Beam Retrieval: General End-to-End Retrieval for Multi-Hop Question Answering

Jiahao Zhang[†] **Haiyang Zhang**[†*] **Dongmei Zhang**[†] **Yong Liu**[‡] **Shen Huang**[‡]

[†]Beijing University of Posts and Telecommunications, Beijing, China
{relyourself, zhhy, zhangdm}@bupt.edu.cn

[‡]Tencent Research, Beijing, China
{owenyongliu, springhuang}@tencent.com

Abstract

Multi-hop QA involves finding multiple relevant passages and step-by-step reasoning to answer complex questions. While previous approaches have developed retrieval modules for selecting relevant passages, they face challenges in scenarios beyond two hops, owing to the limited performance of one-step methods and the failure of two-step methods when selecting irrelevant passages in earlier stages. In this work, we introduce Beam Retrieval, a general end-to-end retrieval framework for multi-hop QA. This approach maintains multiple partial hypotheses of relevant passages at each step, expanding the search space and reducing the risk of missing relevant passages. Moreover, Beam Retrieval jointly optimizes an encoder and two classification heads by minimizing the combined loss across all hops. To establish a complete QA system, we incorporate a supervised reader or a zero-shot GPT-3.5. Experimental results demonstrate that Beam Retrieval achieves a nearly 50% improvement compared with baselines on challenging MuSiQue-Ans, and it also surpasses all previous retrievers on HotpotQA and 2WikiMultiHopQA. Providing high-quality context, Beam Retrieval helps our supervised reader achieve new state-of-the-art performance and substantially improves (up to 28.8 points) the QA performance of zero-shot GPT-3.5[1].

1 Introduction

Question Answering (QA) has been a mainstream research in natural language processing (NLP) for a long time. With the development of pretrained language models (PLMs), simple QA tasks can be solved by adopting a BERT-like PLM (Devlin et al., 2019). As a result, researchers have been increasingly drawn to more complex QA benchmarks, such as multi-hop QA. This presents a significant challenge, as it requires reasoning across multiple and diverse documents to accurately answer complicated multi-hop questions. Many high-quality multi-hop QA datasets have been introduced, such as HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020) , MuSiQue (Trivedi et al., 2022) and so on. Figure 1 illustrates an example of an actual question taken from MuSiQue-Ans dataset.

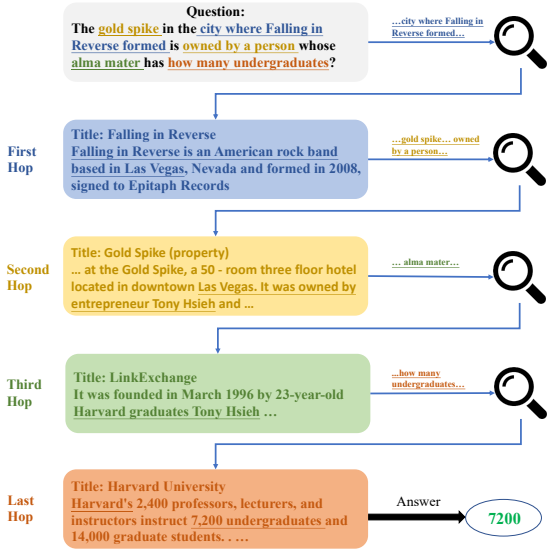

Figure 1: An example of multi-hop QA from MuSiQue-Ans benchmark.This complicated 4-hop question requires the model to select relevant passages based on the question and previous chosen passages.

Mainstream QA models for multi-hop QA often follow a retrieve-and-read paradigm (Zhu et al., 2021), including a passage retriever to filter out extraneous information and a reader to obtain the final answer (Tu et al., 2020; Wu et al., 2021; Trivedi et al., 2022; Li et al., 2022; Yin et al., 2022). However, these methods have primarily focused on two-hop scenarios, exhibiting poor performance in more complex situations and providing low-quality context for downstream QA task.

Previous studies have proposed two types of retrievers for use in reading comprehension settings. One type is one-step methods. SAE (Tu et al., 2020) and MuSiQue SA Selector (Trivedi et al., 2022) concatenate each candidate passage and the question as inputs fed to BERT, then select out the most relevant passages

* Corresponding author

[1]Preprint. Under review. Code is available at https://github.com/canghongjian/beam_retriever

with the highest scores. Such methods do not utilize the dependency between relevant passages, resulting in a limited performance. The other type is two-step methods. S2G (Wu et al., 2021) and FE2H (Li et al., 2022) select the first hop passage in the same way as one-step. At the second stage they identify the second hop relevant passage through pairing the selected passage with the other candidate passages. C2FM (Yin et al., 2022) selects three passages at the first stage, then it combines them two by two and identifies the true passage pair at the second stage. Notice that C2FM will not utilize the unselected passages in the second stage, leaving limitations in retrieval. These methods show strong performance in retrieval. However, they are customized for two-hop issues in HotpotQA and become inapplicable when faced with datasets involving more hops. Furthermore, two-step methods exhibit limited robustness, as the entire retrieval process is susceptible to failure if the first stage identifies irrelevant passages. In conclusion, previous retrievers exhibit poor performance when handling questions with more than 2 hops.

