Subgraph Retrieval Enhanced Model for Multi-hop Knowledge Base Question Answering

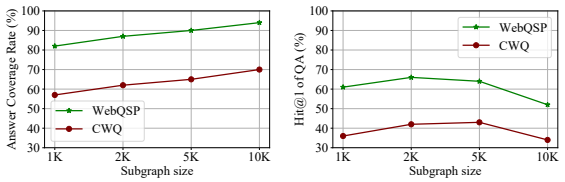
Jing Zhang[1], Xiaokang Zhang[1], Jifan Yu[2], Jian Tang[3],
Jie Tang[2], Cuiping Li[1]*, Hong Chen[1]
[1]School of Information, Renmin University of China, Beijing, China,
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]Mila - Quebec AI Institute.
{zhang-jing,zhang2718,licuiping,chong}@ruc.edu.cn,
yujf18@mails.tsinghua.edu.cn, jian.tang@hec.ca,
jietang@tsinghua.edu.cn

Abstract

Recent works on knowledge base question answering (KBQA) retrieve subgraphs for easier reasoning. The desired subgraph is crucial as a small one may exclude the answer but a large one might introduce more noises. However, the existing retrieval is either heuristic or interwoven with the reasoning, causing reasoning on the partial subgraphs, which increases the reasoning bias when the intermediate supervision is missing. This paper proposes a trainable subgraph retriever (SR) decoupled from the subsequent reasoning process, which enables a plug-and-play framework to enhance any subgraph-oriented KBQA model. Extensive experiments demonstrate SR achieves significantly better retrieval and QA performance than existing retrieval methods. Via weakly supervised pre-training as well as the end-to-end fine-tuning, SR achieves new state-of-the-art performance when combined with NSM (He et al., 2021), a subgraph-oriented reasoner, for embedding-based KBQA methods. Codes and datasets are available online[1].

1 Introduction

Knowledge Base Question Answering (KBQA) (Zhang et al., 2021) aims to seek answers to factoid questions from structured KBs such as Freebase, Wikidata, and DBPedia. KBQA has attracted a lot of attention, as the logically organized entities and their relations are beneficial for inferring the answer. Semantic parsing-based (SP-based) methods (Das et al., 2021; Lan and Jiang, 2020; Sun et al., 2020) and embedding-based methods (He et al., 2021; Sun et al., 2018, 2019) are two mainstream methods for addressing KBQA. The former ones heavily rely on the expensive annotation of the intermediate logic form such as SPARQL. Instead of parsing the questions, the later ones directly represent and rank entities based on their relevance to input questions. Among them, the models which first retrieve a question-relevant subgraph and then perform reasoning on it (He et al., 2021; Sun et al., 2018, 2019) reduce the reasoning space, showing superiority compared with reasoning on the whole KB (Chen et al., 2019a; Saxena et al., 2020; Xu et al., 2019) (Cf. Table 2 for empirical proof).



(a) Answer Coverage Rate (b) Hits@1 of QA

Figure 1: The impact of subgraph size on (a) answer coverage rate and (b) QA performance (Hits@1) of NSM (He et al., 2021) on WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018).

Subgraph retrieval is crucial to the overall QA performance, as a small subgraph is highly likely to exclude the answer but a large one might introduce noises that affect the QA performance. Figure 1(a) presents the answer coverage rates of the subgraphs with different sizes on two widely-used KBQA datasets, WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018). We extract the full multi-hop topic-centric subgraph and control the graph size by the personalized pagerank (PPR) (Haveliwala, 2003) scores of entities. We also present the QA performance (Hits@1) of NSM (He et al., 2021), a state-of-the-art embedding-based model, under the same sizes of the subgraphs in Figure 1(b). It is observed that although larger subgraphs are more likely to cover the answer, the QA performance drops dramatically when the subgraph includes more than 5,000 nodes. Moreover, it is inefficient to extract such a full multi-hop subgraph for online QA. The results show that such heuris-

*Corresponding author.

[1]https://github.com/RUCKBReasoning/SubgraphRetrievalKBQA

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 5773 - 5784

tic retrieval is far from optimal. To improve the retrieval performance, PullNet (Sun et al., 2019) proposes a trainable retriever, but the retrieving and the reasoning processes are intertwined. At each step, a LSTM-based retriever selects new relations relevant to the question, and a GNN-based reasoner determines which tail entities of the new relations should be expanded into the subgraph. As a result, the inference as well as the training of the reasoner needs to be performed on the intermediate partial subgraph. Since the intermediate supervision is usually unobserved, reasoning on partial subgraphs increases the bias which will eventually affect the answer reasoning on the final entire subgraph.

This paper proposes a subgraph retrieval enhanced model for KBQA, which devises a trainable subgraph retriever (SR) decoupled from the subsequent reasoner. SR is devised as an efficient dual-encoder that can expand paths to induce the subgraph and can stop the expansion automatically. After that, any subgraph-oriented reasoner such as GRAFT-Net (Sun et al., 2018) or NSM (He et al., 2021) can be used to delicately deduce the answers from the subgraph. Such separable retrieval and reasoning ensure the reasoning only on the final entire instead of the intermediate partial subgraphs, which enables a plug-and-play framework to enhance any subgraph-oriented reasoner.

We systematically investigate the advantages of various training strategies for SR, including weakly supervised/unsupervised pre-training and end-to-end fine-tuning with the reasoner. Instead of the ground truth paths, we extract the shortest paths from a topic entity in the question to an answer as the weak supervision signals for pre-training. When the QA pairs themselves are also scarce, we construct pseudo (question, answer, path) labels for unsupervised pre-training. To further teach the retriever by the final QA performance, we enable the end-to-end fine-tuning, which injects the likelihood of the answer conditioned on a subgraph as the feedback from the reasoner into the prior distribution of the subgraph to update the retriever.

