Answer Complex Questions: Path Ranker Is All You Need

Xinyu Zhang[1†], Ke Zhan[1†], Enrui Hu[1†], Chengzhen Fu[1†], Lan Luo[1], Hao Jiang[1], Yantao Jia[1], Fan Yu[1], Zhicheng Dou[2], Zhao Cao[1*] and Lei Chen[3]
[1] Distributed and Parallel Software Lab, Huawei
[2] Gaoling School of Artificial Intelligence, Renmin University of China
[3] Department of Computer Science and Engineering, Hong Kong University of Science and Technology
{zhangxinyu35,zhanke2,huenrui1,fuchengzhen,caozhao1}@huawei.com

ABSTRACT

Currently, the most popular method for open-domain Question Answering (QA) adopts "Retriever and Reader" pipeline, where the retriever extracts a list of candidate documents from a large set of documents followed by a ranker to rank the most relevant documents and the reader extracts answer from the candidates. Existing studies take the greedy strategy in the sense that they only use samples for ranking at the current hop, and ignore the global information across the whole documents. In this paper, we propose a purely rank-based framework **T**hinking **P**ath **R**e-**R**anker (**TPRR**), which is comprised of **T**hinking **P**ath **R**anker (**TPR**) for generating document sequences called "a **path**" and **E**xternal **P**ath **R**eranker (**EPR**) for selecting the best path from candidate paths generated by TPR. Specifically, TPR leverages the scores of a dense model and conditional probabilities to score the full paths. Moreover, to further enhance the performance of the dense ranker in the iterative training, we propose a **"thinking"** negatives selection method that the top-K candidates treated as negatives in the current hop are adjusted dynamically through supervised signals. After achieving multiple supporting paths through TPR, the EPR component which integrates several fine-grained training tasks for QA is used to select the best path for answer extraction. We have tested our proposed solution on the multi-hop dataset "HotpotQA" with a full wiki setting, and the results show that TPRR significantly outperforms the existing state-of-the-art models. Moreover, our method has won **the first place**[1] in the HotpotQA official leaderboard since Feb 1, 2021 under the Fullwiki setting. Code is available at *https://gitee.com/mindspore/mindspore/tree/master/model_zoo/research/nlp/tprr*.

CCS CONCEPTS

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

† Equal contribution.
∗ Corresponding author.
[1]HotpotQA Leaderboard: https://hotpotqa.github.io/. The leaderboard shows the rank with our method name "TPRR". As of May 2021, TPRR is still at the top of the Fullwiki leaderboard.

KEYWORDS

Passage ranking; Path ranker; Multi-hop QA; Multi-turn ranking

ACM Reference Format:
Xinyu Zhang, Ke Zhan, Enrui Hu, Chengzhen Fu, Lan Luo, Hao Jiang, Yantao Jia, Fan Yu, Zhicheng Dou, Zhao Cao and Lei Chen. 2021. Answer Complex Questions: Path Ranker Is All You Need. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462942

1 INTRODUCTION

Open-domain Question Answering (QA) is a task that aims to answer questions over large-scale text documents. Most recent methods adopt the "*Retriever & Reader*" pipeline [4, 24, 28], where the retriever firstly retrieves a list of candidate documents given the question, and then the reader extracts the answer from these candidates. Since retriever is the crucial building block of this pipeline, it will be elaborated in the following. It fetches the supporting documents for answer extraction. Traditional approaches usually select the dense retriever to represent the question and documents separately and implement retrieval via the technique of maximum inner-product search (MIPS) [12]. However, the two-tower dense retriever possesses weak interactions between query and documents [3, 9, 14] , and suffers from the irrelevance of questions and documents. Therefore, recent studies attempt to use the pretrained language models (PLM [13]) as the "re-ranker" to select the top-1 candidate for extracting answer, which is the state-of-the-art approach in the open-domain QA [5].

In fact, the ranking mechanism has been widely used in the single-hop QA where the answer of the query is explicit in a single piece of text evidence. For some complex queries where we focus on multi-hop QA[29], the role of ranker is more prominent. Most answers of multi-hop questions (*e.g.*, the question in Figure 1) need more than one evidence piece to reason, and each evidence piece can be associated through bridge entities [31]. Therefore, it is far from enough for the retriever to conduct only one round of retrieval. It is also necessary to implement multi-turn retrieval and iterative query updating. A good retriever should be able to generate progressive candidate sets [27, 30] and supporting document sequences called **path**s [1]. After the interaction of each round of retrieval results, the best candidate provided for the next round should be ranked first by the ranker in the current hop.

Recent studies about ranking in multi-hop retrieval can be summarized into two categories. The first group of methods, such as

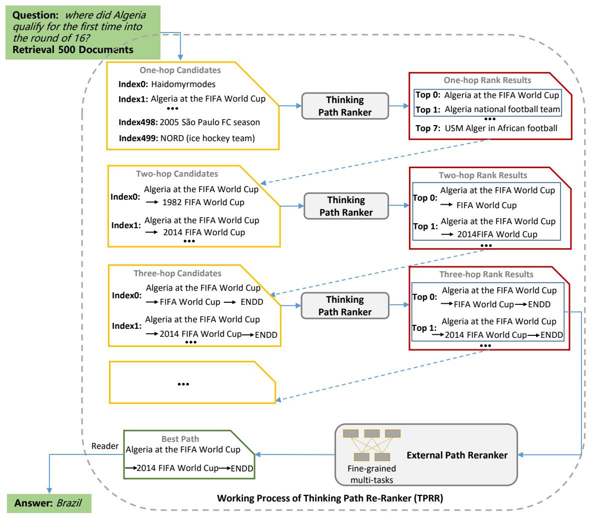
Question: where did Algeria qualify for the first time into the round of 16?
Retrieval 500 Documents
One-hop Candidates
Index0: Haidomyrmodes
Index1: Algeria at the FIFA World Cup
Index498: 2005 São Paulo FC season
Index499: NORD (ice hockey team)
Thinking Path Ranker
One-hop Rank Results
Top 0: Algeria at the FIFA World Cup
Top 1: Algeria national football team
Top 7: USM Alger in African football
Two-hop Candidates
Index0: Algeria at the FIFA World Cup → 1982 FIFA World Cup
Index1: Algeria at the FIFA World Cup → 2014 FIFA World Cup
Thinking Path Ranker
Two-hop Rank Results
Top 0: Algeria at the FIFA World Cup → FIFA World Cup
Top 1: Algeria at the FIFA World Cup → 2014FIFA World Cup
Three-hop Candidates
Index0: Algeria at the FIFA World Cup → FIFA World Cup → ENDD
Index1: Algeria at the FIFA World Cup → 2014 FIFA World Cup → ENDD
Thinking Path Ranker
Three-hop Rank Results
Top 0: Algeria at the FIFA World Cup → FIFA World Cup → ENDD
Top 1: Algeria at the FIFA World Cup → 2014 FIFA World Cup → ENDD
Reader
Best Path
Algeria at the FIFA World Cup → 2014 FIFA World Cup → ENDD
External Path Reranker
Fine-grained multi-tasks
Answer: Brazil
Working Process of Thinking Path Re-Ranker (TPRR)

Figure 1: The overview of the TPRR framework.