We observe that auto-regressive language generation is quite similar to multi-hop retrieval. Auto-regressive language models (Radford and Narasimhan, 2018; Raffel et al., 2020; Lewis et al., 2020) decode next token based on previous generated tokens, while multi-hop retrieval identifies next hop passage based on previous selected passages (and the original question). Naturally we align multi-hop retrieval with language generation by introducing the concept of text decoding. It has been observed that some failures in previous retrieval systems may have been attributed to issues encountered during their early stages. To reduce the risk of missing hidden relevant passages, we employ the beam search paradigm to multi-hop retrieval and present Beam Retrieval, a general end-to-end retrieval framework for multi-hop QA, by keeping track of multiple partial hypotheses of relevant passages at each step. We begin with the multi-hop question and select B passages with the highest scores from the candidate set with n passages, where scores are obtained by a classification head on top of an auto-encoder model. At subsequent hops, it takes B question-passage pairs as new inputs and selects the next hop's B relevant passages with the highest scores calculated by another classification head. This process continues until the last hop is reached. From this we can see Beam Retrieval is a general multi-hop retrieval framework, as it works in scenarios with more than two hops, just like the way as auto-regressive language generation. Furthermore, Beam Retrieval produces multiple, not just one, hypotheses of relevant passages at each step. This approach expands the search space, enhances the probability of obtaining the truly relevant passages, and mitigates the impact of retrieval errors that may occur in the early stages.

To reduce the gap between training and reasoning, Beam Retrieval is designed to train using the same beam size B as it employs during reasoning. At each hop, Beam Retrieval computes the loss between nearly $B \times n$ hypotheses and the ground-truth relevant passages at each hop. This approach jointly optimizes the encoder and two classification heads by minimizing the combined loss across all hops. In summary, Beam Retrieval produces a chain of relevant passages with the highest score using a single forward pass, effectively learning the entire multi-hop retrieval process. Consequently, it trains and reasons in an end-to-end way.

Beam Retrieval can also serve as a plugin in QA domain, providing high-quality relevant context and enhancing the performance of downstream QA tasks. Based on Beam Retrieval, we implement a multi-hop QA system to extract the answers by incorporating a supervised reader (Li et al., 2022; Yin et al., 2022) or a zero-shot large language model (LLM) (Brown et al., 2020; OpenAI, 2023) following conventional machine reading comprehension setting. We validate Beam Retrieval by extensive experiments on three benchmark datasets MuSiQue-Ans, HotpotQA and 2WikiMultiHopQA, and experimental results demonstrate that Beam Retrieval surpasses all previous retrievers by a large margin. Consequently, Beam Retrieval substantially improves the QA performance of downstream QA reader on all three datasets.

We highlight our contributions as follows:

- We propose Beam Retrieval, a general end-to-end retrieval framework for multi-hop QA. Beam Retrieval keeps multiple hypotheses of relevant passages at each step and is adapted to question with a variable hop.
- Our Beam Retrieval performs end-to-end training and inference with the same beam size, which optimizes an encoder and two classification heads by minimizing the combined loss across all hops, reducing the gap between training and reasoning.
- We evaluate our multi-hop QA system on three multi-hop QA datasets to validate the effectiveness of Beam Retrieval. Beam Retrieval achieves a nearly 50% improvement compared with baselines on challenging MuSiQue-Ans, and it also surpasses all previous retrievers on HotpotQA and 2WikiMultiHopQA. Providing high-quality context, Beam Retrieval helps our supervised reader achieve new state-of-the-art performance and substantially improves (up to 28.8 points) the QA performance of zero-shot GPT-3.5.

2 Related Work

Multi-Hop QA Multi-hop QA requires the model to reason over multiple and different scattered documents to give an answer of a complicated multi-hop question. Many high-quality multi-hop QA datasets have been introduced, such as HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020) , MuSiQue (Trivedi et al.,

2022) and so on. According to statistics in (Mohammadi et al., 2022), HotpotQA is the most commonly used dataset in this domain. However, HotpotQA is developed through direct crowdsourcing of 2-hop questions, without taking into account the complexity of composition. As a result, it has been demonstrated that this dataset can be largely solved without employing multi-hop reasoning, thereby enabling models to bypass the need for connected reasoning (Min et al., 2019; Chen and Durrett, 2019; Trivedi et al., 2020). 2WikiMultiHopQA shares a similar format with HotpotQA, with the exception of incorporating an additional entity recognition task and extending the scope to include 4-hop questions. They use a limited set of hand-authored compositional rules. In comparison to these datasets, MuSiQue-Ans presents a considerably more challenging task, as it is less susceptible to being solved through disconnected reasoning. This dataset comprises questions that require 2 to 4 hops, further emphasizing its complexity.

Auto-Regressive Language Generation In recent years, there has been an increasing interest in open-ended language generation performed by large language models (LLMs) (OpenAI, 2023; Touvron et al., 2023). Generally, LLMs adopt either an encoder-decoder (Chung et al., 2022) or a decoder-only (Brown et al., 2020) architecture, which generates text by predicting one word or token at a time, conditioning on the previously generated words or tokens. Besides the improved transformer architecture and massive unsupervised training data, better decoding methods have also played an important role. Decoding methods are applied to guide the language generation process and select the most appropriate output sequences based on certain criteria (Sutskever et al., 2014; Fan et al., 2018; Holtzman et al., 2020).