We conduct extensive experiments on WebQSP and CWQ. The results reveal four major advantages: (1) SR, combined with existing subgraph-oriented reasoners, achieves several gains (+0.4-9.7% Hits@1 and 1.3-8.7% F1) over the same reasoner that is performed with other retrieval methods. Moreover, SR together with NSM creates new state-of-the-art results for embedding-based KBQA models. (2) With the same coverage rate of the answers, SR can result in much smaller subgraphs that can deduce more accurate answers. (3) The unsupervised pre-training can improve about 20% Hits@1 when none of the weak supervision data is provided. (4) The end-to-end fine-tuning can enhance the performance of the retriever as well as the reasoner.

Contributions. (1) We propose a trainable SR decoupled from the subsequent reasoner to enable a plug-and-play framework for enhancing any subgraph-oriented reasoner. (2) We devise SR by a simple yet effective dual-encoder, which achieves significantly better retrieval and QA results than the existing retrieval methods. (3) NSM equipped with SR, via weakly supervised pre-training and end-to-end fine-tuning, achieves new SOTA performance for embedding-based KBQA methods.

2 Related Work

KBQA solutions can be categorized into SP-based and embedding-based methods. SP-based methods (Bao et al., 2016; Berant and Liang, 2014; Das et al., 2021; Lan and Jiang, 2020; Liang et al., 2017; Qiu et al., 2020b; Sun et al., 2020) parse a question into a logic form that can be executed against the KB. These methods need to annotate expensive logic forms as supervision or are limited to narrow domains with a few logical predicates. Embedding-based methods embed entities and rank them based on their relevance to the question, where the entities are extracted from the whole KB (Miller et al., 2016; Saxena et al., 2020) or restricted in a subgraph (Chen et al., 2019a; He et al., 2021; Sun et al., 2018; Zhang et al., 2018). They are more fault-tolerant but the whole KB or the ad-hoc retrieved subgraph includes many irrelevant entities. Some works such as PullNet (Sun et al., 2019), SRN (Qiu et al., 2020a), IRN (Zhou et al., 2018), and UHop (Chen et al., 2019b) enhance the retrieval by training the retriever, but the retrieving and the reasoning are intertwined, causing the reasoning on partially retrieved subgraphs. Because of such coupled design, the reasoner in SRN, IRN, and UHop is degenerated into a simple MLP. On the contrary, thanks to the decoupled design, the reasoner can be complicated to support more complex reasoning. Other works propose more complicated reasoner for supporting the numerical reasoning in KBQA (Feng et al., 2021).

Open-domain QA (OpenQA) aims to answer questions based on a large number of documents. Most of the OpenQA models also consist of a retriever to identify the relevant documents and a reasoner to extract the answers from the documents. The retriever is devised as a sparse term-based method such as BM25 (Robertson and Zaragoza, 2009) or a trainable dense passage retrieval method (Karpukhin et al., 2020; Sachan et al., 2021), and the reasoner deals with each document individually (Guu et al., 2020) or fuses all the documents together (Izacard and Grave, 2021). Different from the documents in openQA, the subgraphs in KBQA can be only obtained by multi-hop retrieval and the reasoner should deal with the entire subgraph instead of each individual relation to find the answer. Although some openQA research proposes multi-hop document retrieval (Asai et al., 2020), the focus is the matching of the documents rather than the relations to the questions in KBQA. Thus the concrete solution for KBQA should be different from openQA.

3 Problem Definition

A **knowledge base** (KB) G organizes the factual information as a set of triples, *i.e.*, $G = \{(e, r, e')|e, e' \in E, r \in R\}$, where E and R denote the entity set and the relation set respectively. Given a factoid question q, KBQA is to figure out the answers A_q to the question q from the entity set E of G. The entities mentioned in q are topic entities denoted by $E_q = \{e_q\}$, which are assumed to be given. This paper considers the complex questions where the answer entities are multi-hops away from the topic entities, called multi-hop KBQA.

Probabilistic Formalization of KBQA. Given a question q and one of its answers $a \in A_q$, we formalize the KBQA problem as maximizing the probability distribution $p(a|G, q)$. Instead of directly reasoning on G, we retrieve a subgraph $\mathcal{G} \subseteq G$ and infer a on $\mathcal{G}$. Since $\mathcal{G}$ is unknown, we treat it as a latent variable and rewrite $p(a|G, q)$ as:

$$p(a|G, q) = \sum_{\mathcal{G}} p_\phi(a|q, \mathcal{G}) p_\theta(\mathcal{G}|q). \quad (1)$$

In the above equation, the target distribution $p(a|G, q)$ is jointly modeled by a subgraph retriever $p_\theta(\mathcal{G}|q)$ and an answer reasoner $p_\phi(a|q, \mathcal{G})$. The subgraph retriever p_θ defines a prior distribution over a latent subgraph $\mathcal{G}$ conditioned on a question q, while the answer reasoner p_ϕ predicts the likelihood of the answer a given $\mathcal{G}$ and q. The goal is to find the optimal parameters θ and ϕ that can maximize the log-likelihood of training data, *i.e.*,

$$\mathcal{L}(\theta, \phi) = \max_{\theta,\phi} \sum_{(q,a,G)\in\mathcal{D}} \log \sum_{\mathcal{G}} p_\phi(a|q, \mathcal{G}) p_\theta(\mathcal{G}|q), \quad (2)$$

where $\mathcal{D}$ is the whole training data. Thanks to this formulation, the retriever can be decoupled from the reasoner by firstly training the retriever p_θ and then the reasoner p_ϕ on the subgraphs sampled by the retriever. Via drawing a sample $\mathcal{G}$ (Sachan et al., 2021), we can approximate Eq. (2) as:

$$\mathcal{L}(\theta, \phi) = \max_{\theta,\phi} \sum_{(q,a,\mathcal{G})\in\mathcal{D}} \log p_\phi(a|q, \mathcal{G}) + \log p_\theta(\mathcal{G}|q), \quad (3)$$

where the first and the second term can be optimized for the reasoner and the retriever respectively. The concrete reasoner can be instantiated by any subgraph-oriented KBQA model such as the GNN-based GRAT-Net (Sun et al., 2018) and NSM (He et al., 2021).

