the existing Open Information Extraction tools.
- We propose a triple fact based retriever to retrieve the document in an explainable way. We retrieve document in a less-computation and more explainable way by conducting the matching on the field of the triple fact instead of the full text .
- We propose a triple fact based updater to update the question during multi-hop retriever process. Instead of considering exponential candidates of updater-clue, our search space for the updater-clue is reduced to the size of triple fact set $O(|T_d|)$.
- The experiments on the real-world dataset show that the proposed approach can significantly gain better result compared with the state-of-the-art techniques.

The rest of this paper is organized as follows. In Section II, we formally introduce the relevant concepts and define the document retriever problem for multi-hop question. In Section III, we illustrate our solution, *i.e.*, an end-to-end triple-fact based retrieval model to the multi-hop question in detail which is composed of a triple-fact based retriever and triple-fact based updater. In Section IV, we demonstrate the effectiveness of our triple retriever model through extensive experiments. We discuss the related works in Section V. Finally, we conclude the paper in Section VI.

II. Problem definition

Multi-hop QA contains two sub-tasks, document retriever and document understanding. Document retriever is to retrieve some related documents, which yield a document path


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

Document 1: [Walter Davis (football)]
Walter Otto Davis (29 September 1888 – 20 May 1937) was a Welsh professional footballer who played at centre forward for Millwall for ten years in the 1910s. He also made five appearances for the Welsh national team.

Triple facts:
① <Walter Otto Davis, was, a Welsh professional footballer>
② <Walter Otto Davis, played at, centre forward>
③ <Walter Otto Davis, played for, Millwall>
④ <Walter Otto Davis, played for, ten years in the 1910>
⑤ <Walter Otto Davis, made, five appearances for the Welsh national team >

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

Document 2: [Millwall F.C.]
Millwall Football Club is a professional football club in Bermondsey, South East London, England. They compete in the EFL Championship, the second tier of English football. Founded as Millwall Rovers in 1885, the club has retained its name despite having last played in the Millwall area of the Isle of Dogs in 1910. …

Triple facts:
⑥ <Millwall Football Club, is, a professional football club>
⑦ <Millwall Football Club, is in, Bermondsey, South East London, England >
⑧ < Millwall Football Club, compete, the second tier of English >
⑨ < Millwall Football Club, founded, in 1885 >
…

Fig. 1: An example for document retriever task for a multi-hop question. Given a multi-hop question Q, we need to retrieve a document path *"document 1 - document 2"* from the open domain corpus, *e.g.*, Wikipedia. In the first one hop retrieval, our triple-fact based retriever model retrieves document 1 depending on the semantic matching relationship of question and t_2 in document 1. After the first one hop retrieval, our triple-fact based updater model select t_3 as the update-clue to update the question. In this way, we retrieve document 2 based the semantic matching between t_9 with the new question Q'.

$\hat{D} = \{d_1, d_2, ..., d_r\}$ from a open-domain corpus D to answer a question q. Currently, it is the bottleneck for multi-hop QA problem. Document understanding is then applied to extract the exact answer a from the related document path $\hat{D} = \{d_1, d_2, ..., d_r\}$. This work is focused on the retriever problem.

Definition 1. *(Multi-hop Question) A multi-hop question q refers to a natural language question needs more than one document information to answer. We need to fetch a document path $P_D = \{d_1, d_2, ..., d_n\}$ to answer the question.*

Take the Fig 1 as an example, to answer this multi-hop question, we need to retrieve two documents, *i.e.*, document 1 and document 2. As the question intention is to find the foundation time for the exact football club "Millwall", from the question context, we can only know the information about the club is "Walter Otto Davis played at". Document 1 tells the name of the football club. Document 2 is needed to retrieved as it contains the exact information of the foundation time of the football club.

Definition 2. *(Triple fact) A triple fact denoted as $\langle subject, predicate, object\rangle$ [2] [6]. A triple fact indicates a relationship between subject and object captured by the predicate.*

Triple extractor. For each natural language document d, we aim to extract a triple fact set T_d to cover the information covered by the document. The triple fact set T_d is a structured representation of the document d and composed by a minimal number of triple facts $t =< s, p, o >$ extracted from the document d.

[2]To simplification, we use $\langle s, p, o\rangle$ to denote the triple fact.

Retriever-updater framework. Given a multi-hop question q, our task is to fetch a document path $P_D = \{d_1, d_2, ..., d_n\}$ from open-domain corpus D for answer the question. For the multi-hop question, as the $d_i(i > 1)$ share less or even no overlap with the question q, we need a retriever-updater-retriever framework to obtain P_D. The framework consists of two parts: single retriever and question updater.

Single retriever. For the multi-hop question problem, our task is to retrieve a document path P_d which needs an iterative retriever work flow. We take each iteration of retriever as a hop, and aim to fetch the most related document at one hop. Single retriever targets on retrieving the most related documents d for the input query q from the open domain corpus D at one hop. Specially, we use the structured triple facts T_d instead of the original unstructured text d to do the retriever matching. We aim to find the most related document d by calculating the distance score s between T_d with q. We choose the top-k documents by the distance score s .

Question updater. After retrieving the document of one hop using single retriever, the question updater aims to update the question with useful information within the retrieved document. Specifically, we aim to find a clue triple fact t', *i.e.*, *updater-clue* from the triple fact set T_d of last-hop retrieved document to generate a new question q' with the question q. With the generated new question q', we run the single retriever again to fetch the most related documents for the next hop retrieval.

III. The Framework of a Triple-Fact Based Retrieval Model

For a multi-hop question q, we target retrieving a reasoning document path $P_D = \{d_1, d_2, ..., d_n\}$ which needs an iterative retriever process. In our proposed triple-fact-based retrieval

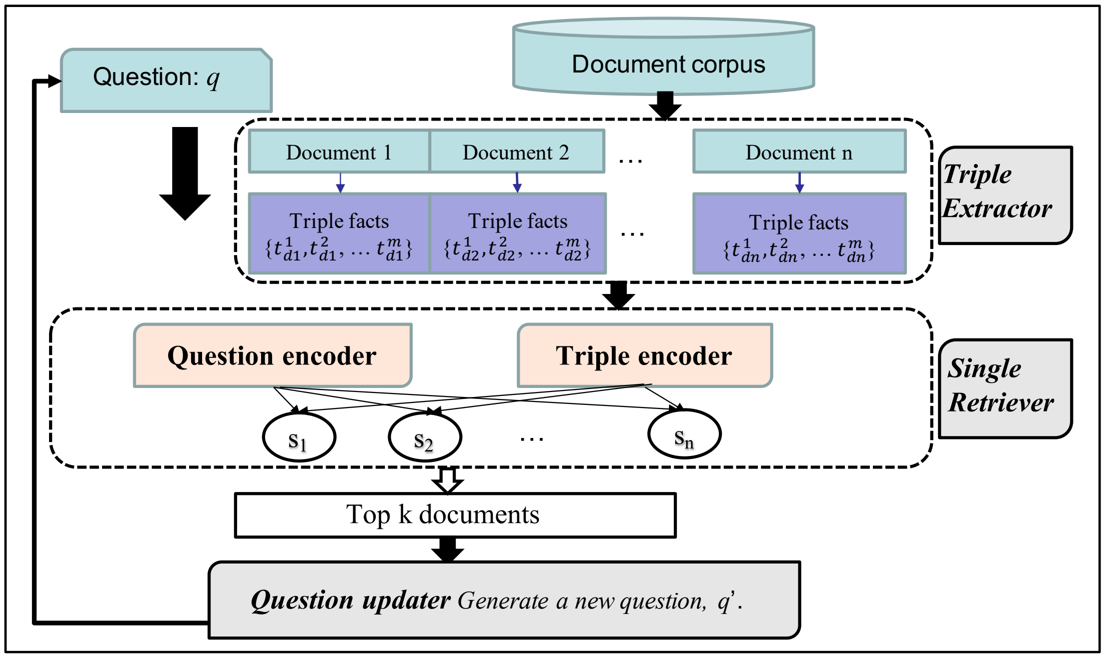

Question: q
Document corpus
Document 1
Document 2
Document n
Triple facts $\{t_{d1}^1, t_{d1}^2, \dots t_{d1}^m\}$
Triple facts $\{t_{d2}^1, t_{d2}^2, \dots t_{d2}^m\}$
Triple facts $\{t_{dn}^1, t_{dn}^2, \dots t_{dn}^m\}$
Triple Extractor
Question encoder
Triple encoder
s_1
s_2
s_n
Single Retriever
Top k documents
Question updater Generate a new question, q'.