3 Preliminary

Basic Decoding Methods

Auto-regressive language models compute the conditional probability of each word in the target sequence based on the previous words. Let $z_t = f(y_1, ..., y_{t-1})$ represent the output of a decoder-only model given the sequence of tokens predicted so far, $(y_1, ..., y_{t-1})$, which for notational simplicity we write as $y_{<t}$. The output $z_t \in \mathbb{R}^V$ (where V is the cardinality of the enumerated vocabulary $\mathcal{V}$)

The probability distribution over the next possible token being word $w_i \in \mathcal{V}$ is the softmax:

$$P(y_t = w_i|y_{<t}) = \frac{exp(z_{t,i})}{\sum_{j=1}^{V} exp(z_{t,j})} \quad \forall i \in \{1, ..., V\} \tag{1}$$

Most decoding strategies strive to find the most likely overall sequence, i.e. pick a $\hat{\mathbf{y}}$ such that:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \prod_{t=1}^{N} P(y_t|y_{<t}, y_0) \tag{2}$$

where y_0 denotes the initial context word sequence. Since no sub-exponential algorithm is available for determining the optimal decoded sequence (Chen et al., 2018), alternative approximation methods, such as greedy search and beam search, are utilized.

Greedy Search Greedy search is the simplest decoding method. It selects the word with the highest probability as its next word:

$$\hat{y_t} = \arg\max_{y_t} P(y_t|y_{<t}) \tag{3}$$

Beam Search Beam search approximates finding the most likely sequence by performing breadth-first search over an expanded search space. At time step $t-1$ in decoding, the method keeps track of K partial hypotheses, denoted as $Y_{t-1}^k = \{y_1^k, y_2^k, ..., y_{t-1}^k\}$, $k \in [1, K]$. The next set of partial hypotheses is chosen by expanding every path from the existing set of K hypotheses, and then choosing the K with the highest scores. Log-likelihood of the partial sequence is used as the scoring function, which is denoted as $S(Y_{t-1}) = \log p(y_1, y_2, ..., y_{t-1})$. This iterative process continues until hypotheses reach the EOS token or exceed the predefined maximum length.

Problem Formulation

Given a k-hop question Q and a candidate set with n passages as $\mathcal{D} = \{p_1, p_2, ..., p_n\}$, multi-hop retrieval aims to produce a relevant passages chain $(\hat{p}_1, \hat{p}_2, ..., \hat{p}_k)$. Most existing work formulates it as a one-step or two-step sequence labeling task, classifying every passage $p_i \in \mathcal{D}$ as relevant or not. However, this method lacks generality and precision.

In contrast, we align multi-hop retrieval task with text decoding, proposing a more general retrieval framework with higher precision. Conceptually, a passage $p_i \in \mathcal{D}$ corresponds to a token $w_i \in \mathcal{V}$ and the question Q corresponds to a special start token "<s>". Similarly, we also denote the output of a multi-hop retriever as $\acute{z}_t = \acute{f}(Q, \hat{p}_1, ..., \hat{p}_{t-1})$, given the concatenated sequence of question and passages identified so far, $(Q, \hat{p}_1, ..., \hat{p}_{t-1})$, which we write as $\hat{p}_{<t}$ for short. The output $\acute{z}_t \in \mathbb{R}^n$.

We use an auto-encoder language model as an encoder to derive embeddings for the concatenated sequence $(Q, \hat{p}_1, ..., \hat{p}_{t-1}, \acute{z}_t)$. Subsequently, a fully connected layer is utilized to project the final dimension of the "[CLS]" representations of these embeddings into a 2-dimensional space, representing "irrelevant" and "relevant" respectively. The logit in "relevant" side serves as the score for the sequence. This scoring process is denoted by a function $S(\acute{z}_t|\hat{p}_{<t})$, and it is shown in Figure 2.

The probability distribution over the next possible relevant passage being $\mathrm{p}_i \in \mathcal{D}$ is the softmax:

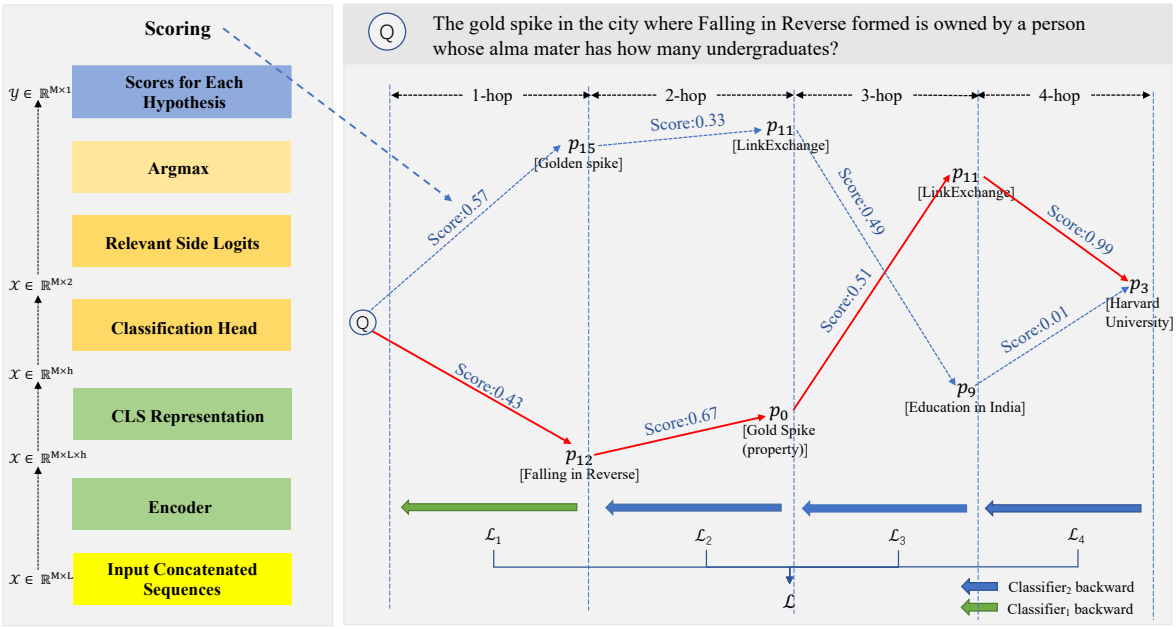

Figure 2: A visualization of Beam Retrieval with a beam size of 2 for the example in Figure 1. The left part shows how to obtain scores for each hypothesis, where M denotes the number of hypotheses at each hop, L denotes the max length of the hypotheses and h denotes the output dimension of the encoder. The right part shows how Beam Retrieval reasons and trains in an end-to-end way, where the red path refers to the ground-truth relevant passages.

$$\acute{P}(\hat{p}_t = \mathrm{p}_i|\hat{p}_{<t}) = \frac{S(\acute{z}_t|\hat{p}_{<t})}{\sum_{p\in\mathcal{D}\setminus\{\hat{p}_1,...,\hat{p}_{t-1}\}} S(p|\hat{p}_{<t})} \quad (4)$$
$$\forall \acute{z}_t \in \mathcal{D} \setminus \{\hat{p}_1, ..., \hat{p}_{t-1}\}$$

We should keep the uniqueness of each passage within the sequence, as there is no duplicated passages in the only one ground-truth relevant passage chain. This requirement differs from the text decoding process, where such uniqueness is not necessarily enforced.

4 Beam Retrieval

Beam Retrieval is designed to handle a k-hop multi-hop questions Q and accurately selects the most relevant passages, providing nearly noiseless context for downstream QA tasks. In this section, we clarify how Beam Retrieval infers and trains in an end-to-end way, which is illustrated in Figure 2.

Scoring

As described in Section 3, every hypothesis will be scored at each step during beam search. Beam Retrieval also employs a scoring function $S(\acute{z}_t|\hat{p}_{<t})$ as illustrated in Figure 2, which utilizes an encoder and two classification heads to obtain scores for each hypothesis of passages. At the first hop, for every passage $p_i \in \mathcal{D}$ we concatenate "[CLS] + Q + p_i + [SEP]" to the encoder and derive the encoded (Q, p_i) representations $\mathbf{H}^i = [\mathbf{h}^i_1, \mathbf{h}^i_2, ..., \mathbf{h}^i_{L_i}] \in \mathbb{R}^{L_i \times h}$, where L_i denotes the length of the concatenated sequence and h denotes the output dimension of the encoder. Then a classification head named "$classifier_1$" project every $\mathbf{H}^i$ into a 2-dimensional space, representing "irrelevant" and "relevant" respectively. We take the logit in "relevant" side as the score for the sequence (Q, p_i). At subsequent hop t, we concatenate "[CLS] + Q + $\hat{p}_1$ + ... + $\hat{p}_{t-1}$ + $\acute{z}_t$ + [SEP]" for every $\acute{z}_t \in \mathcal{D} \setminus \{\hat{p}_1, ..., \hat{p}_{t-1}\}$. We use the same encoder but another classification head named "$classifier_2$" to obtain the score of concatenate sequence $(Q, \hat{p}_1, ..., \hat{p}_{t-1}, \acute{z}_t)$ in the same way. The structures of "$classifier_1$" and "$classifier_2$" are totally same, the only difference is "$classifier_1$" handles a fixed n sequences while "$classifier_2$" deals with a variable number of sequences in an expanded search space.

End-to-End Inference

Compared with previous customized two-step retrieval methods (Wu et al., 2021; Li et al., 2022; Yin et al., 2022), Beam Retrieval employs the beam search paradigm to retrieve multiple relevant passages at each hop, discovering all the relevant passages of Q in an end-to-end way. Let B be the predefined beam size. Starting from the question Q, Beam Retrieval pairs it with n passages in $\mathcal{D}$ and scores these n concatenated sequences through the encoder and $classifier_1$, choosing the B passages with the highest scores as the first selected passages. At subsequent hop t, Beam Retrieval keeps track of B partial hypotheses, denoted as

$\mathcal{P}_{t-1}^b = \{\hat{p}_1^b, ..., \hat{p}_{t-1}^b\}$, $b \in [1, B]$. Then we concatenate $(Q, \mathcal{P}_{t-1}^b, \acute{z}_t)$ for every $\acute{z}_t \in \mathcal{D} \setminus \mathcal{P}_{t-1}^b$ as input concatenated sequences. In this way Beam Retrieval expands the search space, producing M hypotheses of passages, where M is slightly less than $B \times n$ as we should keep the uniqueness of each passage within the sequence. Then we score these hypotheses using the encoder and $classifier_2$, choosing the B hypotheses with the highest scores. This process continues until the last hop is reached, and we take the passage sequence with the highest score.