4 Subgraph Retriever (SR)

The retriever needs to calculate $p_\theta(\mathcal{G}|q)$ for any $\mathcal{G}$, which is intractable as the latent variable $\mathcal{G}$ is combinatorial in nature. To avoid enumerating $\mathcal{G}$, we propose to expand top-K paths relevant to q from the topic entities and then induce the subgraph following these paths.

4.1 Expanding Paths

Path expanding starts from a topic entity and follows a sequential decision process. Here a path is defined as a sequence of relations $(r_1, \cdots, r_{|p|})$, since a question usually implies the intermediate relations excluding the entities. Suppose a partial path $p^{(t)} = (r_1, \cdots, r_t)$ has been retrieved at time t, a tree can be induced from $p^{(t)}$ by filling in the intermediate entities along the path, *i.e.*, $T^{(t)} = (e_q, r_1, E_1, \cdots, r_t, E_t)$. Each E_t is an entity set as a head entity and a relation can usually derive multiple tail entities. Then we select the next relation from the union of the neighboring relations of E_t. The relevance of each relation r to the question q is measured by the dot product between their embeddings, *i.e.*,

$$s(q, r) = f(q)^\top h(r), \quad (4)$$

where both f and h are instantiated by RoBERTa (Liu et al., 2019). Specifically, we input

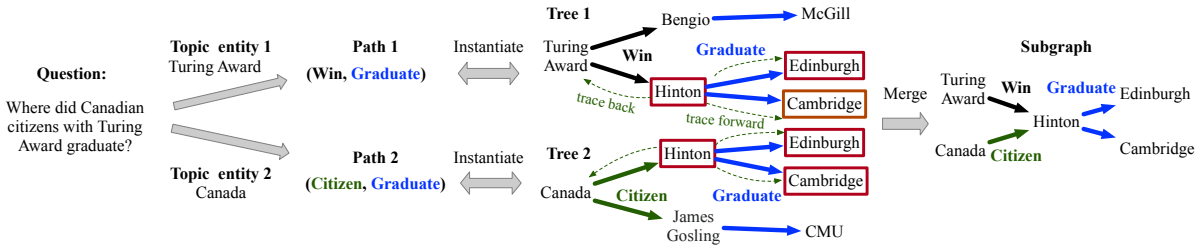


Figure 2: Illustration of the subgraph retrieving process. We expand a path from each topic entity as well as induce a corresponding tree, and then merge the trees from different topic entities to form a unified subgraph.

the question or the name of r into RoBERTa and take its [CLS] token as the output embedding. According to the assumption (Chen et al., 2019b; He et al., 2021; Qiu et al., 2020a; Zhou et al., 2018) that expanding relations at different time steps should attend to specific parts of a query, we update the embedding of the question by simply concatenating the original question with the historical expanded relations in $p^{(t)}$ as the input of RoBERTa, *i.e.*,

$$f(q^{(t)}) = \text{RoBERTa}([q; r_1; \cdots; r_t]), \quad (5)$$

Thus $s(q, r)$ is changed to $s(q^{(t)}, r) = f(q^{(t)})^\top h(r)$. Then the probability of a relation r being expanded can be formalized as:

$$p(r|q^{(t)}) = \frac{1}{1 + \exp\left(s(q^{(t)}, \text{END}) - s(q^{(t)}, r)\right)}, \quad (6)$$

where END is a virtual relation named as "END". The score $s(q^{(t)}, \text{END})$ represents the threshold of the relevance score. $p(r|q^{(t)})$ is larger than 0.5 if $s(q^{(t)}, r) > s(q^{(t)}, \text{END})$ and is no larger than 0.5 otherwise. We select the top-1 relation with $p(r|q^{(t)}) > 0.5$. The expansion is stopped if none of the probabilities of the relations is larger than 0.5. Finally, the probability of a path given the question can be computed as the joint distribution of all the relations in the path, *i.e.*,

$$p_\theta(p|q) = \prod_{t=1}^{|p|} p(r_t|q^{(t)}). \quad (7)$$

where $|p|$ denotes the number of relations in p, $t = 1$ indicates the selection at the topic entity and $t = |p|$ denotes the last none-stop relation selection. Since the top-1 relevant path cannot be guaranteed to be right, we perform a top-K beam search at each time to get K paths. From each topic entity, we obtain K paths which result in nK paths in total by n topic entities. nK paths correspond to nK instantiated trees.

4.2 Inducing Subgraph

We take the union of top-K trees from one topic entity into a single subgraph, and then merge the same entities from different subgraphs to induce the final subgraph. This can reduce the subgraph size, *i.e.*, the answer reasoning space, as the subgraphs from different topic entities can be viewed as the constraints of each other. Specifically, from the n subgraphs of the n topic entities, we find the same entities and merge them. From these merged entities, we trace back in each subgraph to the root (*i.e.*, a topic entity) and trace forward to the leaves. Then we only keep the entities and relations along the tracing paths of all the trees to form the final subgraph. For example in Figure 2, given a question "Where did Canadian citizens with Turing Award graduate?" with two topic entities "Turing Award" and "Canada"[2], we can explain it by the two expanded paths (Win, Graduate) and (Citizen, Graduate) and merge the trees induced by them to form a unified subgraph. Only the top-1 path is presented in the figure for a clear illustration.