DDRQA [30] and IRRR [22], conducts retrieval and ranking alternately. They first retrieve top-K candidates from corpus, then leverage a separate ranking model(*e.g.*, graph neural network [25] or PLM) to select top-1 candidate and use it to update query (*e.g.*, concatenate with the original question). The updated query is then used for the next round of retrieval until the final answer is found. The second group tends to utilize the retrieval scores of the candidate set in each hop for re-ranking the retrieval paths through beam or greedy search [8]. Recurrent-neural-network retriever (RNN retriever) [1] has adopted this strategy.

However, from the experimental evaluation index "*top-K Paragraph Extract Match (PEM)*" [2, 11] provided by these work, in the ranking modes mentioned above, there is still a big gap between ranked results and correct results. This situation can be attributed to the two reasons in the training stage of the ranker. **Firstly**, in each re-ranking process of multi-hop retrieval, the ranker is only trained by the labels in the current turn, which makes the training task of the ranker be a greedy task. The ranker trained in such a way lacks the awareness of long term rewards obtained from each sample. It tries to rank the positive sample of the current hop to the top-1, but may lack the ability of finding good full paths in the whole retrieval process. In current approaches, such as RNN retriever, there is a gap between "*greedy*" training strategy and "*beam*" inference strategy. **Secondly**, the negative samples for training the ranker are not "*hard*" enough. In the works above, the negative samples for ranker are usually obtained by in-batch method [16] or random sampling from a fixed candidate set [18]. The distribution of negative samples will have a great impact on the ranking performance of the retriever or ranker, which has been proved in the paper about ANCE[26]. This paper has also demonstrated that the in-batch and random sampling methods can bring weaker gradients updating to the dense retriever or ranker.

To address these challenges, we introduce a simple path-ranking recursive framework named **T**hinking **P**ath **R**e-**R**anker (**TPRR**), which can transfer the retrieval candidates corresponding to multi-hop question to a reasoning supporting document path, and select the best path for answer extraction from multiple candidate paths. There are two important modules in TPRR. one is the **T**hinking **P**ath **R**anker (**TPR**) for generating supporting document sequences, and the other is the **E**xternal **P**ath **R**eranker (**EPR**) for selecting the best path from candidate paths generated by TPR.

Specifically, in order to avoid the greedy training method, TPR first combines the scoring of PLM and conditional probability to construct the ranking score of the whole path for training so that the supervised signal of each hop can be back propagated to previous hops. Meanwhile, to enhance the training of each ranker, we use the ranked top-K candidates in the current hop as the *thinking* negative samples that can be adjusted dynamically through the current hop supervised signal. These negative samples are also used to construct the ranking candidates of the next hop. Due to the end-to-end optimization of TPR, with the increase of training steps, the negative samples of each hop will be harder gradually, the training pattern can fully stimulate PLM's ability of ranking.

The goal of TPR is to generate candidate paths by dynamic ranking, so it uses pure text features and a ranking task for training. However, the coarse-grained text features and a single ranking task are not enough for selecting a best path for QA. Therefore, EPR first splits the text in the candidate path from TPR into a fine-grained format, then uses the PLM to deep fuse the information of different granularity and generate corresponding representation. Finally, we feed these representation to the multiple QA tasks for joint training. The process can make EPR rank the candidate paths from a different view from TPR. It can further select the best path for QA.

Experimental results about PEM on HotpotQA dataset [29] have shown that TPRR outperforms current state-of-the-art "*Retriever & Ranker*" models. As for the HotpotQA leaderboard performance, our method also has won the championship on the official leaderboard at the time of our submission. Our contributions are summarized as follows: (1) *Simple e2e ranking framework*: The proposed TPRR framework only comprises some general PLMs. There are no entities and text spans extraction steps in this framework. (2) *The whole path ranking in TPR*: TPR builds a path scoring mechanism for ranking, which makes the gradients back propagate to the whole path, and provides the model with "overall view". (3) *Thinking training augment in TPR*: TPR constructs a thinking training augment method through dynamic negatives, which can enhance the training difficulty gradually and improve the performance of dense ranker. (4) *Fine-grained features and tasks in EPR*: Compared with TPR, EPR introduces the fine-grained text representation and multi-tasks of QA for joint training. This method can select the best path for answer extraction in a different view from TPR.

2 RELATED WORK

2.1 Open-domain Multi-hop QA

Since Chen et al. [4] has firstly proposed open-domain QA task, many papers have made improvement on the retriever (Nie et al., 2019 [21]; Guu et al, 2020 [10]) and the reader (Ni et al., 2019 [20]) module in the basic pipeline. They all retrieve one document to answer single-hop questions. However, for questions which require finding and reasoning among multiple documents, these one-shot retrieve-and-read models do not work. Yang et al. [29] presented

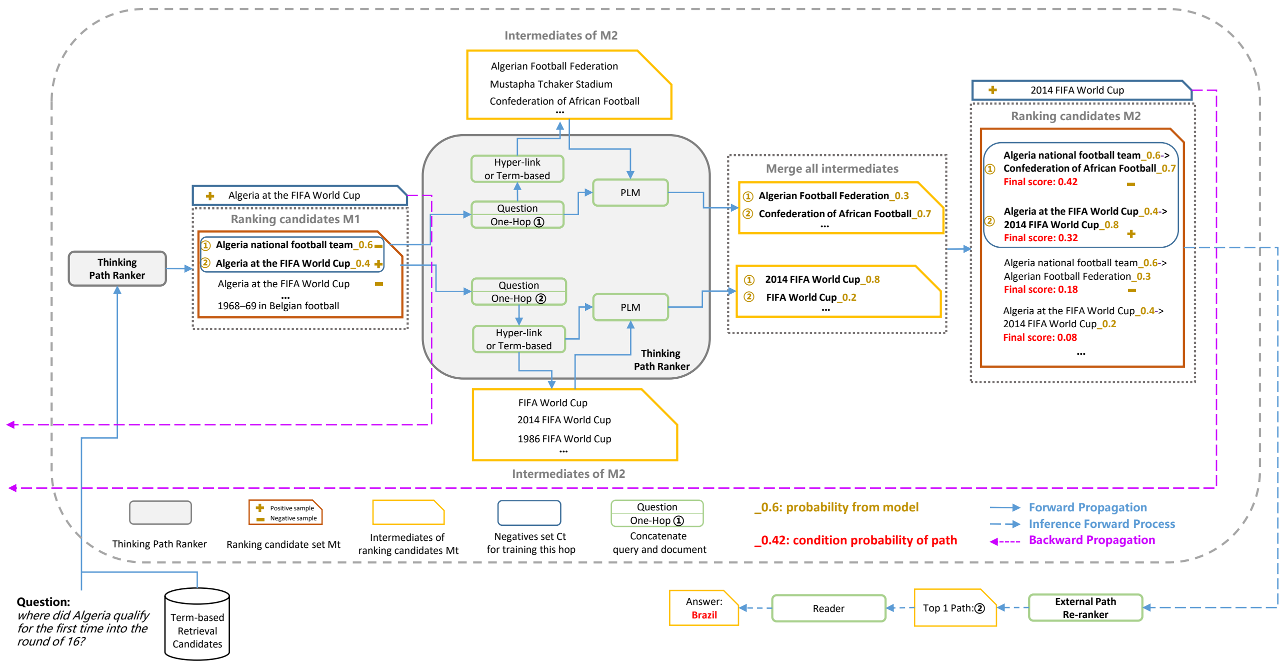
Intermediates of M2
Algerian Football Federation
Mustapha Tchaker Stadium
Confederation of African Football
...
Hyper-link or Term-based
Question One-Hop ①
PLM
Question One-Hop ②
Hyper-link or Term-based
PLM
Thinking Path Ranker
FIFA World Cup
2014 FIFA World Cup
1986 FIFA World Cup
...
Intermediates of M2
Algeria at the FIFA World Cup
Ranking candidates M1
① Algeria national football team_0.6
② Algeria at the FIFA World Cup_0.4
Algeria at the FIFA World Cup
...
1968–69 in Belgian football
Thinking Path Ranker
Merge all intermediates
① Algerian Football Federation_0.3
② Confederation of African Football_0.7
...
① 2014 FIFA World Cup_0.8
② FIFA World Cup_0.2
...
2014 FIFA World Cup
Ranking candidates M2
① Algeria national football team_0.6-> Confederation of African Football_0.7
Final score: 0.42
② Algeria at the FIFA World Cup_0.4-> 2014 FIFA World Cup_0.8
Final score: 0.32
Algeria national football team_0.6-> Algerian Football Federation_0.3
Final score: 0.18
Algeria at the FIFA World Cup_0.4-> 2014 FIFA World Cup_0.2
Final score: 0.08
...
Thinking Path Ranker
Positive sample
Negative sample
Ranking candidate set Mt
Intermediates of ranking candidates Mt
Negatives set Ct for training this hop
Question One-Hop ① Concatenate query and document
_0.6: probability from model
_0.42: condition probability of path
Forward Propagation
Inference Forward Process
Backward Propagation
Question: where did Algeria qualify for the first time into the round of 16?
Term-based Retrieval Candidates
Answer: Brazil
Reader
Top 1 Path:②
External Path Re-ranker