Beam Retrieval finishes the multi-hop retrieval task using a single forward pass, where it calls k times encoder, 1 time $classifier_1$ and $k-1$ times $classifier_2$. Additionally, as we can see in Figure 2, for methods that select only one passage at a time, choosing irrelevant passage at first stage could result in the failure of the entire multi-hop retrieval process. In conclusion, Beam Retrieval reduces the risk of missing hidden relevant passage sequences by keeping the most likely B hypotheses at each hop and eventually choosing the hypothesis that has the overall highest score.

Jointly Optimization

We jointly optimize the encoder, $classifier_1$ and $classifier_2$ across all hops in an end-to-end manner. Let $(\mathrm{p}_1, \mathrm{p}_2, ..., \mathrm{p}_k)$ be the ground truth relevant passages. At the first hop, the loss can be represented as:

$$\begin{aligned}\mathcal{L}_1 = -\sum_{p \in \mathcal{D}} l_{1,p} log S(p|Q) + \\ (1 - l_{1,p}) log(1 - S(p|Q))\end{aligned} \tag{5}$$

where $l_{1,p}$ is the label of p and $S(p|Q)$ is the score function described in Section 3. At subsequent hop t, the loss can be represented as:

$$\begin{aligned}\mathcal{L}_t = -\sum_{b=1}^{B} \sum_{p \in \mathcal{D} \setminus \mathcal{P}_{t-1}^b} l_{t,p} log S(p|\mathcal{P}_{t-1}^b, Q) \\ + (1 - l_{t,p}) log(1 - S(p|\mathcal{P}_{t-1}^b, Q))\end{aligned} \tag{6}$$

where $l_{t,p}$ is the label of p. It is important to note that not all datasets offer the ground-truth relevant passage for each hop. Consequently, for $t \in [1, k]$ we define $l_{t,p}$ under two scenarios: one with a provided order of relevant passages and another without a specified order. If the order of ground-truth relevant passages is given, $l_{t,p}$ is set as:

$$l_{t,p} = \begin{cases} 1 & \text{if } p = \mathrm{p}_t \\ 0 & \text{if } p \neq \mathrm{p}_t \end{cases} \tag{7}$$

Otherwise $l_{t,p}$ is set as:

$$l_{t,p} = \begin{cases} 1 & \text{if } p \in \{\mathrm{p}_1, \mathrm{p}_2, ..., \mathrm{p}_k\} \\ 0 & \text{if } p \notin \{\mathrm{p}_1, \mathrm{p}_2, ..., \mathrm{p}_k\} \end{cases} \tag{8}$$

Hence the overall training loss of Beam Retrieval can be:

$$\mathcal{L} = \sum_{i=1}^{k} \mathcal{L}_i \tag{9}$$

5 Experimental Setup

Datasets

We focus on the retrieval part of Multi-hop QA and conduct experiments on three benchmark datasets MuSiQue-Ans (Trivedi et al., 2022), distractor-setting of HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020). MuSiQue-Ans, HotpotQA, 2WikiMultihopQA have 20K, 90K and 167K training instances, respectively. MuSiQue-Ans requires model to answer the complicated multi-hop questions, while HotpotQA and 2WikiMultihopQA additionally require model to provide corresponding supporting sentences. In the setting of Beam Retrieval augmented LLM, we evaluate our method on the partial part of three multi-hop datasets, where we use the 500 questions for each dataset sampled by (Trivedi et al., 2023).

HotpotQA and 2WikiMultihopQA share a similar format and have 2-hop and 2,4-hop questions respectively. In both datasets, each question is accompanied by 10 passages, where only a few of them (2 in HotpotQA and 2 or 4 in 2WikiMultihopQA) are relevant to the question. Futhermore, 2WikiMultihopQA has entity-relation tuples support, but we do not use this annotation in our training or evaluation. Main experiments are conducted on MuSiQue-Ans, which has 2,3,4-hop questions and is more challenging, as it requires explicit connected reasoning (Trivedi et al., 2022).

Models

Beam Retrieval

Beam Retrieval selects all the relevant passages in an end-to-end way. We employ the base and the large version of DeBERTa (He et al., 2021) as our encoder. We use a single RTX4090 GPU and set the number of epochs to 16 and the batch size to 1 (here batch size means the number of examples taken from dataset, and the actual batch size is the hypotheses number M). Owing to our multiple calls of encoder during training, we set gradient checkpointing to True, otherwise it requires huge amount of memory. We use BERT-Adam with learning rate of 2e-5 for the optimization and set the max position embeddings to 512. Considering the long concatenated sequences, we adopt a truncation method. If the total length exceeds the max length, we calculate the average length of each passage and truncate the extra part. In addition, for effective training, we will stop the loss calculation if all the hypotheses do not hit the ground truth. To enhance the robustness of model, we shuffle the inner order of the concatenated passages within the hypothesis.

Downstream Reader

We implement a downstream reader to receive the retrieved relevant passages as the context C, and we concatenate input "[CLS] + Q + [SEP] + C + [SEP]" to feed our reader. Specifically, we conduct experiments with two types of reader: supervised setting and zero-shot LLM setting.