Fig. 2: The framework of our triple-fact-based retrieval model. Given a multi-hop question q, we aim to fetch a document path P_d from a large-scale open-domain corpus. At the very beginning, we extract a triple fact set for each document as the structure representation. The retrieval work is an iterative semantic matching task indeed, *i.e.*, find the related document to the question in multiple times. We do the matching problem on the field of the extracted triple facts instead of the unstructured text for a source document.

framework, we do an explainable document retriever work on the field of triple facts instead of the original plain text of each document. We briefly introduce our triple-fact-based retrieval framework here and give a detailed explanation in the rest section. Our triple-fact based retrieval framework contains three components as shown in Fig 2: a triple extractor, a single retriever and a question updater.

First, for each document d, a triple extractor extracts a triple fact set for a document, which covers all the information that the document conveys. Then we conduct a single retriever by using the triple facts instead of the original unstructured text to do one-hop retrieval. Specifically, we encode the question q and the triple facts of documents into a semantic embedding vector. We calculate the question-document relevance score s_i in the vector space. For one-hop retrieval, we fetch k candidate documents based the relevance score s_i. As one-hop retrieved documents cannot satisfy the complete needs to answer a multi-hop question, we need a reasoning multi-hop retrieval. That is, before conducting a new hop $i+1$ retrieval ($i > 0$), a question updater is applied to update the question knowledge q with the retrieved document d_i. Then iteratively do the single retriever task with the new question q' until the document path is retrieved.

A. Triple Extractor

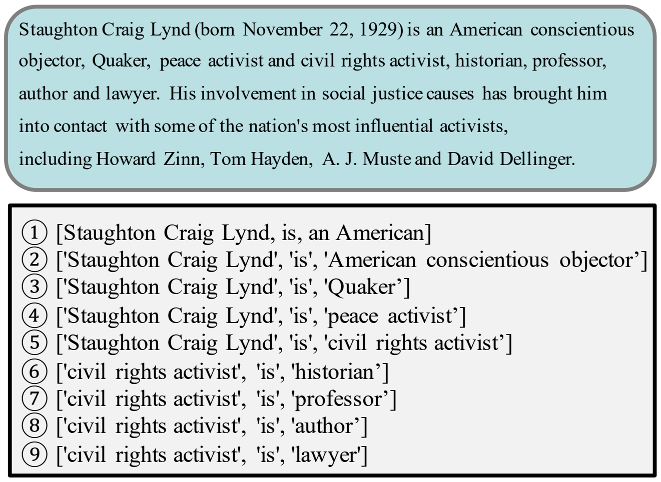
Essentially, the retriever task of question answer is to understand the semantic meaning behind the question and document. Structured factual expression is at the core of the natural language understanding task scenarios [18]. We propose a novel method to reduce the burden of document semantic understanding need by inducing prior information for the document, *i.e.*, the triple facts. The structured triple fact is much more readable for machine than the unstructured text.

Given an input document with unstructured text d, Triple Extractor aims to fetch a minimum size of triple facts set T_d which contains the complete factual information for the document. The triple facts set needs to cover the facts described in the document, as if some facts are missed in the set, the retriever performance will be hurt. The size of a set should be small so that the efficiency of retriever can have a guarantee.

Triple Fact generation. To obtain a complete set of triple facts for an input document d, we do coreference resolution over the document text and extract a union triple fact set $T_o = \{t_{d1}, t_{d2}, t_{d3}, ..., t_{dm}\}$, consisting of triples with format of $\langle s, p, o \rangle$. We use state-of-the-art Open information extraction (OIE) tools [21], [22] to extract relations and their arguments in an unsupervised manner efficiently as they do not need a specific-domain training dataset.

Triple Fact construction. As the union triple set maintains the informative facts in the document, there exists noise and redundancy over the extracted triple factst_{di} of the set. Many works which target solving the selection issue of triple facts [23]–[25], always use automated methods to disambiguate triple facts, *i.e.*, link the phrases within a triple to an existing Knowledge Base (KB), and calculate the diversity and similarity based on the link result. The selection performance [26] is limited by the error of the link process and the incomplete information capacity of the KB.

To construct a non-redundant triple fact set, a clustering



Staughton Craig Lynd (born November 22, 1929) is an American conscientious objector, Quaker, peace activist and civil rights activist, historian, professor, author and lawyer. His involvement in social justice causes has brought him into contact with some of the nation's most influential activists, including Howard Zinn, Tom Hayden, A. J. Muste and David Dellinger.

① [Staughton Craig Lynd, is, an American]
② ['Staughton Craig Lynd', 'is', 'American conscientious objector']
③ ['Staughton Craig Lynd', 'is', 'Quaker']
④ ['Staughton Craig Lynd', 'is', 'peace activist']
⑤ ['Staughton Craig Lynd', 'is', 'civil rights activist']
⑥ ['civil rights activist', 'is', 'historian']
⑦ ['civil rights activist', 'is', 'professor']
⑧ ['civil rights activist', 'is', 'author']
⑨ ['civil rights activist', 'is', 'lawyer']

Fig. 3: An example displays a document from Wikipedia and our extracted union triple fact set result. The triple facts are extracted by [21], [22] and contain noisy triple facts and redundant triple facts.

algorithm is always applied on the extracted triple facts [18], [19]. The existing clustering algorithm which can be applied on the triple facts, *i.e.*, HAC algorithm, needs a high time complexity $O(m^3)$. To effectively and efficiently solve the noise and redundancy problem in the union triple fact set, we propose our partition-based triple construction algorithm with pruning strategy. It aims to construct a complete-minimized triple fact set T_d for each document d. We first give the definitions for the type of triple fact in the scenario of question answer document retriever.

- *Noise triple fact*: Some extracted triple facts of the T_o have no distinct expression ability to the extracted source document as they can be extracted from many document sources or contain no specific semantic explicitly. This kind of triples can have a negative effect for the document retriever for question answer. For example, triple facts 6-9 in the Fig 3 can be recognized as noisy triple facts as the semantic meaning conveyed by these triple facts are conditionally right in the scene of this document. Moreover, we find the required information of the related document for the question is always concerned with an entity. Thus, we define the relatedness of a triple to its document as follows.

$$R(t,d) = \frac{|E_t \cap E_d|}{|E_d|} \tag{1}$$

E_t denotes the linked entities in the triple fact t, and E_d denotes all the linked entities in the document d.
- *Redundant triple*: There exists many redundancy over the triples and the size of the triple fact set has a severe effect on the document retriever efficiency. We define two types of redundancy triples. `mother-child`: If the information displays by a triple t_i is covered by another triple t_j, we call this pair of triples as a "mother-child" pair, we will use the mother triple t_j to replace the child triple t_j. For example, in Fig 3, triple fact 1 t_1 and triple fact 2 t_2 is a "mother-child" pair as the semantic meaning of triple fact 1 can be covered by triple fact 2, $s(t_1) \subset s(t_2)$.
`sibling`: if two triples (t_i, t_j) share a high similarity on their structure and semantic meaning, we call these two triples as a sibling pair. For this sibling triple pair, we generate a new "fusion" triple, t_{ij}. Specifically, we derive the sibling pair (t_i, t_j) with respect to specific structure and semantic meaning from the union triple fact set, we will generate a "fusion" triple by fusing the object (predicate-object) information. For example, the triple facts 2 and 3 in Fig 3 are recognized as a sibling triple fact pair, we will generate a "fusion" triple fact denoted as "[Staughton Craig Lynd, is, American conscientious objector, Quaker]".