Figure 2: Overview of the forward and backward process in the proposed Thinking Path Ranker (Taking a two-hop question as an example). In the forward process, the ranking candidates M_t of each hop need to be generated through term-based or hyper-links, which have been marked in the red framework. Then, rank the top-K candidates as the current negatives C_t (if positive exists, delete it and postpone it) through conditional path scores, which are marked as the order number and +/- symbols in M_t. Finally, use original C_t to construct the ranking candidates M_{t+1} of the next hop.

HotpotQA, which tests whether the model is applicable to complicated multi-hop questions and uses Wikipedia as the corpus for multi-hop QA to date. A lots of researchers have proposed various methods to address HotpotQA challenge. One kind of method is question decomposition, DecompRC [19] decomposes the complex question into simpler sub-questions and still use the one-shot model for each sub-question. When reasonable sub-questions are not extracted, it's hard to collect evidence correctly. Considering that most QA problems are entity-centric, some papers use entity mention to retrieve reasoning path. Cognitive Graph [6] trains a model to predict the spans for next-hop which helps to extract the most evidential entity. Transformer-XH [31] expands one-shot retrieval results by using entities in the question and retrieval documents to get more documents linked. The evidence coverage is improved but inevitably noise is also introduced for downstream reader. However, entity-extraction usually depends on external tools that exists extraction error and makes pipeline complex.

2.2 Multi-step Retriever for QA

As multi-hop QA requires complex reasoning across multiple documents, a multi-step retriever module that explores evidence documents iteratively is essential. Most recently, Qi introduces GoldEn Retriever [23], which adopts iterative TF-IDF retrieval by generating a new query for each retrieval step. Based on this question reformulation design, DDRQA [30] additionally employs entity-linked document graph to iteratively retrieve, rerank and filter documents, and adaptively determine when to stop the retrieval process. RNN Retriever [1] learns to sequentially retrieve evidence paragraphs to form the reasoning path given the history of previously retrieved paragraphs and the graphical structure. Based on this design, Hop Retriever [17] additionally incorporates implicit entity-level relation into the hyperlinks-based retrieval stage. In the methods above, although there are rankers in the retriever pipelines, each ranker still has two problems discussed in the introduction. One is that the ranker can only pay attention to the ranking of each hop rather than the whole; the other is that the negative samples are too simple in the training process of ranker. This paper will focus on these problems.

3 OUR PROPOSED FRAMEWORK: THINKING PATH RE-RANKER (TPRR)

3.1 Problem Definition

The pipeline of solving multi-hop QA task contains two parts: multi-turn retriever to provide supporting evidences and reader to extract answer spans. More formally, given an open-domain multi-hop question q and a large-scale knowledge source R, the multi-turn retriever collects a retrieval sequence consisting of multiple passage sets $D_q : \{D_1, D_2, \cdots, D_n\}$ (n is the number of retrieval rounds, the total number of documents in D_q is M). Then, select top-K passages $D_{Kq} : \{d_1 \in D_i, d_2 \in D_i, \cdots d_K \in D_i\}$ $(\forall D_i \in D_q, k \ll M)$ that provide sufficient information from D_q to answer q through the ranker. Finally, all supporting evidences in D_{Kq} and q (concatenated together in textual level) are fed into reader to extract answer.

3.2 Overall Architecture

In order to solve the two problems mentioned above of current rankers in multi-hop QA, we propose the Thinking Path Re-Ranker (TPRR) framework. The forward and backward propagation process of TPRR in training stage has been illustrated in Figure 2. It can be seen that the question "*where did Algeria qualify for the first time into the round of 16?*" is answered through a two-hop reasoning process. The best supporting documents include the first hop document (title) "*Algeria at the FIFA World Cup*", where the relevant content "*In 2014, Algeria qualified for the first time into the round of 16*" is mentioned and the second hop document "*2014 FIFA World Cup*", where the answer "*Brazil*" appears in the sentence "*It took place in Brazil from 12 June to 13 July 2014*" of this document. In the forward process, we use the question and top-500 retrieval candidates based on TF-IDF as inputs of the first TPR. In each TPR, there are three main steps. Firstly, each hop ranking candidates M_t can be obtained through recalling from large-scale datasets or using the linked documents based on the hyperlinks in the previous hop. Then, the scores of current whole path are calculated by combining the path scores passed from previous hops and the scores of PLM. Finally, rank top-K (K is 2 in the Figure 2) candidates of the current hop from M_t through path scores, and use them as the negative samples C_t (If a positive sample exists, delete it and postpone a new negative) for training the PLM in the current hop. Meanwhile, the original negatives C_t without positive sample deletion in current hop are used to construct the ranking candidates M_{t+1} of next hop. In the backward process of the thinking path ranker, since the ranking scores of the previous hops are introduced into the calculation of the current hop, the supervision signal applied to current hop can also be back propagated to previous. After outputting the multiple candidate paths through TPR, the fine-grained multi-tasks re-ranker EPR is used to select the best path from the candidate paths. Finally, the reader extracts the final answer from the best path. **In the following sections, Sections 3.3 and 3.4 focus on the two innovative modeling designs of TPR respectively. Section 3.5 presents the training details of TPR. In Section 3.6, we discuss the paths inference in TPR. Finally, Section 3.7 introduces the modeling process and function of EPR.**

3.3 Path Scoring based on TPR

In this section, we present how the path scores are computed based on TPR. Taking two-hop ranking as an example, the ranking process based on TPR can be formulated as Equation (1) by the conditional probability of the path, where d_i^1 is the $i-th$ element of the ranked set from one-hop TPR, d_j^2 is from the two-hop, and $P\left(d_j^2, d_i^1 \middle| q\right)$ represents the conditional probability score of the path from d_i^1 to d_j^2 in the case of question q.