5 Training Strategies

In this section, we discuss the pre-training and the end-to-end fine-tuning strategies to train the retriever. Figure 3 illustrates the whole framework and the training procedure.

5.1 Weakly Supervised Pre-Training

Since the ground truth subgraphs are not easy to be obtained, we resort to the weak supervision signals constructed from the (q, a) pairs. Specifically, from each topic entity of a question, we retrieve all the shortest paths to each answer as the supervision signals, as paths are easier to be obtained than graphs. Since maximizing the log-

[2]Some work views "Canada" as a constraint, which is not easy to be distinguished with the topic entity "Turing Award". Thus this paper treats both of them as topic entities.

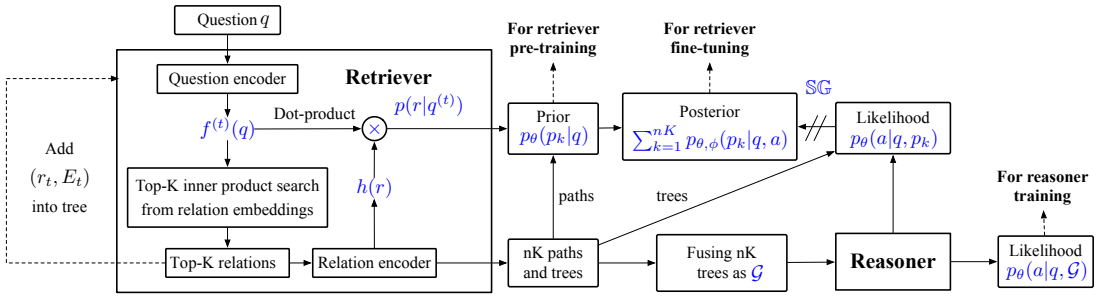


Figure 3: Overview of SR and its training strategies. Given a question, SR generates nK paths via iteratively expanding the relations. We pre-train the retriever based on the prior of each path and train the reasoner based on the likelihood of the subgraph fused from the nK trees. For end-to-end training, the retriever is fine-tuned on the posterior of each path that consists of the prior and the likelihood of it. $\mathbb{SG}$ is the stop-gradient operation.

likelihood of a path equals to $\sum_{t=1}^{|p|} \log p(r_t|q^{(t)})$ according to Eq. (7), we can maximize the probabilities of all the intermediate relations in a path. To achieve the goal, we decompose a path $p = (r_1, \cdots, r_{|p|})$ into $|p|+1$ (question, relation) instances, including $([q], r_1)$, $([q; r_1], r_2)$, ..., $([q; r_1; r_2; \cdots; r_{|p|-1}], r_{|p|})$, and an additional END instance $([q; r_1; r_2; \cdots; r_{|p|}], \text{END})$, and optimize the probability of each instance. We replace the observed relation at each time step with other sampled relations as the negative instances to optimize the probability of the observed ones.

5.2 Unsupervised Pre-Training

When the (q, a) pairs are also scarce, we train the retriever in an unsupervised manner independent from the (q, a) pairs. We leverage the NYT dataset, a distant supervision dataset for relation extraction (Riedel et al., 2010) to construct the pseudo (q, a, p) labels. In this dataset, each instance is denoted as a tuple $(s, (e_1, r, e_2))$, where s is a sentence that refers to the relation r between two entities e_1 and e_2 mentioned in the sentence s. For two instances $(s_1, (e_1, r_1, e_2))$ and $(s_2, (e_2, r_2, e_3))$, we treat e_1 as the topic entity and e_3 as the answer. Then we concatenate s_1 and s_2 as the question, and concatenate r_1 and r_2 as the corresponding path to train the retriever. The training objective is the same as the weakly supervised pre-training.

5.3 End-to-End Fine-tuning

End-to-end training is an alternative to fine-tune the separately trained retriever and the reasoner jointly. The main idea is to leverage the feedback from the reasoner to guide the path expansion of the retriever. To enable this, we optimize the posterior $p_{\theta,\phi}(\mathcal{G}|q, a)$ instead of the prior $p_\theta(\mathcal{G}|q)$, since the former one contains the additional likelihood $p_\phi(a|q, p_k)$ which exactly reflects the feedback from the reasoner. We do not directly optimize the posterior $p_{\theta,\phi}(\mathcal{G}|q, a)$, because $\mathcal{G}$ is induced from nK paths, making it unknown which path should receive the feedback from the likelihood computed on the whole $\mathcal{G}$. Instead, we approximate $p(\mathcal{G}|q, a)$ by the sum of the probabilities of the nK paths and rewrite the posterior of each path by Bayes' rule (Sachan et al., 2021), *i.e.*,

$$\begin{aligned} p_{\theta,\phi}(\mathcal{G}|q,a) &\approx \sum_{k=1}^{nK} p_{\theta,\phi}(p_k|q,a), \\ &\propto \sum_{k=1}^{nK} p_\phi(a|q,p_k) p_\theta(p_k|q), \end{aligned} \tag{8}$$

where $p_\theta(p_k|q)$ is the prior distribution of the k-th path that can be estimated by Eq. (7), and $p_\phi(a|q, p_k)$ is the likelihood of the answer a given the k-th path. Essentially, $p_\phi(a|q, p_k)$ estimates the answer a on the single tree induced by the k-th path instead of the fused subgraph by nK paths. As a result, the reasoning likelihood on each tree can be reflected to the corresponding path that induces the tree. The reasoner for estimating $p_\phi(a|q, p_k)$ is the same as that for calculating $p_\phi(a|q, \mathcal{G})$.