To handle the two problems over the extracted union triple set, we propose our clustering-based pruning algorithm for triple set construction as shown in Algorithm 1.

Algorithm 1 Triple Fact Set Construction

```
Input: An input document $d$, a threshold size $l$, the linked entities in document $d$, $E_d$.
Output: A complete-minimized fact set $T_d, |T_d| = l$.
1: Generate an union triple fact set $T_o$ for $d$;
2: Calculate the relatedness score $s_{ti}$ for each triple facts $t_i \in T_o$;
3: Generate a pruning subset triple fact set $T'$ according to $s_{ti}$;
4: Partition the triple facts of $T'$ into small canopies $C_0 = C_i$;
5: #do inner clustering and merge over each canopy in parallel way.
6: while $|T_d| \leq l$ do
7:     for each $C_i \in C_O$ do
8:         Detect child triple facts $T_c$ of mother-child pairs in $C_i$, $C'_i \leftarrow C_i - T_c$ ;
9:         Detect sibling triple fact pairs $T_s$ and generate "fusion" triple facts $T_f$. $C'_i \leftarrow C'_i - T_s + T_f$
10:    end for
11:    $T_d = Union(C_i)$
12: end while
```

Algorithm details: We first prune the noise triple fact according to Eq.1 in line $2-3$. We partition the triple facts into smaller canopies instead of directly doing clustering. A canopy of triple facts consists of all the triple fact sharing the same "subject-predicate" or "subject" structure. The intuition of "subject-predicate" canopy is all the triple facts indeed describe a fact from different aspects. For example, triple fact 1-5 in Fig 3 which will be put in a canopy as they convey different role information for the person "Staughton Craig Lynd". The intuition of "subject" canopy is all the triple facts indeed describe different things about the same entity [18]. Then we conduct clustering in each canopy in a parallel way.

For line 8 in the Algorithm 1, it aims to generate a set C'_i without cover-redundancy. It is a set cover problem that has been proved to be NP-hard [27]. The problem is stated as follows:

Given a triple fact set $C_i = \{t_1, t_2, t_3, ..., t_n\}$, we aim to derive a subset $C'_i \subset C_i$, in which any pair of triples fact does not maintain a mother-child relation. $\forall t_i, t_j \in C'_i$, $s(t_i) \not\subset s(t_j)$ and $s(t_j)) \not\subset s(t_i)$.

To solve this problem, we propose a greedy-based algorithm to construct the non-redundant triple fact set $\hat{C}_i$. We first build a coverage set $\hat{C}_i$. As there exists three mother-child pairs in an input triple fact set C_i, $(t_1, t_2), (t_1, t_3), (t_4, t_1)$ [3], the coverage set is displayed as $\hat{C}_i$= $\{\{t_1, t_2, t_3\}, \{t_2\}, \{t_3\}, \{t_1, t_2, t_3, t_4\}, ..., \{t_n\}\}$. Then we use a greedy algorithm over $\hat{C}_i$ to generate C'_i. Specifically, each time we choose the maximum size item from the coverage set $\hat{C}_i$ which has no overlap with C'_i to populate C'_i until C'_i covers all triple facts in C_i. In the above example, we first select the maximum item $\{t_1, t_2, t_3, t_4\}$, we add the mapped triple t_4 into C'_i and then repeat to search the maximum item from $\hat{C}_i$ (which already deduct the chosen items). The time complexity of this step is $O(n)$.

For line 9 in Algorithm 1, the generation for C'_i without overlap-redundancy, this step aims to reduce the size of the final triple fact set without information loss. We do an iteration traverse over the triple fact set and directly obtain the sibling pairs whose similarity score is larger than a threshold α, using the generated fusion triple fact to replace the two triple facts of a sibling pair. This step takes $O(n^2)$ time. So the total time cost of Algorithm 1 is $O(n^2)$.

B. Single Retriever

To answer a multi-hop question q, it is acquired to retrieve documents which contain clues from a large-scale corpus D. Clues for a question refers to the reasoning information about the answer or the exact answer. Take Fig 1 as an example, document 2 contains the exact answer, the foundation time for the football club and document 1 contains the name of the football club which is latent necessary information. In this way, document retrieval task is a semantic matching problem on finding the clues. Specifically, the challenge of semantic matching problem is to project the natural question and document into high-quality feature spaces and use the distance between the two feature representations as the document-question relevance score.

To solve this matching problem, conventional word-based techniques [1], *e.g*, TF-IDF, BM25 algorithm, are proposed to use the overlapped syntactic information between document and question for retriever. These works fail to understand the semantic meaning behind the question and document. The recent PLM-based encoder models [3], [17] encode a document and a question to embedding vectors for capturing the semantic information. But the encoded embedding vectors capture basis and contain information loss as they compress the whole text information for long text document, into a single vector. Moreover, these works lack explainable facts for the exact match clues when retrieving a specific document. The document is retrieved as it is near to the question in the feature space. The exact matching information to the question is uncertain to these retrieval model.

To overcome these weakness of the exiting methods, we propose our explainable single retriever model. Given an input

[3]A mother-child pair (t_i, t_j) refers $s(t_i) \subset s(t_j)$.

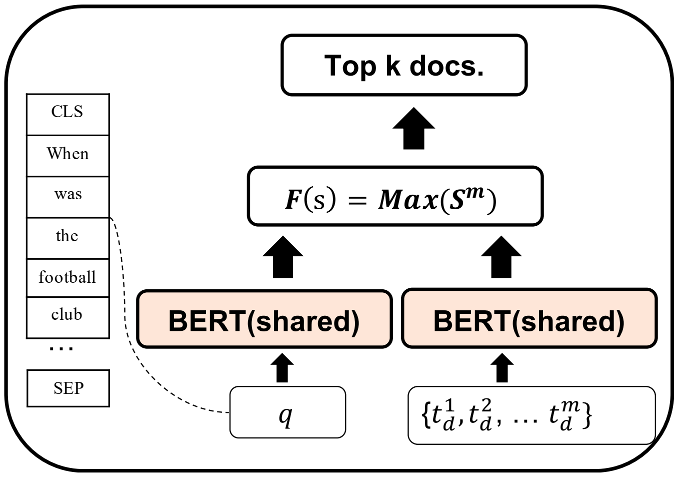


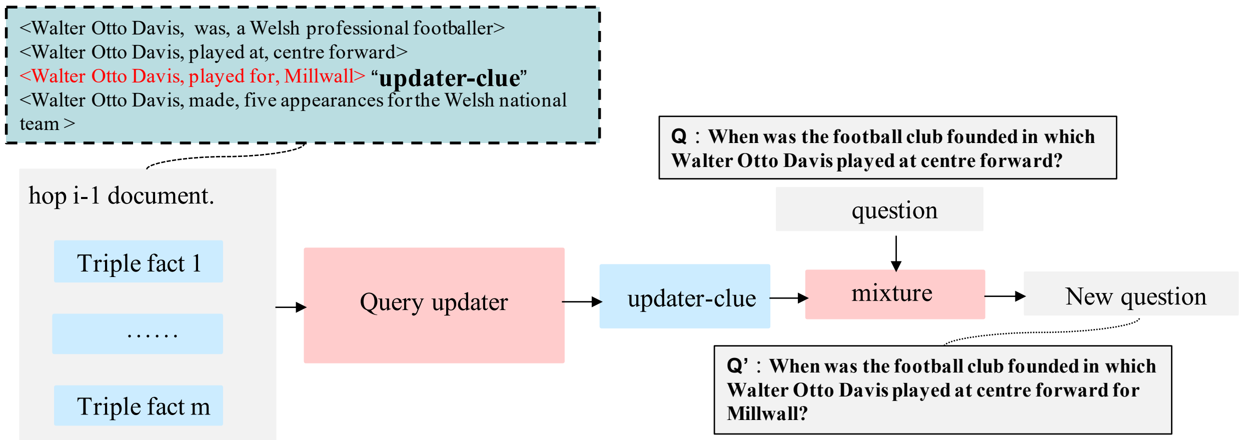
Fig. 4: The framework of single retriever. We use a parameter-shared encoder model to encode the question and document triple facts to semantic embedding representations. We choose the top k document based on the similarity score between v_q and E_{vd} by Equ 1.

question q, we deploy a retrieval over the extracted triple facts instead of the unstructured text of each document d. To calculate the relevance score s between the question q and a document triple fact set T_d, we propose to a PLM-based maximum matching strategy. We quantify the match information of the document to the question by some triple facts. In this way, our target is to find the clue triple facts in a document to the question. Represent each document d by its corresponding m triple facts, *i.e.*, $T_{di} = \{t_i^1, t_i^2, ..., t_i^m\}$, the number of matched triple facts to a question q is unknown and also no existing dataset can be used. From our exploration on the retriever task for QA problem, we propose our "*One Fact*" hypothesis.