$$P\left(d_j^2, d_i^1 \middle| q\right) = P\left(d_j^2 \middle| q, d_i^1\right) \cdot P\left(d_i^1 \middle| q\right) \quad (1)$$

By generalizing Equation (1) to multi-hop form, it can be rewritten as Equation (2). $P\left(d^1, d^2, \cdots d^t \middle| q\right)$ can be regarded as the score of the whole path, and it can be factorized by conditional probability chain rule.

$$P\left(d^1, d^2, \cdots d^t \middle| q\right) = \prod_{t=1}^{n} P\left(d^t \middle| q, d^1, \cdots d^{t-1}\right) \quad (2)$$

The conditional probability $P\left(d^t \middle| q, d^1, \cdots d^{t-1}\right)$ can be determined by the function $f\left(q, d^1, \cdots, d^t\right)$ that includes a PLM dense ranker (Bert-base is used in this paper). Then, we separate q and d with [SEP] tokens, prefix a [CLS] token, and append a final [SEP] token.

$$\text{input}_{\text{BERT}}\left(\mathbf{q}, \mathbf{d}\right) = [\text{CLS}]q[\text{SEP}]d^1[\text{SEP}]\cdots d^t[\text{SEP}] \quad (3)$$

The output of the scoring function $f(\cdot)$ can be calculated by the projection for CLS vector of Bert, where the projection matrix W_p can project the CLS vector into a scalar.

$$f\left(q, d^1, \cdots d^t\right) = \text{W}_\text{p}\text{BERT}_{\text{CLS}}\left(\text{input}_{\text{BERT}}\left(\mathbf{q}, \mathbf{d}\right)\right) \quad (4)$$

Furthermore, $P\left(d^t \middle| q, d^1, \cdots d^{t-1}\right)$ can be derived by Equation (4), where $\mathbf{M_t}$ shown in the corresponding part of Figure 3 is the **ranking candidate set** of the t-hop.

$$P\left(d_m^t \middle| q, d^1, \cdots d^{t-1}\right) = \frac{\exp f\left(q, d^1, \cdots d^{t-1}, d_m^t\right)}{\sum_{m' \in \mathrm{M_t}} \exp f\left(q, d^1, \cdots d^{t-1}, d_{m'}^t\right)} \quad (5)$$

Therefore, after obtaining the condition probability in through Equation (5), we first use the Equation (2) to calculate the path scores up to the current hop t. Then, according to the scores, we select top-K candidate paths denoted as $\mathbf{C_t}$ as **the negative samples for training t-hop ranker** (details in Equation (6)). Finally, we use the current negative samples to generate ranking candidates for the next hop (details in Equation (7)), and so on. The negatives selection process has been detailed in Section 3.4, and an example calculation of the condition probability in the Equations (5), (6), and (2) is shown in Figure 3.

3.4 Thinking Training Augment in TPR

In this section, we analyze the thinking negatives selection process and training augment theory. At the end of Section 3.3, we extractes top-K documents from the current ranking candidate set $\mathbf{M_t}$ through path scores, and the top-K documents are used as the negative set $\mathbf{C_t}$ for calculating the ranking loss of this hop. The output probability can be calculated as follows.

$$P\left(d_c^t \middle| q, d^1, \cdots d^{t-1}\right) = \frac{\exp f\left(q, d^1, \cdots d^{t-1}, d_c^t\right)}{\sum_{c' \in \mathrm{C_t}} \exp f\left(q, d^1, \cdots d^{t-1}, d_{c'}^t\right)} \quad (6)$$

Meanwhile, the top-K negatives set $\mathbf{C_t}$ is generated as the follow.

$$\mathbf{C_t} = \underset{c \in \mathrm{M_t} = \mathrm{G}(\mathrm{c_{t-1}})}{\text{TopK}} \left(P\left(d^1, d^2, \cdots d_c^t \middle| q\right)\right) \quad (7)$$

In the t-th hop, the function $G(\cdot)$ can generate the ranking candidate set $\mathbf{M_t}$ ($|\mathbf{M_t}| > |\mathbf{C_{t-1}}|$) based on the negative set $\mathbf{C_{t-1}}$ in the previous hop, where term-based retriever or hyperlinks methods can be utilized for generating this new set. In order to simplify the operation in the training process, we fix the number of the ranking candidates corresponding to each d_c^t, which is shown in the process of "from d13 to d21, d22, d23" in Figure 3.

The function TopK $(\cdot)$ can select top-K candidates with higher path scores as the negative set $\mathbf{C_t}$. The selection process is shown in Figure 3. In the red dotted framework, the top-2 candidates are

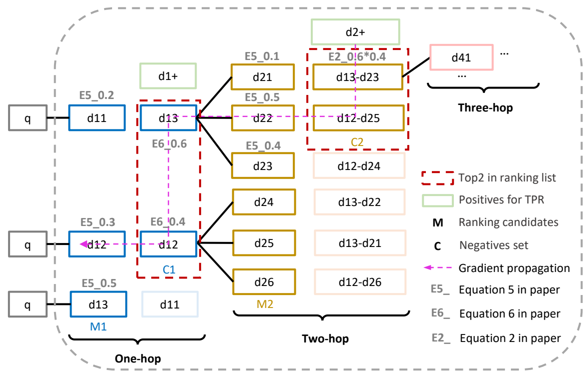

Figure 3: The negative generation process and gradient back-propagation of thinking training augment in TPR.

selected as the negative samples of the training in the current hop, and further generate the ranking candidates of the next hop.

More importantly, the "top-K" mechanism can generate "harder" negative samples for the training of each TPR. In fact, there exists a general training strategy for dense retriever and ranker, that is, *negative samples with near zero loss have near zero gradients and contribute little to model convergence. The convergence of dense retrieval model training relies on the informativeness of constructed negatives*, which has been proved by ANCE[26] and it is verified that the maximum gradient gain can be obtained by selecting the samples with higher scores predicted by the recent model as the training negatives.

TPR adopts the idea as the backup, and designs a thinking negatives generation mechanism which can dynamically enhance the ranker training in a multi-hop retrieval. In Figure 3, the supervised signal of each hop can be back propagated through the access formed by $\mathrm{TopK}(\cdot)$. Therefore, the selected top-K samples can be adjusted with the parameters of the dense ranker in the TPR, which means that the confusion degree of negative samples is proportional to the convergence of the model. In Equation (8), $\mathrm{H}(\mathrm{C_t})$ represents the confusion degree with the positives, the right term is the reciprocal of gradient norm. When the gradient is close to zero, it means that the model is converging, which can produce more confusing negative samples.

$$\mathrm{H}(\mathrm{C_t}) \propto \left\| \nabla \log P\left(d^1, d^2, \cdots d^t \middle| q\right)\right\|^{-1} \tag{8}$$

3.5 Training and Objective Functions in TPR

In order to obtain better ranking performance, the list-wise ranking cross-entropy is adopted. The loss function of TPR in each hop can be formulated, where $|\mathrm{C_t}|+1$ is the size of list wise ($|\mathrm{C_t}|$ negatives and 1 positive), p_i are the real labels of the candidate paths.

$$L^t = -\sum_{i=1}^{|\mathrm{C_t}|+1} p_i \log\left(P\left(d^1, d^2, \cdots d_i^t \middle| q\right)\right) \tag{9}$$