In summary, the whole objective function for each training instance $(q, a, \mathcal{G})$ is formalized as:

$$\begin{aligned} \mathcal{L} &= \underbrace{\max_\phi \log p_\phi(a|q,\mathcal{G})}_{\text{Reasoner}} \\ &+ \underbrace{\max_\theta \log \sum_{k=1}^{nK} \mathbb{SG}(p_\phi(a|q,p_k)) p_\theta(p_k|q)}_{\text{Retriever}}, \end{aligned} \tag{9}$$

where the stop-gradient operation $\mathbb{SG}$ is to stop updating the parameters ϕ. The reasoner is updated

the same as the two-stage training by computing the likelihood $p_\phi(a|q, \mathcal{G})$ on $\mathcal{G}$ sampled by the retriever (without using information from the answer a). As a result, there is no mismatch between the training and evaluation when computing $p_\phi(a|q, \mathcal{G})$, as $\mathcal{G}$ relies only on the prior at both.

Intuitively, we train the reasoner to extract the correct answer given the subgraph induced from nK highest scoring paths. And we train the retriever to select nK paths which collectively have a high score to deduce the answer when taking the feedback from the reasoner into account. Although the two components are jointly trained, the reasoning is still performed on the retrieved entire subgraph at each epoch. We present the training process in Appendix.

6 Experiments

In this section, we conduct extensive experiments to evaluate the subgraph retrieval (SR) enhanced model. We design the experiments to mainly answer the four questions: (1) Does SR take effect in improving the QA performance? (2) Can SR obtain smaller but higher-quality subgraphs? (3) How does the weakly supervised and unsupervised pre-training affect SR's performance? (4) Can end-to-end fine-tuning enhance the performance of the retriever as well as the reasoner?

6.1 Experimental Settings

Datasets. We adopt two benchmarks, WebQuestionSP (WebQSP) (Yih et al., 2016) and Complex WebQuestion 1.1 (CWQ) (Talmor and Berant, 2018), for evaluating the proposed KBQA model. Table 1 shows the statistics.

Evaluation Metrics. We evaluate the retriever by the answer coverage rate, which is the proportion of questions for which the topic-nK retrieved paths contain at least one answer. This metric reflects the upper bound of the QA performance and is denoted as Hits@K. For QA performance, We use Hits@1 to evaluate whether the top-1 predicted answer is correct. Since some questions have multiple answers, we also predict the answers by the optimal threshold searched on the validation set and evaluate their F1 score.

Baseline Models. We compare with embedding-based KBQA models, in which EmbedKGQA (Saxena et al., 2020) directly optimizes the triplet (topic entity, question, answer) based on their direct embeddings. KV-Mem (Miller et al., 2016) BAMNet (Chen et al., 2019a) store triplets in a key-value structured memory for reasoning. GRAFT-Net (Sun et al., 2018), BAMNet (Chen et al., 2019a), NSM (He et al., 2021), and PullNet (Sun et al., 2019) are subgraph-oriented embedding models. We also compare with the SP-based models, in which QGG (Lan and Jiang, 2020) generates the query graph for a question by adding the constraints and extending the relation paths simultaneously, SPARQA (Sun et al., 2020) proposes a novel skeleton grammar to represent a question, and CBR-KBQA (Das et al., 2021) leverages BigBird (Zaheer et al., 2020), a pre-trained seq2seq model to directly parse a question into a SPARQL statement that can be executed on graph DBs. SR is default trained by weakly supervised pre-training and the default path number is set to 10.

Table 1: Data statistics. The number of QA pairs for training, validating and testing are presented.

Dataset	#Train	#Validation	#Test
WebQSP	2,848	250	1,639
CWQ	27,639	3,519	3,531

Table 2: QA performance on WebQSP and CWQ (%).

Model	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
SP-based models				
SPARQA	-	-	31.6	-
QGG	73.0	73.8	36.9	37.4
CBR-KBQA	-	72.8	-	**70.0**
Embedding-based models				
KV-Mem	46.6	34.5	18.4	15.7
EmbedKGQA	66.6	-	32.0	-
BAMnet	55.6	51.8	-	-
GRAFT-Net (GN)	66.4	60.4	36.8	32.7
NSM	68.5	62.8	46.3	42.4
PullNet	68.1	-	45.9	-
Our Models				
SR+NSM	68.9	64.1	**50.2**	**47.1**
SR+GN	65.2	61.2	46.5	41.4
SR+NSM w E2E	**69.5**	**64.1**	49.3	46.3
SR+GN w E2E	66.7	63.1	49.0	42.7

6.2 Overall QA Evaluation

We compare with state-of-the-art KBQA models and present the Hits@1 and F1 scores in Table 2.

Table 3: The answer coverage rate of SR on WebQSP and CWQ (%).