One Fact matching. There exists one most related triple fact in the related document to a specific question.

Take Fig 1 as an example, the blue triple fact refers to the matched "fact". This triple fact contains the exact related information to the question q. The other triple facts is unrelated for the retrieval to the question q. Based on this hypothesis, we propose our PLM-based maximum-matching retriever model.

Text Encoder. We use a pre-trained language model, *i.e.*, Bert [15] with excellent semantic understanding ability to encode the question and document triple facts. For an input question q, we tokenize the question text into a sequence of tokens $\{w_1, w_2, w_3, ..., w_q\}$ and add specific labels "[CLS]" and "[SEP]" to the token sequence. The tokenized sequence of the question in Fig 1 is shown is the left in the Fig 4. We fine-tune the pre-trained language model to encode the question, we take the final hidden state for the special [CLS] label as the representation for the input sentence, $v_q \in R^d$. For a triple fact $t_i^j \in T_{di} =< s, p, o >$, we flatten the triple fact to a sentence-level representation. In a similar encode way, we obtain the encode embedding representations $E_d \in R^{m \times d}$ for the document's triple facts.



<Walter Otto Davis, was, a Welsh professional footballer>
<Walter Otto Davis, played at, centre forward>
<Walter Otto Davis, played for, Millwall> "updater-clue"
<Walter Otto Davis, made, five appearances for the Welsh national team >
hop i-1 document.
Triple fact 1
……
Triple fact m
Query updater
updater-clue
question
mixture
New question
Q : When was the football club founded in which Walter Otto Davis played at centre forward?
Q' : When was the football club founded in which Walter Otto Davis played at centre forward for Millwall?

Fig. 5: The question updater framework. We select one piece of triple fact from the hop $i-1$ document to augment the question information Q. We aim to generate a new question Q' which contains unseen clues from question to the next hop retrieval.

Score Calculation Strategy. Based on "one fact" hypothesis, our score calculation strategy is to choose the maximum score from the triple fact set. The calculation metrics for a question-document is defined in Equation 2 as the maximum distance score of the triple fact to the question. The distance score is defined as the inner product of dense vector representations of the question q and triple fact in Equation 4.

$$F(v_q, E_d) = \max(S^m), \tag{2}$$

$$S^m = \{s_1, s_2, ..., s_m\}, \tag{3}$$

$$s_i = cos(v_q, v_{di}), v_{di} \in E_d. \tag{4}$$

Training objective. We aim to fetch top k documents based on the relevance score calculated from $F(v_q, E_d)$. More specifically, we train a binary cross-entropy loss to maximize the relevance score of the ground document in our single retriever task as the following. $\hat{D}$ refers to the negative documents to a question q.

$$L = -\log F(v_q, E_d) - \Sigma_{\hat{d}\in\hat{D}} \log(1 - F(v_q, E_{\hat{d}}) \tag{5}$$

C. Question Updater

Another critical challenge in document retriever for multi-hop question is the need of a dynamically question updater during the multi-hop retriever procedure. Take the multi-hop question q in Fig 1 as an illustration example, after one-hop of single retrieval, we fetch the hop 1 ground document, *i.e.*, document 1. We need to use some knowledge of the retrieved document to update the question before conduct next hop retriever task. We need augment the "updater-clue", *i.e.*, the red triple fact from the triple fact set of document 1 to update the question q, generate a new question q' as shown in Fig 1.

The intuition behind the question updater is that the required hop i ($i > 1$) document d_i shares little or no overlap with the original question q, there exists clue information, *i.e.*, denoted as *updater-clue*, on the last hop retrieved document d_{i-1}. The updater-clue is necessary to next hop document retrieval. For the question q in Fig 1, we can get the exact football club name information from document 1 after hop 1 retrieval. Without knowing this new information of the football club name, we cannot do the correct next hop document retriever which is to obtain the foundation year of the football club.

However, the existing works conduct the question updater by concatenating the whole text of retrieved document to the question [17] or using a recurrent neural network on the embedding vectors [3], [5]. Obviously, these full-text updater methods induce large noise information to the question. For example, if we use the new question which is a concatenation of question q and document 1 to do next hop search in Fig 1, the retrieval result is dominated by the information of document 1 and unrelated to the desired need of the question.

Others propose to generate some clue spans from the retrieved document in their query generator model [13], [14]. Specifically, they take some overlapped tokens of the retrieved document and the question q as an input to train a reader model for generate a new question q'. However, this type of work is limited on the efficiency by the exponential candidates from the retrieved document. For a document $d = \{w_1, w_2, ..., w_a\}$ (w_i refers to the token in the document), there exists $O(2^a)$ candidates of the clue spans which made the updater computation very time consuming.

To effectively and efficiently solve the question updater problem, we conduct a triple-fact-based updater model. Specifically, as shown in Fig 5, we aim to select a triple fact within the retrieved document to generate a new question with the original question. The query updater module is to score the selected triple fact. We concatenate the question q and a triple fact t_i from the hop $i-1$ retrieved document into a sequence $L = \{w_{q1}, w_{q2}, ..., w_{t1}, w_{t2}, ...\}$, we encode the sequence into an representation embedding and calculate the similarity score of L and q' on the training process. q' is the ground question for next retrieval. We select the triple fact t' with maximum score as the updater-clue, and we add the knowledge of updater-clue into the original question to generate a new question q' in a de-duplication way.

This updater way is simple but much more effective and efficient. The computation cost for selecting a candidate clue is equal to the size of the triple fact set $O(|T_d|)$, which is far less than selecting one from $O(2^a)$ combinations of tokens in the document.

IV. Experiments

In this section, we demonstrate that our proposed model, "Triple-fact based retrieval model", aiming to solve the problem of document retriever for multi-hop QA problem in an open-domain scenario, can be effectively applied to the structured representation of document and well learn the semantic matching relationship between questions and documents. We first describe the dataset and evaluation metrics used in the experiments. Specifically, we introduce the complete setting in our model. We compare our experiment result with several baseline methods from two aspects. **1.** We validate whether the structured representation is more readable to machine, and the semantic matching problem can be better solved when applied to the structured representation than the unstructured representation. **2.** We evaluate our triple-fact-based retriever-updater solution, which working on the field of the structured representation can outperform other state-of-the-art document retrieval works for multi-hop QA problem.

A. Dataset and Evaluation metrics

	Bridge	Comparison	Total
Train	72991	17456	90447
Test	5918	1487	7405

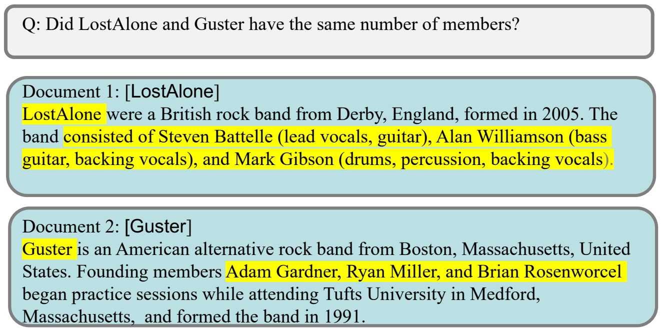
TABLE I: The statistics of HotpotQA dataset.