Calculate the first-order gradient for Equation (9), the following equivalent transformation can be carried out.

$$\begin{aligned}\nabla L^t &= \nabla \log\left(P\left(d^1, d^2, \cdots d_i^t \middle| q\right)\right)\\ &= \sum_{t=1}^{n} \nabla \log\left(P\left(d^t \middle| q, d^1, \cdots d^{t-1}\right)\right)\end{aligned} \tag{10}$$

The first order gradient of the whole pipeline loss can be described as Equation (11), which means that each TPR (at the t-th hop) is supervised by $n-t+1$ times in a backward process. Users should have the opportunity to adjust the gain of each gradient term at the loss function to obtain the best training performance.

$$\nabla L = \nabla \sum_{t=1}^{n} L^t = \sum_{t_1=1}^{n} \sum_{t_2=1}^{n} \nabla \log\left(P\left(d^t \middle| q, d^1, \cdots d^{t_2-1}\right)\right) \tag{11}$$

Therefore, we add a series of soft-weighted factors α_t to the whole path loss function so as to adjust the gradient gain of each hop model (the default setting is 1). The final loss function for the whole path can be described by the follow.

$$L = \sum_{t=1}^{n} \alpha_t L^t, \sum\nolimits_t \alpha_t \Big/ n = 1 \tag{12}$$

Meanwhile, to make TPR stop ranking automatically after a certain number of hops, we add the end document tag "**ENDD**" to the ranking candidates set $\mathbf{M_t}$ of each hop during training, and insert the END hop after the last hop in each training sample, where the positive sample is "**ENDD**". The whole process is shown in Figure 4, which can make the model learn the ability of stopping iteration automatically when "**ENDD**" is ranked in top-1 of the ranking candidate set $\mathbf{M_t}$.

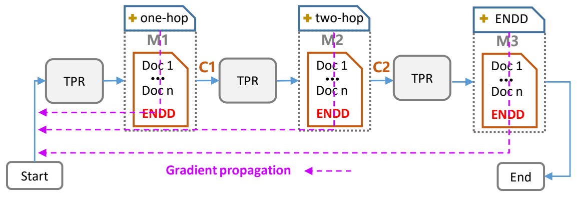

Figure 4: The end document "ENDD" is added in each $\mathrm{M_t}$ set, and ranking it to the top-1 means that stop iteration.

3.6 Adaptive Path Inference in TPR

Similar to the training process, inference of TPR contains a paths generation process based on conditional probability score and the sum of scores for all paths is 1, which is shown in Equation (13), where p_i represents the $i-th$ path.

$$\sum\nolimits_i P\left(p_i \middle| q\right) = 1, p_i \in \left\{d^1, d^2, \cdots d^t\right\} \tag{13}$$

Therefore, users can set their own threshold δ_S $(0 < \delta_S \leq 1)$ to select the number of paths, that is, sum the path scores from high to low until it exceeds or is equal to the threshold, and output all accumulated paths. The following formulation has shown this process, where P_{ranked} represents the probability and it is ranked from high score to low, S is the number of selected paths, p_S is the selected paths set.

$$p_S = \left\{ p_i \middle| p_i \in \sum_{i=1}^{S} P_{ranked}\left(p_i \middle| q\right) \geq \delta_S \right\} \tag{14}$$

In order to further improve the performance of paths generation, we conduct beam search for each hop candidates in the selected paths and re-grouped and re-rank all paths.

3.7 External Path Re-ranker and Reader

In order to generate the related candidate paths and keep the iterative framework concise, we only adopt the ranking task and pure paragraph-level text concatenation for TPR. However, aiming at the QA task, the best path for answer extraction should be selected carefully, which means that the pure features and task in TPR are not enough. Therefore, we introduce a fine-grained external path re-ranker (EPR), which can refine the text features in the existing paths, and establish different joint training tasks for QA to re-rank the top-1 path for reader from the candidate paths.

Fine-grained Path Context Encoder To form a more fine-grained path text representation than TPR, we use the following special tokens to divide the text in the path into three granularity: title, sentence and paragraph, and form a new text format. Meanwhile, we use the Transformer-layer to generate the vector representation on the positions of these special tokens.

$$[\text{CLS}]\,[q]\,question\,[/q] < t > title_1 < /t >$$
$$sent_{1,1}\,[s]\,sent_{1,2}\,[s] \cdots [\text{SEP}]$$

Multi-task Joint Prediction The fine-grained context encoding vectors have been obtained. Then, we use these vectors to implement more fine-grained joint QA-oriented training tasks. Three tasks are selected: (i) predict path ranking scores through the vector of [CLS]; (ii) predict supporting paragraphs through the representation on the special token <t>; (iii) predict supporting sentences based on the vector of token [s]. All three tasks are jointly trained through Binary Cross-Entropy (BCE) Loss, which can make the model give full consideration to the performance of QA, so as to select the best path to answer questions. HGN-reader[7] describes the loss functions of each task, and will not be repeated for simplicity. Finally, after equipping the selected best path by EPR into the PLM reader, the final answer is extracted.

4 EXPERIMENTAL SETUP

4.1 Dataset and Evaluation Metrics

We evaluate our method and other compared models on the HotpotQA benchmark[29], which is a crowd-sourced multi-hop question answering dataset. The dataset with 113k Wikipedia-based question-answer pairs is divided into three parts, in which there are 90,654 pairs for training, 7,405 pairs for development and other 7,405 pairs for testing. Each pair include sentence-level supporting facts as evidence besides the question and answer. There exist two reasoning types required to answer the questions. The first is **Bridge** which means that some bridge entities must be identified along the reasoning path, and the other is **Comparison** which requires the model understand the properties compared between two entities in questions. **80%** of the questions in the dataset are bridge type, like *"Which team does the player named 2015 Diamond Head Classic's MVP play for ?"* which requires finding bridge *"Buddy Hield"* corresponding to entity *"the MVP"* to answer it. **20%** are Comparison questions, for example, *"Did LostAlone and Guster have the same number of members ?"* needs comparing the member number of two band. We focus on the full wiki setting which requires the model to retrieve multi-hop supporting paragraphs from all Wikipedia pages.

We evaluate both the performance on the coverage of the ground-truth supporting paragraphs in the retrieval results and the QA benchmark metrics. Firstly, we measure the coverage accuracy using the Paragraph Exact Match (PEM), which calculates the percentage of questions whose supporting paragraphs can be all exactly retrieved. Then for QA performance, the standard answer and supporting facts Exact Match (EM) and F1 score are reported separately for answer extraction and supporting sentences prediction. The joint Exact Match (EM) and F1 score are measured for the overall comparison. Both the QA testing results on development set and test set are provided.

4.2 Experimental Settings

Thinking Path Ranker. We implement TPR based on Bert-base. Firstly, due to the property of HotpotQA data, the number of hops is set as 2. Then, about the setting of ranking candidates for each question, the top 50 TF-IDF-based paragraphs from 5M paragraphs of Wikipedia are used as the one-hop initial candidates, the value of top-K in the one-hop is 8, and the four two-hop ranking candidates corresponding to each one-hop paragraph are generated through the hyper-links of the top 8 one-hop paragraphs. Meanwhile, in the training stage, batch size is set to 2, the learning rate is set to 3e-5 with the linear scheduling warm-up (rate 0.1), and select the default loss mode. We also expand the positive training paths for comparison questions by reversing the golden paths in the training stage. In the inference, in order to compare with the existing work, the number of output paths is set to 8, and we use beam search for further performance with the beam size 8 and bidirectional hyperlinks. Finally, in terms of the different PLM training modes in the ranker, TPR has two versions:

- **Independent version**: It is short for **Ind**. PLM of each hop does not share model parameters and is initialized independently.
- **Share version**: PLM of each hop uses the same trainable model parameters.