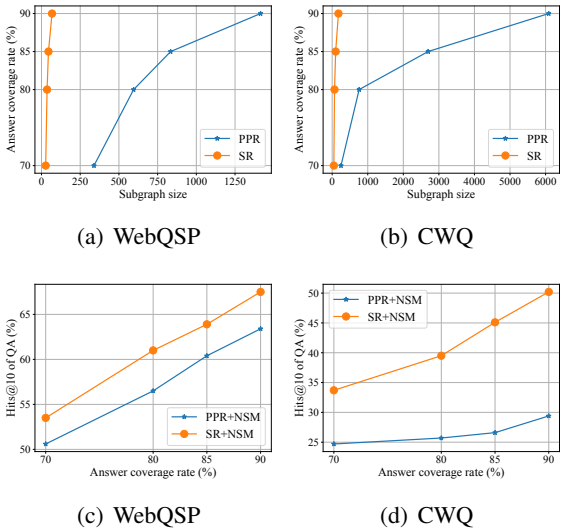
Model	WebQSP				CWQ			
	Hits@1	Hits@5	Hits@10	Hits@20	Hits@1	Hits@5	Hits@10	Hits@20
SR w SuperT	78.0	90.7	90.5	95.0	-	-	-	-
SR	71.2	87.5	90.5	92.9	51.5	72.6	79.5	83.7
SR w/o QU	66.9	82.6	86.2	89.4	36.5	46.4	50.8	54.9
SR w/o PE	52.7	71.4	74.8	78.1	49.4	62.3	66.6	71.6
SR+NSM w E2E	79.3	89.8	91.6	93.3	53.5	73.4	78.4	82.8
SR+GN w E2E	81.8	90.2	92.9	94.4	56.1	74.5	79.6	83.9

SP-based Models. The SP-based model CBR-KBQA achieves the best performance on CWQ. This is expected, as CBR-KBQA leverages a pre-trained seq-to-seq model to parse the input question into a SPARQL statement. However, the model depends on the annotated SPARQL statements, which are expensive to be annotated in practice.

Embedding-based Models. Among these models, KV-Mem and EmbedKGQA retrieve the answers from the global key-value memory built on the KB or the original whole KB, which enjoys high recall but suffers from many noisy entities. Compared with these global retrievals, BAMNet builds the key-value memory on a subgraph, but it is a full multi-hop topic-entity-centric subgraph, which is also noisy. GRAFT-Net and NSM calculate PPR scores to control the subgraph size, but the ad-hoc retrieval method is still far from optimal. PullNet reinforces the retrieval by learning a retriever, but the retriever and the reasoner are intertwined, causing the partial reasoning on part of a subgraph, which increases the reasoning bias.

Our Models. Compared with the above embedding-based models, a performance improvement on both the datasets can be observed, *e.g.*, NSM injected by SR (SR+NSM) improves 0.4% Hits@1 and 1.3% F1 on WebQSP, 3.9% Hits@1 and 4.7% F1 on CWQ compared with the original NSM. We also show that SR can be adapted to different subgraph-oriented reasoners. Beyond NSM, when injecting SR to GRAFT-NET, it also significantly improves 9.7% Hits@1 and 8.7%F1 on CWQ. SR+GN underperforms GN on WebQSP because GN filters out the relations of the knowledge graph not in the training set of WebQSP. We do not inject SR into BAMNet as the model needs entity types in the subgraph, which is temporarily ignored by SR .

Summary. The overall evaluation shows that SR takes effect in improving the QA performance when injecting it before a subgraph-oriented reasoner, and SR equipped with NSM creates a new state-of-the-art model for embedding-based KBQA.



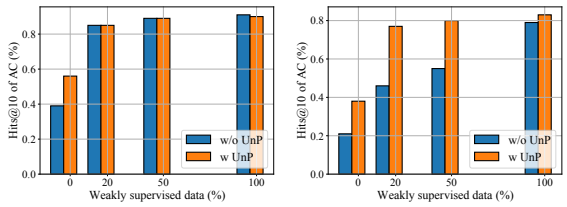
(a) WebQSP (b) CWQ

(c) WebQSP (d) CWQ

Figure 4: Comparison of the answer coverage rate under various subgraph sizes (Top row) and the QA performance (Hits@1) under various answer coverage rates (Bottom row).

6.3 Retriever Evaluation

Quality of Retrieved Subgraph. We evaluate whether the proposed SR can obtain smaller but higher-quality subgraphs, which are measured by not only the direct subgraph size and answer coverage rate but also the final QA performance. For a fair comparison, we fix the reasoner as NSM, and vary the retriever as SR and the PPR-based heuristic retrieval (Sun et al., 2018; He et al., 2021). PPR+NSM are performed on the same knowledge graph of the proposed SR+NSM. The result of the trainable retriever in PullNet (Sun et al., 2019) is ignored, because its code is not published and the



(a) WebQSP (b) CWQ

Figure 5: Retriever's performance by pre-training in terms of Hits@10 of answer coverage rate (AC). UnP denotes unsupervised pre-training.

value of some key parameters that seriously impact the model's performance is unknown.

We report the comparison results in Figure 4. The top row presents the answer coverage rates of the subgraphs with various sizes. It is shown that when retrieving the subgraphs of the same size, the answer coverage rate of SR is significantly higher than PPR. The bottom row presents the QA performance (Hits@1) on the subgraphs with various answer coverage rates. It is shown that by performing the same NSM on the subgraphs with the same coverage rate, the subgraphs retrieved by SR can result in higher QA performance than PPR.

Summary. The above results show that SR can obtain smaller but higher-quality subgraphs.

Effect of Question Update, Path Ending, and Subgraph Merge. We investigate the effects of the strategies used in SR, including the question updating strategy (QU) which concatenates the original question with the partially expanded path at each time step, the path ending strategy (PE) which learns when to stop expanding the path, and the subgraph merging strategy (GM) which induces a subgraph from the top-nK paths.

Table 3 indicates that based on SR, Hits@1 drops 4.3-15.0% when removing QU (SR w/o QU) and Hits@1 drops 2.1-18.5% when changing PE to the fixed path length T (SR w/o PE), where the optimal T is set to 3 on both WebQSP and CWQ.