HotpotQA [28] is a human-annotated large-scale multi-hop QA dataset. There are 90447 questions for training and 7405 questions for testing. We conduct our experiments on the *full wiki* setting for the open-domain scenario. Under the *full wiki* setting, the task is to retrieve a document path from the entire Wikipedia corpus, *i.e*, 5M Wikipedia documents, for a question. The length of the ground document path is 2 for both training and testing questions. There are two types of questions in the dataset defined by the reasoning types, **Bridge** and **Comparison**. Bridge question refers to the retrieval of the ground document path as a chain reasoning process. We need to identify bridge information before the second hop can be answered by filling in the bridge, the question in Fig 1 is a bridge question. For comparison question, the retrieval of the ground document path is a simultaneous reasoning process. The retrieved document satisfies one of the two properties for the question, *e.g.*, the question in Fig 6. We take an intersection of the documents to generate a document path.

Wikihop [29] is a cross-document multi-step Reading comprehension dataset. There are 43738 questions in the train dataset and questions in the validation dataset. Specifically, the original dataset doesn't provide the grounded document supervision, we process the dataset to satisfy our retriever task setting[4].

Metrics. To evaluate the document-level retrieval accuracy for the multi-hop question, we use the following metrics. To evaluate the result for one hop retrieval, Paragraph Recall (PR) is used to evaluate if one of the ground-truth document d_g is included in the retrieved documents D_r. $\forall q \in Q_{pr}, \exists d_g \in D_r, PR = \frac{|Q_{pr}|}{|Q|}$ .To evaluate the result of our retrieval model

[4]Please refer to our full version: URL.


Q: Did LostAlone and Guster have the same number of members?

Document 1: [LostAlone]
LostAlone were a British rock band from Derby, England, formed in 2005. The band consisted of Steven Battelle (lead vocals, guitar), Alan Williamson (bass guitar, backing vocals), and Mark Gibson (drums, percussion, backing vocals).

Document 2: [Guster]
Guster is an American alternative rock band from Boston, Massachusetts, United States. Founding members Adam Gardner, Ryan Miller, and Brian Rosenworcel began practice sessions while attending Tufts University in Medford, Massachusetts, and formed the band in 1991.

Fig. 6: A comparison question and its document path which is composed by document 1 and document 2. Each document contains one piece of information in yellow, denoted as a property, for answering the question.

for a document path, Paragraph Exact Match (PEM) is used to evaluate if all of the ground-truth documents D_g are included among the retrieved documents D_r. $PEM = \frac{|Q_{pem}|}{|Q|}, \forall q \in Q_{pem}, D_g \subset D_r$.

B. Settings and Implementation Details

Triple Extractor. There exists a trade-off between the size of triple facts l and the length of a triple fact when representing the original document knowledge. We set the maximum length of a triple fact as 256 and the threshold size l in Alg 1 as 40 from ablation study. We index the 5M Wikipedia documents provided by [28] and the corresponding triple fact sets with Elasticsearch 7.13 [30].

We deploy the two state-of-the-art sentence-level OIE tools, StanfordIE [22] and MinIE [21] to extract the triple fact set T_d^s, T_d^m for each document. Specifically, we first conduct coreference resolution [5] over the document and then use the NLTK tool [6] to process the document d to a normalized sentence set $S_d = \{s_1, s_2, ..., s_j\}$. The sentence-level triple extractors [21], [22] are then used to extract triple facts for each sentence $s_i \in S_d$. We generate a union triple fact set $T_o = \{T_d^s \cup T_d^m\}$ from the two extracted results. A complete-minimized fact set T_d will be constructed by Alg 1 over T_o.

Retrieval model. We choose a shared-parameter BERT [15] as the pre-trained language model for the single retriever and updater to encode the triple facts of a document and the question. We keep the same hyper-parameters to [1], [3] on model training. The experiments for the above two learning models are running on a machine with 8 32GB V100 GPUs and all the codes are implemented in python.

Training data construction for single retriever. As for the training data of single retriever, we choose a ground document with the highest score from the document path by BM25 on the field of our triple fact set. For the negative document construction, we index from the whole Wikipedia corpus and choose the top 9 documents except the ground documents. Each question is trained on a 10-size set of 1 positive document and 9 negative documents.

[5]https://github.com/huggingface/neuralcoref
[6]https://www.nltk.org/

C. Effectiveness of Triple Extractor

To evaluate the effect of structured representation, *i.e.*, a triple fact set T_d, on the semantic matching problem between documents and questions, we conduct experiments on the different match fields of a document for the input natural language question.

1) Effectiveness for structured representation: For each question q, we apply a non-learning token-based algorithm, *i.e.*, BM25 [30] for semantic similarity calculation on the HotpotQA dataset. We retrieve top $k = 10$ documents from the Wikipedia corpus for each question. We report the experimental results in the Table II. For hop $i = 1$, we show the PR score as the number of expected ground document is 1. For hop $i = 2$, we report the PEM score as that both of the two ground documents are needed to be retrieved.

- Text matching. We do the document retrieval on the field of text. We calculate semantic similarity between the input question q and the entire document plain text d.
- TFS matching. We do the document retrieval on the field of triple fact set. We calculate the semantic similarity between the input question q and the triple fact set T_d which is generated by Alg 1.

From Table II, we can find that the retrieval result of TFS matching gains an improved performance over the Text matching. For hop $i = 1$ retrieval, TFS matching outperforms Text matching by 2.6% on the PEM metrics for the total train question and 0.4% for the total test question. For hop $i = 2$ retrieval, TFS matching outperforms Text matching by 5.3%. The experiment result shows that the structured representation can maintain the same knowledge for the source document to text representation and contain more factual information. There are fewer noise tokens, *i.e.*, words containing no distinct and semantic meaning, in the triple fact set of a document, and the matching can be less affected by this noise information.

Especially, TFS matching outperforms Text matching by 22.2% on the PEM metrics for the Comparison question. For the comparison question, the second document always shares less overlap with the question. A tiny number of words always describe the related information to the question over a long sequence of tokens of the document. By the structured representation, we improve the information expression capability to the source document and promote the retrieval that learns the matching between question and document.

2) Triple extraction performance: Furthermore, we validate the extraction performance of our Triple fact set construction algorithm in Table III. For the open information extraction, it is hard to validate the quality of the extraction except for benchmark datasets [31]. At the same time, our work can be a validation to verify the effectiveness of extraction by using the extracted structure to do the semantic matching. In this way, document retrieval can be seen as a practical application to validate the triple extraction problem [22]. We conduct the document retriever on the field of different triple fact sets.

- StanfordIE-TFS. We do the document retrieval on the field of T_d^s.
- MinIE-TFS. We do the document retrieval on the field of T_d^m.

From table III, we can find the constructed triple fact set T_d performs well on maintaining the source document information than the two baseline works. Besides, the document retrieval performance of MinIE-TFS is better than StanfordIE-TFS. MinIE-TFS outperforms StanfordIE-TFS by 4.3%, 2.3%, 5.2% on the train hop-1, train hop-2, test hop-1 for the bridge question respectively. For the comparison question, MinIE-TFS outperforms StanfordIE-TFS by 4.9%, 1% on the train hop-1, test hop-1. From the analysis of the extraction data by human inspection, we find MinIE shows a better extraction ability for the long sentence. And the Wikipedia document is always composed of long sentences. Besides, MinIE minimizes their extracted triple fact set instead of the actual extraction. However, both extractors only consider the inner relationship between the triple facts and do not prune their extractions for a good semantic representation.