(i) **Supervised Reader** For MuSiQue-Ans dataset, we train a reading comprehension model following BertForQuestionAnswering (Devlin et al., 2019; Wolf et al., 2020). For HotpotQA and 2WikiMultihopQA, we train a multi-task reader which extracts the answer and the supporting facts of the question, following FE2H (Li et al., 2022) and C2FM (Yin et al., 2022), where you can refer to Appendix A for details. For supervised setting, we employ the large version of DeBERTa for MuSiQue and 2WikiMultihopQA and the xxlarge version of DeBERTa for HotpotQA. We use a single RTX4090 GPU to train the large version reader and a single A100 to train the xxlarge version reader. We set the number of epochs to 12 and the batch size to 4. We use BERTAdam with learning rate of 5e-6 for the optimization and set the max position embeddings to 1024. To enhance the robustness of model, we shuffle the inner order of the concatenated passages within the context.

(ii)**Zero-Shot LLM** In addition to the supervised reader above, we also incorporate a zero-shot LLM as the downstream reader to benchmark the QA performance of Beam Retrieval augmented LLM. For zero-shot LLM setting, we use *gpt-3.5-turbo* provided from API of OpenAI[2]. We use the template described in Appendix B to obtain the answers directly.

Evaluation Metrics

Generally, we use Exact Match (EM) and F1 score to evaluate the retrieval performance. Retrieval EM means whether the passage-level prediction is totally same as the ground truth, while retrieval F1 is the harmonic mean of precision and recall, and both of them are irrespective of the inner order between relevant passages. In retrieve-and-read setting, retrieval EM is particularly critical, as missing relevant passages can significantly impact the performance of downstream reader.

For MuSiQue-Ans, we report the standard F1 based metrics for answer (**An**) and support passages identification (**Sp**). Actually, **Sp** F1 in MuSiQue-Ans is equivalent to retrieval F1. For HotpotQA and 2WikiMultihopQA, we report the EM and F1 metrics for answer prediction task (**Ans**) and supporting facts prediction task (**Sup**). In Beam Retrieval augmented LLM setting, we report the answer F1.

6 Results

Appropriate Beam Size We first explore the influence of different beam size on MuSiQue-Ans dataset, as shown in Table 1, where the encoder is base version. Interestingly, Beam Retrieval performs well even with a beam size of 1, and a beam size of 2 yields the most benefits, which is consistent with (Sutskever et al., 2014). It is worth mentioning that in our experimental setting, the candidate set size n ranges from 10 to 20. As the beam size expands, both the necessary training memory and training duration increase rapidly. For instance, a beam size of 4 demands approximately double the memory and triple the training duration in comparison to a beam size of 1. Due to these considerations, we do not conduct experiments with a beam size larger than 4. In conclusion, we employ beam sizes of 1 and 2 for Beam Retrieval in our subsequent experiments.

beam size	**EM**	**F1**	**Mem** (%)	**Speed** (%)
1	74.18	87.46	100%	100%
2	**75.47**	**88.27**	119%	58%
3	74.56	87.84	150%	42%
4	74.43	87.65	194%	36%

Table 1: Influence of different beam size among retrieval performance, training memory required and training speed. A beam of size 2 offers the optimal balance between retrieval performance and training costs.

Methods	**Retrieval**	
	EM	**F1**
MuSiQue-Ans		
EE (Trivedi et al., 2022)	21.47	67.61
SA (Trivedi et al., 2022)	30.37	72.30
Ex(EE) (Trivedi et al., 2022)	48.78	77.79
Ex(SA) (Trivedi et al., 2022)	53.50	79.24
Beam Retrieval, beam size 1	77.37	89.77
Beam Retrieval, beam size 2	**79.31**	**90.51**
HotpotQA		
SAE (Tu et al., 2020)	91.98	95.76
SA Selector* (Trivedi et al., 2022)	93.06	96.43
S2G (Wu et al., 2021)	95.77	97.82
FE2H (Li et al., 2022)	96.32	98.02
C2FM (Yin et al., 2022)	96.85	98.32
Beam Retrieval, beam size 1	97.29	98.55
Beam Retrieval, beam size 2	**97.52**	**98.68**
2WikiMultihopQA		
SA Selector* (Trivedi et al., 2022)	98.25	99.13
Beam Retrieval, beam size 1	**99.93**	**99.96**

Table 2: Retrieval performance on the development set of MuSiQue-Ans, HotpotQA, 2WikiMultihopQA in comparison with previous work. SA Selector* indicates that we reproduce SA Selector by training it on the full HotpotQA and 2WikiMultihopQA. Beam Retrieval surpasses all previous retrievers by a large margin.

Beam Retrieval Performance We compare our Beam Retrieval with previous retrievers on three multi-hop datasets, as shown in Table 2. Beam Retrieval achieves new SOTA performance across all datasets, significantly

[2] https://openai.com/api/

Methods	Answer		Supporting	
	EM	F1	EM	F1
HotpotQA dev set				
DFGN (Qiu et al., 2019)	55.42	69.23	-	-
HGN (Fang et al., 2020)	-	82.22	-	88.56
SAE (Tu et al., 2020)	67.70	80.75	63.30	87.38
S2G (Wu et al., 2021)	70.8	-	65.7	-
FE2H (Li et al., 2022)	71.72	84.61	66.12	89.65
C2FM (Yin et al., 2022)	71.90	84.65	66.75	90.08
Beam Retrieval, beam size 1	72.09	85.19	67.06	90.29
Beam Retrieval, beam size 2	**72.25**	**85.30**	**67.25**	**90.43**
2WikiHotpotQA test set				
CRERC (Fu et al., 2021)	69.58	72.33	82.86	90.68
NA-Reviewer (Fu et al., 2022)	76.73	81.91	89.61	94.31
BigBird-base model (Ho et al., 2023)	74.05	79.68	77.14	92.13
Beam Retrieval, beam size 1	**88.47**	**90.87**	**95.87**	**98.15**