Table 4 shows that based on SR+NSM, the average subgraph size increases from 174 to 204, and Hits@1 of QA drops 0.1% when removing the subgraph merging strategy (SR+NSM w/o GM) but directly taking the union of all the subgraphs from different topic entities to induce the subgraph. We only present the results on CWQ as most of the questions in WebQSP only contain one topic entity, which does not need the merge operation.

Table 4: Performance of subgraph merging strategy (GM) on CWQ.

Model	Subgraph size	Hits@1 of QA (%)
SR+NSM	174	50.2
SR+NSM w/o GM	204	50.1

Summary. The above results verify the effectiveness of the devised QU, PE, and GM in SR.

6.4 Training Strategy Evaluation

Effect of Pre-training. We investigate the effects of the weakly supervised and the unsupervised pre-training on the SR. Table 3 shows the performance of the supervised training (SR w SuperT) and the weakly supervised pre-training (SR), which indicates that SR is comparable with SR w SuperT when retrieving top-10 paths. Because a single ground-truth path between a topic entity and an answer is provided by WebQSP, which might omit the situation when multiple ground truth paths can be found. In view of this, the weakly supervised way that retrieves multiple shortest paths as the ground truth can provide richer supervision signals. We ignore the supervised training in CWQ because the ground truth paths are not explicitly given in the dataset.

We further vary the proportion of the weakly supervised data in {0%, 20%, 50%, 100%}, and present the corresponding answer coverage rate of the subgraph induced by top-10 paths (*i.e.* Hits@10) in Figure 5. Note 0% means the RoBERTa used in SR don't have any fine-tuning. The performance shows a consistent growth with the weakly generated data size, which demonstrates its positive effect.

Before the weakly supervised pre-training, we create 100,000 pseudo instances for unsupervised pre-training (Cf. Section 5 for details). The results presented by the orange bars show that unsupervised pre-training can significantly improve the original SR (0% weakly supervised data) by about 20% Hits@1. However, with the increase of the weakly-supervised data, adding unsupervised pre-training does not take better effect.

Summary. The above results show the effectiveness of the weakly supervised pre-training. Meanwhile, the unsupervised strategy can be an alternative choice when the QA pairs are scarce.

Effect of End-to-End Fine-tuning. Table 3 shows

both SR+NSM w E2E and SR+GN w E2E improve 2-10.6% Hits@1 of retrieval based on SR. Table 2 shows SR+NSM w E2E improves 0.6% Hits@1 of QA based on SR+NSM on WebQSP, and SR+GRAFT-Net w E2E improves 1.5-2.5% Hits@1 of QA based on SR+GRAFT-Net. Although SR+NSM w E2E underperforms SR+NSM on CWQ, we suggest to reason on the top-1 retrieved results, which are much better than those before fine-tuning .

Summary. The above results indicate that the answer likelihood estimated by the reasoner provides positive feedback for fine-tuning the retriever. With the improvement of the retriever, the reasoner can be also enhanced by the updated subgraphs.

7 Conclusion

We propose a subgraph retriever (SR) decoupled from the subsequent reasoner for KBQA. SR is devised as an efficient dual-encoder that can update the question when expanding the path as well as determining the stop of the expansion. The experimental results on two well-studied benchmarks show SR takes effect in improving the QA performance if injecting it before a subgraph-oriented reasoner. SR equipped with NSM creates new SOTA results for embedding-based KBQA methods if learning SR by weakly supervised pre-training as well as end-to-end fine-tuning.

Acknowledgments

This work is supported by National Natural Science Foundation of China (62076245, 62072460, 62172424); National Key Research & Develop Plan(2018YFB1004401); Beijing Natural Science Foundation (4212022); CCF-Tencent Open Fund.

References

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

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019a. Bidirectional attentive memory networks for question answering over knowledge bases. In *Proceedings of NAACL-HLT*, pages 2913–2923.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019b. Uhop: An unrestricted-hop relation extraction framework for knowledge-based question answering. In *Proceedings of NAACL-HLT*, pages 345–356.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611.

Yu Feng, Jing Zhang, Gaole He, Wayne Xin Zhao, Lemao Liu, Quan Liu, Cuiping Li, and Hong Chen. 2021. A pretraining numerical reasoning model for ordinal constrained question answering on knowledge base. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1852–1861.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938.

Taher H Haveliwala. 2003. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering*, 15(4):784–796.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 553–561.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 23–33.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Yunqi Qiu, Yuanzhuo Wang, Xiaolong Jin, and Kun Zhang. 2020a. Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 474–482.

Yunqi Qiu, Kun Zhang, Yuanzhuo Wang, Xiaolong Jin, Long Bai, Saiping Guan, and Xueqi Cheng. 2020b. Hierarchical query graph generation for complex question answering over knowledge graph. In *Proceedings of the 29th ACM International Conference on Information Knowledge Management*, page 1285–1294.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Devendra Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. 2021. End-to-end training of neural retrievers for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6648–6662.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Haitian Sun, Tania Bedrax Weiss, and William W. Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 2380–2390.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8952–8959.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 641–651.

Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing key-value memory neural networks for knowledge based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2937–2947.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second Conference on Artificial Intelligence*, pages 6069–6076.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2010–2022.

Table 5: Paths learned by the proposed retriever SR.