D. Effectiveness of Single Retriever

In this section, we evaluate the retrieval performance of single retriever. The conventional token-based works always be outperformed by the learning-based model [9] and the document retrieval result can be observed from Table IV . We compare our single retriever with two learning-based models.

- GoldEn [13] is an IR-learning combined work. GoldEn trains a query generator for each hop retrieval, and use the generated question to retrieve document over the whole corpus.
- TPRR [7] is a typical PLM-based model. TPRR encodes the complete document plain text and question to dense representations in a vector space and projects the vector to a scalar score.

Instead of retrieving on the field of document text, our single retriever model learns the semantic matching for question and document in the triple fact level. Unlike existing works [3], [7], [32], which encode the complete document text to a dense representation, we encode the triple facts of each document and calculate the distance between the encoded triple fact set to the question. In this way, we quantify the implicit semantic mapping information to a piece of explicit knowledge, *i.e.*, some triple facts $T_m = \{t_1, t_2, ..., t_m\}$.

There exist three construction strategies for T_m as follows, and we show the retrieval performance of different strategies in Table V.

- "one fact": We quantify the mapping knowledge to one triple fact, $T_m = \hat{t}$. In this way, we choose the most similar triple fact $\hat{t}$ to the question q, and use the cos similarity between the embedding representations, $\hat{\boldsymbol{t}}$ and $\boldsymbol{q}$, as the question-document relevance score.
- "top$k$": We quantify the mapping knowledge to multiple triple facts, $T_k = \{t_1, ..., t_k\}$ for the assumption that the clues may lies on multiple triple facts. We use the mean

		Text matching			TFS matching		
		Bri	Com	total	Bri	Com	Total
Train	hop i = 1	88.5%	78.8%	86.6%	88.4%	92.8%	89.2% (+**2.6**%)
	hop i = 2	34.4%	30.7%	33.7%	35.6%	52.9%	38.9 (+**5.3**%)
Test	hop i = 1	86.1%	76.1%	84.1%	86.3%	77.4%	84.5% (+**0.4**%)

TABLE II: Non-learning based document retriever performance on the Hotpot QA dataset.

		TFS matching		MinIE-TFS		StanfordIE-TFS	
		Bri	Com	Bri	Com	Bri	Com
Train	hop i = 1	88.4%	92.8%	87.4%	81.7%	83.1%	76.8%
	hop i = 2	35.6%	52.9%	32.4%	32.8%	30.1%	53.7%
Test	hop i = 1	86.3%	77.4%	85.4%	77.8%	80.2%	76.8%

TABLE III: Document retrieval result on the field of different triple fact extraction methods.

Model	Bridge	Comparison	Total
TPR	93.43%	85.14%	91.77%
GoldEn	90.23%	87.75%	89.76%
Triple-Retriever-top2	90.32%	85.35%	89.28%
Triple-Retriever-top5	87.49%	84.13%	86.84%
Triple-Retriever-mean	91.13%	84.81%	89.89%
Triple-Retriever	**94.15**%	**87.21**%	**92.73**%

TABLE IV: One hop retrieval Result: The PR result of Top-8 documents for one hop retrieval on Hotpot QA dataset.

score of top-k triple facts to the question as the question-document relevance score as follows.

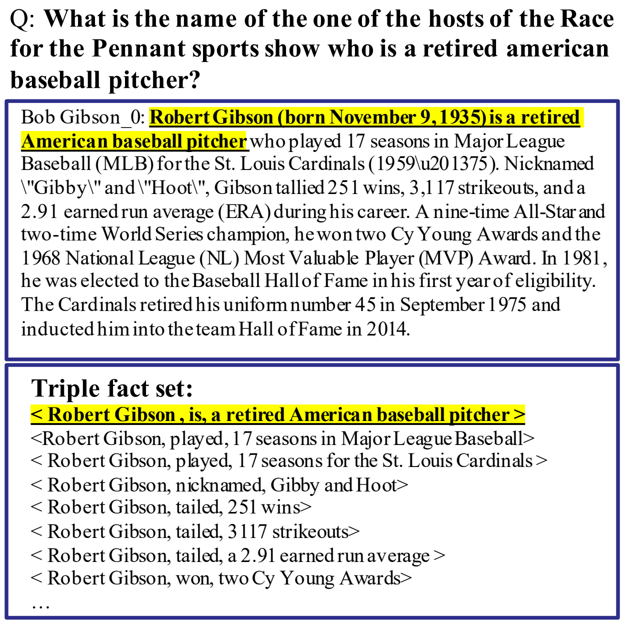
$$F(v_q, E_d) = \frac{1}{k} \sum_{t \in T_k} s(q, t) \tag{6}$$

- "mean": This strategy is equal to the existing full-text compression works in knowledge representation. As the existing works encode the full context in a document to an embedding vector, we simulate this compression strategy in our triple retriever set selection. We use the mean score of all the triple facts in a document as the question-document relevance score as follows.

$$F(v_q, E_d) = \frac{1}{m} \sum_{t \in T_m} s(q, t) \tag{7}$$

We calculate PR metrics of top-8 selected documents generated by each models. From Table IV, Triple-Retriever with "one fact" strategy achieves the best performance on one hop document retrieval. It outperforms GoldEn by 3.92%, 2.97% for the bridge questions and total questions. Specifically, Triple-Retriever applies a pre-trained language model (Bert) to do encoder while GoldEn uses the classical IR method to calculate the semantic relevance score. GoldEn indeed obtains semantic matching information through their query generator model and injects that information into the question. The problem is that the injected semantic information is not adequate to capture the complete relevance of question-document from their heuristic rules.

Triple-Retriever outperforms TPR by 0.72%, 2.07%, 0.96% for the bridge questions, comparison questions and total questions respectively. TPR is a typical PLM-based model which encodes a document and a question to a dense representation. They score the special "CLS" embedding to measure the semantic relevance between the question and document. In this way, the semantic relevance metrics are affected by the other information of the document. Moreover, TPR encodes the question and document to a 768 dimension embedding to capture the semantic information, which inevitably the encoded representation maintains loss. Also, the model makes predication of the related document to a question in an unexplainable way. They fail to locate the exact information in the document matched to the question as their model is a BlackBox for the document retrieval. Triple-Retriever explicitly learns the semantic matching information in the training process. We show the calculation process by an example in Fig 7. The semantic matching information is displayed in yellow and bold. The left tokens in the document are unrelated to the question, denoted as noise information. The noise information which dominates in the full document text can navigate the measure of the TPR model. In Triple-Retriever, the relevance score is calculated on the single yellow-bold triple fact to avoid the noise information problem.


Q: **What is the name of the one of the hosts of the Race for the Pennant sports show who is a retired american baseball pitcher?**

Bob Gibson_0: **Robert Gibson (born November 9, 1935) is a retired American baseball pitcher** who played 17 seasons in Major League Baseball (MLB) for the St. Louis Cardinals (1959\u201375). Nicknamed \"Gibby\" and \"Hoot\", Gibson tallied 251 wins, 3,117 strikeouts, and a 2.91 earned run average (ERA) during his career. A nine-time All-Star and two-time World Series champion, he won two Cy Young Awards and the 1968 National League (NL) Most Valuable Player (MVP) Award. In 1981, he was elected to the Baseball Hall of Fame in his first year of eligibility. The Cardinals retired his uniform number 45 in September 1975 and inducted him into the team Hall of Fame in 2014.

Triple fact set:
< Robert Gibson , is, a retired American baseball pitcher >
<Robert Gibson, played, 17 seasons in Major League Baseball>
< Robert Gibson, played, 17 seasons for the St. Louis Cardinals >
< Robert Gibson, nicknamed, Gibby and Hoot>
< Robert Gibson, tailed, 251 wins>
< Robert Gibson, tailed, 3117 strikeouts>
< Robert Gibson, tailed, a 2.91 earned run average >
< Robert Gibson, won, two Cy Young Awards>
...