Table 3: Overall performance on the development set of HotpotQA and the test set of 2WikiMultihopQA in comparison with previous work. '-': score is unavailable. Beam Retrieval achieves SOTA in both datasets

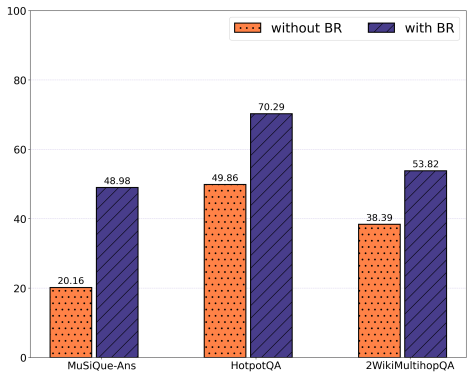

Figure 3: Answer F1 for *gpt-3.5-turbo* under two conditions on three multi-hop datasets. Beam Retrieval substantially improves the zero-shot QA performance of LLM, which is even comparable to some supervised methods.

outperforming existing methods even when using a beam size of 1, and notably attaining a nearly 50% EM improvement (from 53.50 to 77.37[3]) on challenging MuSiQue-Ans. This result highlights the effectiveness of our proposed approach in handling more complex situations. As demonstrated in Table 1, employing a beam size of 2 consistently improves performance on both MuSiQue-Ans and HotpotQA datasets, validating the benefits of an expanded search space. As the high-performance retrievers in HotpotQA are customized for two-hop issues, we do not reproduce them for the other two datasets. A large version encoder is employed for all datasets except 2WikiMultihopQA, where a base version encoder achieves a remarkable 99.9% retrieval precision. Therefore we do not conduct further experiments with larger beam sizes or encoders for this dataset.

[3]MuSiQue-Ans leaderboard: https://leaderboard.allenai.org/musique_ans/submissions/public

Methods	MuSiQue-Ans	
	An	Sp
EE (Trivedi et al., 2022)	40.7	69.4
SA (Trivedi et al., 2022)	52.3	75.2
Ex(EE) (Trivedi et al., 2022)	46.4	78.1
Ex(SA) (Trivedi et al., 2022)	49.0	80.6
Beam Retrieval, beam size 1	66.9	90.0
Beam Retrieval, beam size 2	**69.2**	**91.4**

Table 4: Overall performance on the test set of MuSiQue-Ans. Beam Retrieval achieves a new state-of-the-art.

Downstream QA Performance Table 4 and Table 3 compare multi-hop QA performance between Beam Retrieval augmented supervised reader (hereinafter referred to as Beam Retrieval) and other strong multi-hop systems across three datasets. Thanks to the retrieved high-quality context, Beam Retrieval with beam size of 2 achieves new SOTA on all three datasets. Specifically, on MuSiQue-Ans our Sp performance (91.4) is comparable to Human Score (93.9) reported in (Trivedi et al., 2022). To evaluate the degree of enhancement Beam Retrieval can provide, we compare the QA performance of zero-shot GPT-3.5 under two conditions: one using all candidate passages (referred to as "without BR"), and the other incorporating relevant passages retrieved by Beam Retrieval with beam size of 2 (referred to as "with BR"), which is depicted in Figure 3. Beam Retrieval significantly boosts the zero-shot QA performance of LLM, yielding a 28.8-point improvement on the challenging MuSiQue-Ans, a 20.4-point improvement on HotpotQA, and a 15.5-point improvement on 2WikiMultihopQA.

Ablation Study To understand the strong performance of Beam Retrieval, we perform an ablation study by employing inconsistent beam sizes between training

Methods	Retrieval	
	EM	F1
Beam Retrieval$_{1,1}$	74.18	87.46
Beam Retrieval$_{2,2}$	**75.47**	**88.27**
Beam Retrieval$_{3,3}$	74.56	87.84
w/o Consistent Beam Size		
Beam Retrieval$_{3,2}$	74.31	87.84
Beam Retrieval$_{3,1}$	74.06	87.67
Beam Retrieval$_{2,1}$	75.13	88.17
w/o 2 Classfication Heads		
BR$_{1,1}$ with 4 Classfication Heads	72.16	87.04
BR$_{1,1}$ with 1 Classfication Head	73.11	87.32

Table 5: Ablation study results on MuSiQue-Ans dataset. The subscript $_{x,y}$ indicates training with beam size x and reasoning with beam size y.

and reasoning and using different numbers of classification heads, as illustrated in Table 5. Performance declines when the training beam size differs from the reasoning beam size, and it drops more sharply as the gap between training and reasoning widens. We do not investigate situations where the reasoning beam size exceeds the training beam size, as it is evident that model cannot perform hard reasoning after easy training. We also vary the number of classification heads to verify if two heads are the optimal setting. First we use 4 classification heads as there are up to 4-hop questions and we arrange one head for one hop, however it results in a 2-point decrease in EM. Then we employ a unified classification head, which also leads to a one-point performance drop. These results confirm that using one head for the first hop and another head for subsequent hops is the best configuration.