In terms of the size of search space in the inference, TPR also owns two versions:

- **TPR-base**: The candidates for each question is originated from top-500 retrieved from 5M paragraphs of Wikipedia through TF-IDF.
- **TPR-plus**: Similarly, the candidates are from the top-3000.

External Path Re-ranker. Firstly, to achieve better performance, we use the Albert-xxlarge [15] in the EPR. Then, the training set of the EPR is built based on the output paths of the TPR, where a question corresponds to a positive path and eight negative path. Finally, in the training stage, batch size is set to 32, learning rate is set to 1e-5, the weight of the supporting sentence prediction is 5, others are 1.

5 RESULTS AND ANALYSIS

5.1 Overall Performance

We implement the TPRR framework (*i.e.* independent version, TPR-base/plus, with EPR) on the HotpotQA full wiki setting, the compared results with some once front-page state-of-the-art (SOTA) models on the leaderboard are shown in Table 1. The development results of other models are originated from published papers, and

Table 1: **Performance of answer extraction and supporting sentence prediction on HotpotQA fullwiki setting.**

Models	Dev						Test					
	Ans		Sup		Joint		Ans		Sup		Joint	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
DecompRC [19]	-	43.30	-	-	-	-	30.00	40.65	-	-	-	
Cognitive Graph QA [6]	37.55	49.40	23.11	58.52	12.18	35.28	37.12	48.87	22.82	57.69	12.42	34.92
Semantic Retrieval [21]	46.41	58.70	39.86	71.53	26.53	49.00	45.32	57.34	38.67	70.83	25.14	47.60
Transformer-XH [31]	54.00	66.20	-	-	-	-	51.60	64.07	40.91	71.42	26.14	51.29
RNNRetriever [1]	60.49	73.30	49.16	76.05	35.82	61.43	60.04	72.96	49.08	76.41	35.35	61.18
DDRQA	62.90	76.90	-	-	-	-	62.53	75.91	51.01	78.86	36.04	63.88
HopRetriever [17]	66.56	79.21	56.02	81.81	42.01	68.97	64.83	77.81	56.08	81.79	40.95	67.75
IRRR [22]	65.74	78.41	-		-	-	65.71	78.19	55.93	82.05	42.14	68.59
TPRR-Ind-Base(avg)	**66.89**	**79.61**	**58.49**	**83.23**	**44.24**	**70.40**	-	-	-	-	-	-
TPRR-Ind-Plus(avg)	**67.33**	**80.08**	**60.20**	**84.47**	**45.31**	**71.40**	**66.95**	**79.50**	**59.43**	**84.25**	**44.37**	**70.83**

the test results are from the current leaderboard. In the models in Table 1, some SOTA models are listed as follows.

- **DDRQA/IRRR**:the models that retrieval and ranking are conducted alternately.
- **RNNRetriever/HopRetriever**: the models that after retrieval, utilize the retrieval scores of candidate set in each hop for re-ranking the search paths through beam search.

Since **the experimental results of baseline are all from papers and leaderboard**, we use the method of **"average by repeated experiments"** instead of the significance test. All the experimental results labeled by **"avg"** were **repeated for 10 times** and the results shown in each table are the averaged values.

In Section 1, we have discussed the drawbacks of the two kinds of methods mentioned above, TPRR overcome these problems in design and outperforms the existing models in most of the evaluation metrics. As of **February 2021**, our method has won **the first place** in the HotpotQA official leaderboard. **Our proposed model TPRR significantly outperforms existing best models on joint-F1 (the index for ranking on the HotpotQA leaderboard) by approximate 2%.**

In terms of the path ranking performance, we compared RNNRetriever and HopRetriever which can output ranking paths and have provided the top 1 and 8 PEM in their published papers. To be fairly compared, TPRR uses the same initial retrieval candidate set (*i.e.* top-500 documents based on TF-IDF scores) and PLM (*i.e.* Bert-base) with RNN and Hop Retriever. In Table 2, the tags "bdg", "cmp", "total" represent the "bridge", "comparsion" and overall respectively. Notably, TPRR can outperform the RNNRetriever and HopRetriever that is the current SOTA on the top-1 and top-8 PEM. Especially, TPRR has more significant improvement on the bridge type (**outperform SOTA by approximate 4%**), which demonstrates that the reasoning document paths used to support the bridge type questions can match the modeling mechanism of TPRR. For the comparison type question without explicit reasoning paths, the performance of TPRR is comparable to HopRetriever.

Table 2: Supporting documents collection results on different types of questions.

Model	top-1 PEM			top-8 PEM		
	bdg	cmp	total	bdg	cmp	total
RNN Retriever	70.77	**86.42**	73.91	86.63	**90.38**	87.39
Hop Retriever	82.11	84.26	82.54	90.01	85.41	89.09
TPRR-Ind-Base(avg)	**86.67**	84.26	**86.19**	**93.43**	85.14	**91.77**

5.2 Ablation Analysis

The basic configuration of the ablation experiments about in TPR is the same as that of **TPR-Ind-Base**. **All the experimental results still are averaged through ten times experiments.**

Effectiveness of path scoring mechanism in TPR. In Section 3.3, we design the method to calculate the score of the whole ranking path, and we have analyzed that this modeling method can implement the back propagation of supervision signal on the whole path. To verify the exactness of the conclusion, we select a pre-trained one-hop ranker (offline training through one-hop samples) as the initialization of one-hop PLM, and remove the supervision of one-hop and only apply the supervised signal on the two-hop. Then, we design two experiments: one is that fix the parameters of one-hop ranker and only supervise the two-hop ranker, the other is that relay on the supervised signal to adjust the one-two hop ranker. The process has been shown in the following Figure 5.

Under the same checkpoint, the experimental results are shown in Table 3, where "ol" and "ps" represent "one-hop loss" and "path supervision". The results can notably show the advantages of full-path scoring. The initial one-hop ranker was trained only through one-hop supervised signals independently, when it is fixed, the one-hop ranker can not be corrected by the ranking loss of the two-hop so that the error of one-hop without correction is continuously propagated to the second-hop. On the contrary, the introduction of the full path scoring mechanism makes the supervised signal applied to the two-hop also be back propagated to the ranker in the one-hop, which plays a good role in correcting the one-hop ranker through the path scoring.

Table 3: Ablation experiments results about the effect of path scoring in TPR.

Model	top-1 PEM of TPR			top-8 PEM		
	bdg	cmp	total	bdg	cmp	total
TPRR-Ind-Base	79.15	82.16	79.75	93.43	85.14	91.77
w/o ol + w/o ps	77.85	79.46	78.17	92.48	82.84	90.55
w/o ol + w/ ps	**78.95**	**82.16**	**79.59**	**93.04**	**84.79**	**91.39**

Table 4: Ablation experiments results about thinking training augment of TPR.