Fig. 7: A question and one of its ground document in HotpotQA dataset. The matching information of the document is shown in yellow and bold. For the field of document text, the matching information is a subsequence of tokens of the entire document with arbitrary location and length. For the filed of triple fact, the matching information is a concrete expression of a triple fact.

Another study is concerned with the triple fact selection strategy. For different strategies on construction of T_m in

Table IV, “one-fact” (Triple-Retriever) outperforms “top2” (Triple-Retriever-top2) by 3.83%, 1.89%, 3.45% for the bridge questions, comparison questions and total questions respectively. “one-fact” (Triple-Retriever) outperforms “top5” (Triple-Retriever-top5) by 6.66%, 3.08%, 5.89% for the bridge questions, comparison questions and total questions. The comparison result of ”top2” and ”top5” concludes that the performance is reduced with the increase of the number of triple facts. It proves that the semantic matching information is explicit covered by one single triple fact. The noise triple facts can navigate the retrieval to negative documents and damage the retrieval performance. We also conduct Triple-Retriever on T_o and the PR of top 8 documents is 89.1% (comparison) and 88.7% (bridge). It proves that our constructed triple fact set T_d is a complete factual representation while the actual extraction T_o contains discrete facts .

An interesting observation is the “mean” strategy is better than top$k$ strategy. As the “mean” strategy is a special format of top$k$ where $k = |T_d|$, the result should be dropped according to the above conclusion. From further experiment analysis, we find the reason is concerned about entity-centric influence. The entity-centric influence [2] refers that the semantic relevance always comes from the overlapped entity and the encoder model is fine-tuned to make the entity can be well learned on the encoded embedding in training. In the “mean” strategy, we find most triple facts $t \in T_d$ contain the title entity and the “mean” strategy benefits from the entity-centric influence.

E. retriever-updater model

For the multi-hop QA problem, the task is to retrieve a document path for each question. We derive the top 8 documents for one hop retrieval given the input question. Then the triple fact-based updater model is trained to generate the new question. For the question updater, we train our triple fact-based updater model on the dataset generated by GoldEn [13] query-generator module. As there are is dataset from the HotpotQA website for the ground updated question, GoldEn proposes to use heuristic rules, *i.e.*, select the longest common subsequences of the question and hop-i document, to generate a ground updated question of hop-(i+1). We trained our triple fact-based updater model to generate a new question $q^{'}$. Then, we conduct the single retriever model with the new question $q^{'}$ to obtain the document of next hop retrieval.

We compare the performance of our single-retriever-updater model on solving the document retriever with three start-of-the-art works. Specifically, all of these works utilize pre-trained language model to encode the input.

Model	Bridge	Comparison	Total
TPRR	93.43%	85.14%	91.77%
HopRetriever	90.01%	85.41%	89.09%
MDR	24.58%	89.71%	37.66%
PathRetriever	86.63%	90.38%	87.39%
Triple-fact Retrieval-base	**93.15%**	**86.91%**	**91.92%**
Triple-fact Retrieval	**94.19%**	**89.21%**	**93.22%**

TABLE V: The performance of document path retrieval for multi-hop question.

- PathRetriever [3] is a recurrent graph-based retrieval model to retrieve the document path from a graph structured document corpus. They build a Wikipedia graph by linking documents by the hyperlink relationship. They sequentially retrieve evidence documents from the Wikipedia graph to form a reasoning path by a recurrent neural network.
- MDR [17] employs dense retrieval to solve the multi-hop question with a simple recursive framework. They iteratively encode the question and hop i retrieved document as a query vector and retrieve hop $i+1$ document using efficient maximum inner-product search (MIPS) method.
- HopRetriever [2] leverages structured entity relation and unstructured introductory facts to do the reasoning retrieval for multi-hop question. They encode the textual context around each entity in the document text into mention embedding to represent the implicit structured knowledge.

We select top 8 document paths for each question. The score of a document path is the sum of each hop document as defined by Equ 8. The retrieval performance of this setting is reported as Triple-fact Retrieval-base in Table V.

$$s(p) = s(d_1) + s(d_2) \tag{8}$$

In this way, the retrieved documents are suboptimal and forward retrieved, while the existing works choose a global optimal document path. They supervise the hop i document selection by the hop $i+1$ retrieved document. In our retrieval model, the retrieval process is a sequential-iterative work of single retriever and question updater. To tackle this issue, we use a document path ranking model to select the document path, which scores all the combinations of path given the original question q. The ranking model is same to the single retriever while the only change is to use the document path as the document input. The retrieval performance with the ranking model is reported as Triple-fact Retrieval in Table V.

We calculate PEM metrics of top-8 selected document paths. From Table V, Triple-fact Retrieval achieves the best performance on document path retrieval for multi-hop question. Triple-fact Retrieval-base model has a better performance to TPRR by 1.77% PEM metrics for the comparison questions while has a poor performance for the bridge question. It is due to the global supervision setting in TPRR model and we tackle the suboptimal issue of the path score calculation metrics in Triple-fact Retrieval. Especially, a bridge question needs an important semantic connection for the two hop retrieval in HotpotQA that the hop 2 document is retrieved based on the information in hop 1 document.

Both of our models outperform PathRetriever on the document path retrieval. Triple-fact Retrieval outperforms PathRetriever by 7.56%, 5.83% for the bridge questions and total questions separately. Triple-fact Retrieval-base outperforms PathRetriever by 6.52%, 4.53% for the bridge questions and total questions respectively. The limitation of PathRetriever model comes from the graph structure of search space. The retrieved documents of PathRetriever satisfy a path structure in the Wikipedia graph, that is to say, the retrieved documents need maintain a hyperlink relationship. However, our triple-fact-based model does not show an excellent semantic matching ability as PathRetriever for the comparison question. By manual inspection on the failure comparison questions, we find the hop 2 document has an entity-centric overlap to the original question and no overlap to the hop 1 document actually. In this way, the question updater model induces unnecessary knowledge to the hop 2 retrieval. The RNN layer of PathRetriever can avoid this issue but need a high training time.

Both of our models outperform HopRetriever on the document path retrieval. Triple-fact Retrieval outperforms HopRetriever by 4.18%, 3.8%, 4.13% for the bridge questions, comparison questions and total questions separately. Triple-fact Retrieval-base outperforms HopRetriever by 3.14%, 1.5%, 2.83% for the bridge questions, comparison questions and total questions separately. In fact, HopRetriever tries to increase the weight of *entity* information in their embedding space for semantic matching. The entity information is only part of the semantic information for most multi-hop questions and sometimes, the semantic information is a sequence of non-entity tokens in the document.

V. Related work

Structured Text representation. Many existing works aim to generate a machine-readable representation in a text document as a set of facts. Some works [23], [24] utilize an existing knowledge source, *i.e*, a Knowledge base (KB) to help pruning their extracted results. MinIE [21] and StanfordIE [22] are the two main state-of-the-art triple extractor tools. However, the extracted results are always full of redundancy and noise. SALIE [18] proposes a clustering algorithm to cluster the extracted facts depending on page-rank-based relevance scores. But their algorithm only considers the relevance and diversity over the extracted facts and lacks of the influence reason of the source texts. Others [19], [33] score a triple fact based on the linking score to an existing KB. Our triple construction algorithm can alleviate the link error and the defined score considers the relationship between a triple fact and the source document.

Multi-hop Open-domain question answering. To retrieve the document path for a multi-hop question from an open-domain corpus, the classical IR algorithms, *e.g.*, *TF-IDF* or *BM25* have been applied to conduct the matching of question and document [1], [34] . However, these term-based works fail to capture the semantic meaning of the natural language [9] as they can only capture the lexical overlap relationship.