7 Conclusion

We present Beam Retrieval, a general end-to-end retrieval framework for multi-hop QA. This approach maintains multiple partial hypotheses of relevant passages at each step, expanding the search space and reducing the risk of missing relevant passages. Experimental results on three benchmark datasets prove the effectiveness of Beam Retrieval and demonstrate it could substantially improve the QA performance of downstream reader. In general, Beam Retrieval establishes a strong baseline for complex multi-hop QA, where we hope that future work could explore more advanced solutions.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *Proceedings of NAACL-HLT*, pages 4026–4032.

Yining Chen, Sorcha Gilroy, Andreas Maletti, Jonathan May, and Kevin Knight. 2018. Recurrent neural networks as weighted language recognizers. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2261–2271, New Orleans, Louisiana. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8823–8838, Online. Association for Computational Linguistics.

Ruiliu Fu, Han Wang, Xuejun Zhang, Jun Zhou, and Yonghong Yan. 2021. Decomposing complex questions makes multi-hop QA easier and more interpretable. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 169–180, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ruiliu Fu, Han Wang, Jun Zhou, and Xuejun Zhang. 2022. Na-reviewer: Reviewing the context to improve the error accumulation issue for multi-hop qa. *Electronics Letters*, 58(6):237–239.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2023. Analyzing the effectiveness of the underlying reasoning tasks in multi-hop question answering. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1163–1180, Dubrovnik, Croatia. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. *CoRR*, abs/2011.01060.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xin-Yi Li, Wei-Jun Lei, and Yu-Bin Yang. 2022. From easy to hard: Two-stage selector and reader for multi-hop question answering. *CoRR*, abs/2205.11729.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4249–4257, Florence, Italy. Association for Computational Linguistics.

Azade Mohammadi, Reza Ramezani, and Ahmad Baraani. 2022. A comprehensive survey on multi-hop machine reading comprehension datasets and metrics. *CoRR*, abs/2212.04070.

OpenAI. 2023. Gpt-4 technical report.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150, Florence, Italy. Association for Computational Linguistics.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2020. Is multihop QA in DiRe condition? measuring and reducing disconnected reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8846–8863, Online. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9073–9080. AAAI Press.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing.

In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Bohong Wu, Zhuosheng Zhang, and Hai Zhao. 2021. Graph-free multi-hop reading comprehension: A select-to-guide strategy. *CoRR*, abs/2107.11823.

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*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zhangyue Yin, Yuxin Wang, Yiguang Wu, Hang Yan, Xiannian Hu, Xinyu Zhang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2022. Rethinking label smoothing on multi-hop question answering. *CoRR*, abs/2212.09512.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774.

A Multi-Task Supervised Reader

After receiving the relevant passages $(\hat{p}_1, \hat{p}_2, ..., \hat{p}_k)$ from the retriever, our reader is expected to complete both the answer prediction task and the supporting facts prediction task. Following SAE (Tu et al., 2020) and C2FM (Yin et al., 2022), we also implement a multi-task model to extract the answer and the supporting facts, jointly training the answer prediction and supporting sentence classification in a multi-task learning way.

We define three types of tasks: supporting facts prediction, answer type prediction, and answer span prediction. Following C2FM, we incorporate a special placeholder token "<d>" before each document title and a token "<e>" before each sentence to provide additional information and guide the model to predict at the sentence level.

We concatenate the question and the retrieved passage chain $(\hat{p}_1, \hat{p}_2, ..., \hat{p}_k)$ as "[CLS] + question + [SEP] + $\hat{p}_1$ + $\hat{p}_2$ + ... + $\hat{p}_k$ + [SEP]". We denote the BERT-like PLM output as $H = [h1, ..., h_L] \in \mathbb{R}^{L \times d}$ where L is the length of the input sequence and d is the hidden dimension of the backbone model. For answer type prediction, we perform a 3-class ("Yes", "No" and "Span") classification, with the corresponding loss item denoted as $\mathcal{L}_{type}$. To extract the supporting facts prediction, we apply a linear layer on H to classify each sentence as either a supporting facts sentence or not (using the sentence token "<e>"), with its corresponding loss item denoted as $\mathcal{L}_{sf}$. Similarly, we employ another linear layer to project H and identify the start and end positions of the answer, denoting the start position loss and the end position loss as $\mathcal{L}_{start}$ and $\mathcal{L}_{end}$, respectively, as introduced in BERT (Devlin et al., 2019). Finally, the total answer span loss $\mathcal{L}_{ans}$ is described using the following formulas.

$$\mathcal{L}_{ans} = \lambda_1(\mathcal{L}_{start} + \mathcal{L}_{end}) \tag{10}$$

where λ_1 is 0.5 in our setting. Formally, the total loss $\mathcal{L}_{qa}$ can be jointly calculated as:

$$\mathcal{L}_{qa} = \lambda_2\mathcal{L}_{type} + \lambda_3\mathcal{L}_{sf} + \lambda_4\mathcal{L}_{ans} \tag{11}$$

where λ_2 is 0.2 and λ_3, λ_4 are 1 in our setting. Here each loss function is the cross-entropy loss.

B Zero-Shot GPT-3.5 Prompt

```
You are a qa test machine, you need to answer the
[Question] from the given [Context], you only need to
come out the correct answer without any other words.
[Question]:
[Context]:
```