Model	one-hop		top-8 PEM		
	bdg cnt	cmp cnt	bdg	cmp	total
TPRR-Ind-Base	**5,676**	**1,260**	**93.43**	**85.14**	**91.77**
w/o tns	5,505	1,248	90.37	84.26	89.15

Impact of thinking training augment in TPR. Section 3.4 has introduced that TPRR can select the ranked top-K candidates as negatives to calculate the ranking loss. It is a "thinking" method to enhance the difficulty of distinguishing positives. In the current two-hop experiments, in order to verify its effectiveness, we generate a set of fixed one-hop negatives for each question based on TF-IDF and select top-8 (same as the basic mode), then select the same checkpoint as basic mode for evaluation. The results are shown in Table 4, where tns is short for thinking negative samples. On average, **TPR with tns can outperform the mode without tns by more than 3% in the top-8 PEM.** Then, the "cnt" in the "one-hop" column of this table represents the number of correct predictions in the one-hop, which has shown the significant improvement of thinking negative samples on one-hop ranker.

In order to explain these results, we plot the whole loss curves of the compared models. They own the same top-K value (is set as 8), which is shown in Figure 6. Based on the negatives effect analysis in ANCE[26], the better dense retriever with hard negatives owns a high loss that can be decreased gradually. The loss curves of Figure 6 verifies this point, where the yellow curve with hard negatives is always above the blue curve with fixed negatives. The difference of the height marked in the red line is determined by the difficulty degree of negative samples.

5.3 Performance of Different TPR Versions

In Section 4.3, TPR framework can be derived into four versions from two terms. The inference performances of TPR-base/plus are shown in Table 1, which demonstrates that expanding the search space can improve the upper bound of ranking results. Although more noise will be introduced, the performance of TPR is still improved, which further verifies that TPR owns strong robustness.

The main difference between TPR-share and TPR-independent is whether the PLM parameters in each hop are the same in the training stage. Under the TPR-base setting, the experimental results are shown in the Table 5. On the whole, two versions have outperformed the current SOTA results. The independent version is better than shared version. It means that the different models on each hop

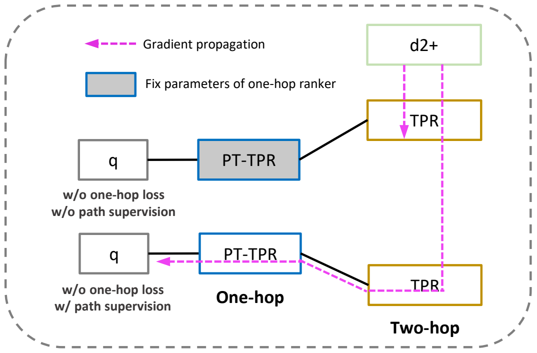

Figure 5: Ablation experiments design about path scoring of TPR, in order to verify the effect of two-hop supervised signal applied on the one-hop.

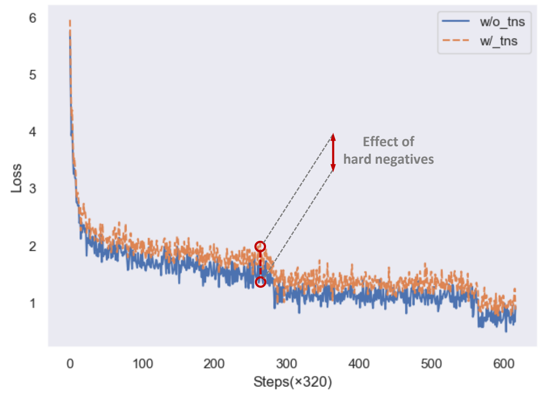

Figure 6: The loss compared curves of w/ tns and w/o tns, and the curves are smoothed at 320 intervals.

can expand the adjustment space and capacity. Meanwhile, the path scoring of TPR can link the different rankers in each hop, and stimulate the potential of each model to obtain the maximum benefit of the whole path ranking. From another perspective, although the performance of the share version model is slightly decreased, it can save trainable parameters and is easy to train and inference quickly. Therefore, users can choose different TPR versions according to computational resources.

5.4 Effect of External Path Reranker

EPR can combine fine-grained information from cross-pages of the path and use more abundant optimization objectives than TPR to select the top-1 path from "another perspective". In order to demonstrate the re-ranking performance of EPR, we adopt the output paths of TPR-base/plus as the input of EPR and evaluate top-1 PEM of the selected single path. The results are shown in Table 6, where "TPR" represents the "TPR-Ind" mode. The experimental results in two modes have shown that **EPR can enhance top-1 (best path for answer extraction) PEM by at least 6%** so that it can improve the order of the best path to a higher level.

Meanwhile, there is a reason why the top 1 index of TPR is lower than that of EPR, that is, the training objectives of the two

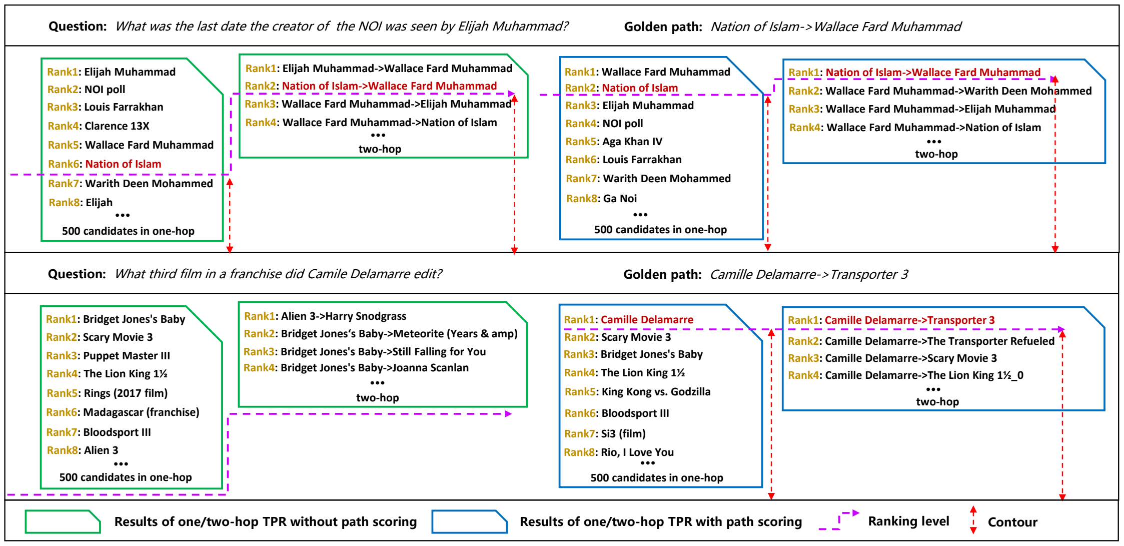
Question: What was the last date the creator of the NOI was seen by Elijah Muhammad?
Golden path: Nation of Islam->Wallace Fard Muhammad
Rank1: Elijah Muhammad
Rank2: NOI poll
Rank3: Louis Farrakhan
Rank4: Clarence 13X
Rank5: Wallace Fard Muhammad
Rank6: Nation of Islam
Rank7: Warith Deen Mohammed
Rank8: Elijah
500 candidates in one-hop
Rank1: Elijah Muhammad->Wallace Fard Muhammad
Rank2: Nation of Islam->Wallace Fard Muhammad
Rank3: Wallace Fard Muhammad->Elijah Muhammad
Rank4: Wallace Fard Muhammad->Nation of Islam
two-hop
Rank1: Wallace Fard Muhammad
Rank2: Nation of Islam
Rank3: Elijah Muhammad
Rank4: NOI poll
Rank5: Aga Khan IV
Rank6: Louis Farrakhan
Rank7: Warith Deen Mohammed
Rank8: Ga Noi
500 candidates in one-hop
Rank1: Nation of Islam->Wallace Fard Muhammad
Rank2: Wallace Fard Muhammad->Warith Deen Mohammed
Rank3: Wallace Fard Muhammad->Elijah Muhammad
Rank4: Wallace Fard Muhammad->Nation of Islam
two-hop
Question: What third film in a franchise did Camile Delamarre edit?
Golden path: Camille Delamarre->Transporter 3
Rank1: Bridget Jones's Baby
Rank2: Scary Movie 3
Rank3: Puppet Master III
Rank4: The Lion King 1½
Rank5: Rings (2017 film)
Rank6: Madagascar (franchise)
Rank7: Bloodsport III
Rank8: Alien 3
500 candidates in one-hop
Rank1: Alien 3->Harry Snodgrass
Rank2: Bridget Jones's Baby->Meteorite (Years & amp)
Rank3: Bridget Jones's Baby->Still Falling for You
Rank4: Bridget Jones's Baby->Joanna Scanlan
two-hop
Rank1: Camille Delamarre
Rank2: Scary Movie 3
Rank3: Bridget Jones's Baby
Rank4: The Lion King 1½
Rank5: King Kong vs. Godzilla
Rank6: Bloodsport III
Rank7: Si3 (film)
Rank8: Rio, I Love You
500 candidates in one-hop
Rank1: Camille Delamarre->Transporter 3
Rank2: Camille Delamarre->The Transporter Refueled
Rank3: Camille Delamarre->Scary Movie 3
Rank4: Camille Delamarre->The Lion King 1½_0
two-hop
Results of one/two-hop TPR without path scoring
Results of one/two-hop TPR with path scoring
Ranking level
Contour