Question	Top-nK (K=2, n is #topic entities) Retrieved Paths
Which airport to fly into <Rome>?	p_1 = Rome:(travel_destination, transportation) p_2 = Rome:(nearby_airports)*
<Santo Domingo> is the capital of the country with what currency?	p_1 = Santo Domingo:(capital_of, currency_used) p_2 = Santo Domingo:(containedby, currency_used)*
What is the home county of the person who said <"Few people have the virtue to withstand the highest bidder?">?	p_1 = (quotation_author, place_of_birth) p_2 = (quotation_author, places_lived)*
What is the most recent movie directed by <Angelina Jolie>?	p_1 = Angelina Jolie:(director_of_film, release_date) p_2 = Angelina Jolie:(producer_of_film, release_date)*
What movie, written by <Katerine Fugate>, featured <Taylor Swift>?	p_1 = Katerine Fugate:(film_writer) p_2 = Katerine Fugate:(film_story_contributor)* p_3 = Taylor Swift:(actor_film) p_4 = Taylor Swift:(nominations, nominated_for)*
What country bordering <Argentina> is in the <Brasilia Time Zone>?	p_1 = Argentina:(location_adjoins) p_2 = Argentina:(import_from)* p_3 = Brasilia Time Zone:(location_time_zone) p_4 = Brasilia Time Zone:(location_used, contained_by)*
Where was the artist that had <This Summer Tour> raised?	p_1 = This Summer Tour:(music_artist, place_of_birth) p_2 = This Summer Tour:(music_artist, artist_origin)*
What position did <Vince Lombardi> play?	p_1 = Vince Lombardi:(basketball_player_position) p_2 = Vince Lombardi:(teams, team_roster_position)*
What city is <Acadia University> in?	p_1 = Acadia University:(headquarters, mailing_address) p_2 = Acadia University:(contained_by)*
What was the first book written by the author of <"The Cricket on the Hearth">?	p_1 = "The Cricket on the Hearth":(author, works_written) p_2 = "The Cricket on the Hearth":(author, book_published)*
Who married to the person who lived in <Downe, Kent>?	p_1 = Downe, Kent:(person_lived, spouse) p_2 = Downe, Kent:(person_lived, children, parent)*
Find the location of the film <Fan Chan>, what language is spoken there?	p_1 = Fan Chan:(film_location, languages_spoken) p_2 = Fan Chan:(film_location, official_language)*

A Appendix

A.1 Interpretability of Retrieved Paths.

We present the top-nK paths learned by the proposed SR for several questions in Table 5 on WebQSP and CWQ. Each path is denoted by its topic entity before the colon. A path denoted by * means it is the new path discovered by SR beyond the ground-truth path provided by WebQSP and CWQ. The paths can explain why an answer is inferred for a question.

A.2 Training Algorithm.

We present the whole training process in Algorithm 1, where we first pre-train the retriever, then train the reasoner based on the retrieved subgraph, and finally end-to-end fine-tune the retriever and the reasoner together.

Algorithm 1: Training Algorithm

Input: $G, \{(q, a)\}$
Output: Learned parameters θ and ϕ.

```
Pre-train the retriever by weakly supervised signals or unsupervised signals plus only 20% weakly supervised signals;
Train the reasoner on the retrieved subgraphs;
/* End-to-End training:     */
while not converge do
    For each (q, a) pair, sample a subgraph G by current retriever;
    Update φ by optimizing the first term of Eq. (9) on all the (q, a, G) instances;
    Update θ by optimizing the second term of Eq. (9) on all the (q, a, G) instances;
end
```

A.3 Experimental Implementation

We provide the training and inference details of all the experiments as below.

General Setting. We use RoBERTa-base in our paper (Liu et al., 2019). The basic RoBERTa contains 12 layers, 768-d hidden size, and 12 attention heads, resulting in 110M parameters in total. On WebQSP and CWQ, the batch size for training both the retriever and the reasoner is set as 16 and 20 respectively.

Supervised Training. WebQSP provides the relation chains corresponding to each (question, answer) pair. For each question, we use each relation chain from each topic entity to the answer as the ground truth path. In this way, we obtain 3,098 (question, path) instances which can be decomposed into 5,394 (question, relation) instances in total for supervised training. The learning rate for supervised training is set as 5e-5. An epoch takes about 5 minutes and the loss function converges within 10 epochs on WebQSP/CWQ. We ignore supervised training on CWQ because the explicit paths are not provided.

Weakly supervised Pre-training. For weakly supervised pre-training, the ground truth paths are unavailable. To create the pseudo paths, for each (question, answer) pair, we extract all the shortest paths between each topic entity and an answer. We create 16,000/150,000 (question, relation) instances in total for weakly supervised pre-training. The learning rate for weakly supervised training is set as 5e-5. An epoch takes about 5 minutes and the loss function converges within 10 epochs.

Unsupervised Pre-training. From the NYT dataset (Riedel et al., 2010), we create 100,000 (sentence, path) pseudo instances. The learning rate for unsupervised training is set as 5e-5. An epoch takes about 5 minutes and the loss function converges within 10 epochs. The unsupervised pre-training is performed once and then SR can be adapted to various KBQA datasets.

End-to-End Training. Before end-to-end training, the retriever needs to be warmed up by weakly supervised pre-training or unsupervised pre-training. The reasoner also needs to be warmed up by supervised training on the (question, answer) pairs. For training the NSM reasoner (He et al., 2021), an epoch with batch size 20 takes 55 seconds and the loss function converges within 80 epochs. The learning rate for warming up the reasoner is set as 1e-4. For end-to-end training, the learning rate is set as 1e-5.

A.4 Inference

We retrieve the top 10 relevant relations at each step which results in 10 paths for each topic entity. The number 10 is determined at the pre-training stage by checking the inflection point of the answer coverage rate on the validation set. The average time of online inference including both the subgraph retrieving and the reasoning can be within 1 second. By comparison, GRAFT-Net and NSM which first retrieve the whole two-hop subgraph and then prune it by the PPR scores spend about 2 to 3 seconds or even 7 to 8 seconds for retrieving some dense subgraphs.