Some works [32], [35] on dense retrieval methods propose to utilize the deep understanding of pre-trained language model (PLM). They encode the question and documents into dense representations to complete the retrieval. It has proved that PLM-based models have significant retrieval improvements over the classical IR works. However, they are limited on simple question which a single document can satisfy the knowledge requirement of answering the question. [3], [7], [13], [14], [17] try to utilize the superior understanding ability of PLM to solve the multi-hop question. [17] encodes the document and question to fixed-dimension embedding vectors and calculate the inner-product score of the two vectors. Their retrieval process lacks of explainable and the embedding vector maintains information loss on capturing the long-text document. They fail to effectively update the question in the multi-hop retrieval as they directly concatenate the full text of retrieved document. [3] proposes to use a recurrent neural network to fetch the document path from the linked Wiki-document graph. Their retrieval result is limited by the hyperlink relationship between documents. If the required documents in a document path share no hyper-link, their model fails to find the exact path. Another similar work, [4] do the multi-hop retriever based on the linkage information, they holds the assumption, the hops document can be retrieved by the entity information. they rerank the document based on the entities in the document which this work has limitation search space on the hyperlink entity as they only choose the entity-described document on their reranking task. Similar shortage lies on [36] as their retrieval is limited by the entity linking.

HopRetriever [2] is a similar work to us as they also consider the other knowledge except the plain text for the retrieval. Specifically, they utilize the entity linking information in their document encoder to enhance the correlations between question and document. Their work captures the relation between entities by the hyperlinks in an explicit way. They are limited to do the retrieval as the document-question relationship is far more than the shared entity information.

MUPPET [5] works on the multi-hop question answer task in a sentence-encoder manner. They use recurrent neural layers to encode the question and document sentence into vector embeddings for document retriever. Furthermore, they induce an reformulation layer to update the question embedding within each hop retrieval. Besides, other works try to apply different neural models on the document retrieval problem. [37] use a GAT network to do the document retriever. That kind of works is more related to the study of deep learning community.

VI. Conclusion

This paper introduces a new triple-fact based retrieval model which retrieves a document path from the Wikipedia corpus to answer multi-hop open-domain questions. We are the first work to do the semantic matching between questions and documents using the structured representation. Compared with unstructured text, the structured knowledge of a document can be well learned by machine. To extract the structured representation of documents, we deploy the state-of-the-art OIE tools on the 5 million Wikipedia documents. To solve the redundancy and noise in the extracted results, we propose an efficient clustering-based algorithm to generate a complete-minimized triple fact set for each document. We verify the structured expression is much more readable for machine by the improved performance of question-document retrieval.

For a multi-hop question, our triple-fact based retrieval model retrieves documents by conducting the semantic matching on the field of our constructed triple fact set.The triple-fact-based model gains an improvement over the retrieval compared with the other models. Specifically, our retrieval process is interpretable from the triple-fact level as we can locate the matching information during the retrieval. Future work involves end-to-end training of our single retriever and updater for improving upon our current two-models training. Moreover, we plan to explore the graph structure of the triple facts for document retrieval.

References

[1] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.

[2] S. Li, X. Li, L. Shang, X. Jiang, Q. Liu, C. Sun, Z. Ji, and B. Liu, "Hopretriever: Retrieve hops over wikipedia to answer complex questions," *AAAI*, 2020.

[3] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, "Learning to retrieve reasoning paths over wikipedia graph for question answering," in *ICLR*, 2019.

[4] R. Das, A. Godbole, D. Kavarthapu, Z. Gong, A. Singhal, M. Yu, X. Guo, T. Gao, H. Zamani, M. Zaheer *et al.*, "Multi-step entity-centric information retrieval for multi-hop question answering," in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, 2019, pp. 113–118.

[5] Y. Feldman and R. El-Yaniv, "Multi-hop paragraph retrieval for open-domain question answering," in *ACL*, 2019, pp. 2296–2309.

[6] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *TKDE*, vol. 30, no. 5, pp. 824–837, 2017.

[7] X. Zhang, K. Zhan, E. Hu, C. Fu, L. Luo, H. Jiang, Y. Jia, F. Yu, Z. Dou, Z. Cao *et al.*, "Answer complex questions: Path ranker is all you need," *SIGIR*, 2021.

[8] F. Hill, A. Bordes, S. Chopra, and J. Weston, "The goldilocks principle: Reading children's books with explicit memory representations," 2015.

[9] Y. Bai, X. Li, G. Wang, C. Zhang, L. Shang, J. Xu, Z. Wang, F. Wang, and Q. Liu, "Sparterm: Learning term-based sparse representation for fast text retrieval," *arXiv preprint arXiv:2010.00768*, 2020.

[10] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[11] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in *SIGIR'94*. Springer, 1994, pp. 232–241.

[12] E. Riloff and W. Lehnert, "Information extraction as a basis for high-precision text classification," *ACM Transactions on Information Systems (TOIS)*, vol. 12, no. 3, pp. 296–333, 1994.

[13] P. Qi, X. Lin, L. Mehr, Z. Wang, and C. D. Manning, "Answering complex open-domain questions through iterative query generation," in *(EMNLP-IJCNLP)*, 2019, pp. 2590–2602.

[14] P. Qi, H. Lee, O. Sido, C. D. Manning *et al.*, "Retrieve, read, rerank, then iterate: Answering open-domain questions of varying reasoning steps from text," *arXiv preprint arXiv:2010.12527*, 2020.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[16] M. Tu, K. Huang, G. Wang, J. Huang, X. He, and B. Zhou, "Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents," in *AAAI*, vol. 34, no. 05, 2020, pp. 9073–9080.

[17] W. Xiong, X. Li, S. Iyer, J. Du, P. Lewis, W. Y. Wang, Y. Mehdad, S. Yih, S. Riedel, D. Kiela *et al.*, "Answering complex open-domain questions with multi-hop dense retrieval," in *ICLR*, 2020.

[18] M. Ponza, L. Del Corro, and G. Weikum, "Facts that matter," in *EMNLP*, 2018, pp. 1043–1048.

[19] X. Lin and L. Chen, "Canonicalization of open knowledge bases with side information from the source text," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 950–961.

[20] Y. Liu and M. Lapata, "Learning structured text representations," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 63–75, 2018.

[21] K. Gashteovski, R. Gemulla, and L. d. Corro, "Minie: minimizing facts in open information extraction." ACL, 2017.

[22] G. Angeli, M. J. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *ACL*, 2015, pp. 344–354.

[23] H. Xin, X. Lin, and L. Chen, "Casie: Canonicalize and informative selection of the openie system," in *ICDE*, 2021, pp. 2009–2014.

[24] A. Moro and R. Navigli, "Wisenet: Building a wikipedia-based semantic network with ontologized relations," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 1672–1676.

[25] L. Jiang, L. Chen, and Z. Chen, "Knowledge base enhancement via data facts and crowdsourcing," in *ICDE*. IEEE, 2018, pp. 1109–1119.

[26] X. L. Dong, "Challenges and innovations in building a product knowledge graph," in *SIGKDD*, 2018, pp. 2869–2869.

[27] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, 1971, pp. 151–158.

[28] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," in *EMNLP*, 2018, pp. 2369–2380.

[29] J. Welbl, P. Stenetorp, and S. Riedel, "Constructing datasets for multi-hop reading comprehension across documents," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 287–302, 2018.

[30] C. Gormley and Z. Tong, *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.", 2015.

[31] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh, "A survey on open information extraction," *arXiv preprint arXiv:1806.05599*, 2018.

[32] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[33] H. Bast, B. Buchhold, and E. Haussmann, "Overview of the triple scoring task at the wsdm cup 2017," *WSDM*, 2017.

[34] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang, "R 3: Reinforced ranker-reader for open-domain question answering," in *AAAI*, 2018.

[35] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," *arXiv preprint arXiv:1906.00300*, 2019.

[36] Y. Nie, S. Wang, and M. Bansal, "Revealing the importance of semantic retrieval for machine reading at scale," *arXiv preprint arXiv:1909.08041*, 2019.

[37] Y. Zhang, P. Nie, A. Ramamurthy, and L. Song, "Answering any-hop open-domain questions with iterative document reranking," *arXiv preprint arXiv:2009.07465*, 2020.