Figure 7: Case study of the path ranking performance in TPR. The compared models include the basic models "w/o ol + w/o ps" and "w/o ol +w/ps" introduced in Section 5.2.

Table 5: Compared performance about TPR-share and TPR-independent (TPRR-Ind-Base).

Model	top-1 PEM of TPR			top-8 PEM		
	bdg	cmp	total	bdg	cmp	total
Share(avg)	78.56	80.68	78.32	91.37	84.32	89.96
Independent(avg)	**79.15**	**82.16**	**79.75**	**93.43**	**85.14**	**91.77**

Table 6: Top-1 path selection performance of EPR.

Model	top-1 PEM Base			top-1 PEM Plus		
	bdg	cmp	total	bdg	cmp	total
TPR(avg)	79.15	82.16	79.75	80.16	88.76	81.89
TPR+EPR(avg)	**86.67**	**84.26**	**86.19**	**86.99**	**92.40**	**88.08**

components are different. The main goal of TPR framework is to find all supporting paths related to question through paragraph-level text information, which only includes a ranking task. However, EPR needs to find a best supporting path to complete QA task, which needs more fine-grained text features to support search. Therefore, the functions of TPR and EPR are completely different.

5.5 Case Study

In this part, we analyze two real cases of the path scoring ablation models about TPR in Section 5.2. In Figure 7, the green framework represents the inference results of the basic model with a fix one-hop model and without path scoring, and the blue is with path scoring. For each question, there are the same 500 one-hop candidates, and we show the top-8 candidates in the one-hop and the top-4 candidates in the two-hop. In the "green" mode, the supervised signal only can adjust the two-hop model, once there is a deviation in the one-hop model, it will provide a great negative impact for the two-hop ranker. For the first case in the green mode, the one-hop ranker can not rank the correct one-hop document to the first, which directly leads to the low probability that the two-hop ranker can rank the correct position. Similarly, for the second question, if the correct document cannot be selected in the top-8 of one-hop, the model without path scoring has no chance to rank the correct path to the top-1. The blue mode with path scoring can propagate the supervised signal to each hop, the ranking level of which is higher significantly.

6 CONCLUSION

In this paper, we propose the pure ranked-base Thinking Path Re-Ranker (TPRR) framework including TPR and EPR to generate the supporting document sequences (path) for multi-hop QA and extract answer. TPR utilizes the PLM and condition probability for scoring the whole path, which makes the supervised signal back propagate to the whole path. Meanwhile, TPR adopts a thinking negatives augment strategy, which uses the top-K ranked samples in this hop as the negatives and to construct the next candidates. With the thinking negative samples, the ranking effect of dense model can be improved gradually. EPR conducts a jointed training with multiple fine-grained tasks aiming at QA, which uses the features and tasks different from TPR to select the best path from another view. The experimental results show that TPRR significantly outperforms the existing state-of-the-art models.

ACKNOWLEDGMENTS

We thank MindSpore for the partial support of this work, which is a new deep learning computing framework[2]. We are grateful to Ruofei Lai and Yongkang Wu from Huawei Poisson Lab for edits and comments on writing the paper.

[2]https://www.mindspore.cn/

REFERENCES

[1] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *8th International Conference on Learning Representations*.

[2] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval. *CoRR* abs/2010.00768 (2020).

[3] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932* (2020).

[4] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1870–1879.

[5] Rajarshi Das, Ameya Godbole, Dilip Kavarthapu, Zhiyu Gong, Abhishek Singhal, Mo Yu, Xiaoxiao Guo, Tian Gao, Hamed Zamani, Manzil Zaheer, et al. 2019. Multi-step entity-centric information retrieval for multi-hop question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. 113–118.

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

[7] Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2019. Hierarchical graph network for multi-hop question answering. *arXiv preprint arXiv:1911.03631* (2019).

[8] Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806* (2017).

[9] Luyu Gao, Zhuyun Dai, Zhen Fan, and Jamie Callan. 2020. Complementing Lexical Retrieval with Semantic Residual Embedding. *CoRR* abs/2004.13969 (2020).

[10] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909* (2020).

[11] Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. *arXiv preprint arXiv:1804.04264* (2018).

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[13] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[14] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 39–48.

[15] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. (2019).

[16] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300* (2019).

[17] Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2021. HopRetriever: Retrieve Hops over Wikipedia to Answer Complex Questions. In *AAAI*.

[18] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181* (2020).

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

[20] Jianmo Ni, Chenguang Zhu, Weizhu Chen, and Julian McAuley. 2019. Learning to Attend On Essential Terms: An Enhanced Retriever-Reader Model for Open-domain Question Answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (ACL)*. 335–344.

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

[22] Peng Qi, Haejun Lee, Oghenetegiri Sido, Christopher D Manning, et al. 2020. Retrieve, rerank, read, then iterate: Answering open-domain questions of arbitrary complexity from text. *arXiv preprint arXiv:2010.12527* (2020).

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

[24] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *CoRR* abs/2010.08191 (2020).

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[26] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwikj. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*.

[27] Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, et al. 2020. Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval. *arXiv preprint arXiv:2009.12756* (2020).

[28] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718* (2019).

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

[30] Yuyu Zhang, Ping Nie, Arun Ramamurthy, and Le Song. 2020. DDRQA: Dynamic Document Reranking for Open-domain Multi-hop Question Answering. *arXiv preprint arXiv:2009.07465* (2020).

[31] Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2019. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations*.