arXiv:2110.04330v1 [cs.CL] 8 Oct 2021

KG-FiD: Infusing Knowledge Graph in Fusion-in-Decoder for Open-Domain Question Answering

Donghan Yu[1]*, **Chenguang Zhu**[2], **Yuwei Fang**[2], **Wenhao Yu**[3]*, **Shuohang Wang**[2], **Yichong Xu**[2], **Xiang Ren**[4], **Yiming Yang**[1], **Michael Zeng**[2]
[1]Carnegie Mellon University [2]Microsoft Cognitive Services Research Group
[3]University of Notre Dame [4]University of Southern California
[1]{dyu2,yiming}@cs.cmu.edu, [2]chezhu@microsoft.com

Abstract

Current Open-Domain Question Answering (ODQA) model paradigm often contains a retrieving module and a reading module. Given an input question, the reading module predicts the answer from the relevant passages which are retrieved by the retriever. The recent proposed Fusion-in-Decoder (FiD), which is built on top of the pretrained generative model T5, achieves the state-of-the-art performance in the reading module. Although being effective, it remains constrained by inefficient attention on all retrieved passages which contain a lot of noise. In this work, we propose a novel method KG-FiD, which filters noisy passages by leveraging the structural relationship among the retrieved passages with a knowledge graph. We initiate the passage node embedding from the FiD encoder and then use graph neural network (GNN) to update the representation for reranking. To improve the efficiency, we build the GNN on top of the intermediate layer output of the FiD encoder and only pass a few top reranked passages into the higher layers of encoder and decoder for answer generation. We also apply the proposed GNN based reranking method to enhance the passage retrieval results in the retrieving module. Extensive experiments on common ODQA benchmark datasets (Natural Question and TriviaQA) demonstrate that KG-FiD can improve vanilla FiD by up to 1.5% on answer exact match score and achieve comparable performance with FiD with only 40% of computation cost.

1 Introduction

Open-Domain Question Answering (ODQA) is the task of answering natural language questions in open domains. A successful ODQA model relies on the acquisition of world knowledge. A popular line of works assume that the knowledge is explicitly stored in an extensive accessible external corpus. They first retrieve a small number of passages from the unstructured text corpus by a *retriever* module and then process the retrieved passages to generate answer by a *reader* module (Karpukhin et al., 2020; Guu et al., 2020; Izacard & Grave, 2020b). One of the early representative works in this paradigm is DrQA (Chen et al., 2017), which uses a TF-IDF based retriever and a neural reader to extract answer from the retrieved passages. With the development of Pre-trained Language Models (PLMs), it shows that dense passage retrieval (DPR) (Karpukhin et al., 2020; Qu et al., 2021) outperforms TF-IDF based retrieval (Chen et al., 2017; Yang et al., 2019). And generative QA systems (Lewis et al., 2020; Roberts et al., 2020) outperform extraction based systems (Devlin et al., 2018; Guu et al., 2020).

Among these readers, Fusion-in-Decoder (FiD) (Izacard & Grave, 2020a) achieves the state-of-the-art performance in ODQA by building generative seq2seq model T5 (Raffel et al., 2019) on top of DPR retrieved passages. The key benefit of FiD is that the answer decoder can efficiently fuse the information from multiple passages. In detail, it separately encodes retrieved passages, each

*Work done during internship at Microsoft.

appended with question, and then concatenates all resulting passage token embeddings to send to the decoder. Although being effective, FiD has two main drawbacks.

(1) *Effectiveness*. Some of the retrieved passages are irrelevant to the input question, which can be noise and hurt the model performance. (2) *Efficiency*. Jointly modeling all retrieved passages (100 in the original paper) is inefficient and limits the real application of such model. Simply reducing the number of retrieved passages sent to the reader will significantly reduce the model performance (Izacard & Grave, 2020a). Therefore, it still remains unstudied how one can improve both the effectiveness and efficiency of FiD.

Besides unstructured text corpus, the world knowledge required to answer questions also exists in knowledge graphs (KG), which represent entities and relations in a structural way. Based on the alignment information between KG entities and passages, one can apply KG to build the inter-relationship between retrieved passages to improve the retrieval accuracy. Thus, we propose a novel method KG-FiD to address both the effectiveness and efficiency drawbacks of FiD by jointly training knowledge graph based passage reranking with answer decoder.

To improve effectiveness, KG-FiD employs passage reranking with knowledge graph, by applying graph neural networks (GNNs) to model structural and semantic information of passages and output scores for reranking. To initialize the node embeddings of GNN, we use the passage embeddings output from the FiD encoder. After reranking the input passages, only a few top-reranked passages are fed into decoder for answer generation, which reduces the number of noisy passages.

To improve efficiency, KG-FiD adopts the intermediate layer representations instead of the final layer in the FiD encoder to initiate passage node embeddings for reranking. Then only a few top reranked passages will be passed into the higher layers of encoder and the decoder for answer generation. This is coupled with a joint training of passage reranking and answer generation. As shown in Section 4.3, this can significantly reduce computation cost while still maintaining a good answer generation performance.

Furthermore, instead of directly using the DPR retriever as the original FiD paper, we propose to apply the GNN based reranking model after the DPR retriever to enhance passage retrieval results and further boost the performance of our model. We use the passage embeddings generated by DPR to initial the node embeddings of GNN, which allows reranking from a much larger set of initial candidate passages from DPR to enhance coverage of answers.

We summarize our contributions as follows:

1. We are the first work fusing knowledge graph information into FiD for open domain QA.
2. We are the first work to jointly optimize graph-based reranker and decoder in FiD. And we propose a solution to make our model KG-FiD much more efficient than vanilla FiD.
3. We conduct extensive experiments on commonly-used ODQA benchmark datasets: Natural Questions and TriviaQA. The results show that KG-FiD can significantly improve both the accuracy and efficiency of baseline models. KG-FiD outperforms the vanilla FiD by up to 1.5% in answer exact match score and achieves on-par performance with FiD with only 40% of its computation cost.

2 Related Work

ODQA with text corpus ODQA usually assumes that a large external knowledge source is accessible and can be leveraged to help answer prediction. For example, previous works (Chen et al., 2017; Karpukhin et al., 2020; Izacard & Grave, 2020a) mainly use Wikipedia as knowledge source which contains millions of text passages. In this case, current ODQA models mainly contains a retriever to select related passages and a reader to generate the answer based on the retrieved passages. Thus, the follow-up works mainly aim to: (1) Improve retriever: from sparse retrieval based on TF-IDF or BM25 (Chen et al., 2017; Yang et al., 2019) to dense retrieval (Karpukhin et al., 2020) based on contextualized embeddings generated by pre-trained language models (PLMs). Moreover, some further improvement are also proposed such as better training strategy (Qu et al., 2021), reranking based on retrieved passages (Wang et al., 2018; Nogueira & Cho, 2019; Mao et al., 2021), and knowledge distillation from reader to retriever (Izacard & Grave, 2020b); (2) Improve the reader: changing from Recurrent Neural Network (Chen et al., 2017) to PLMs such as extractive reader BERT (Karpukhin et al., 2020; Iyer et al., 2021; Guu et al., 2020) and generative reader BART and T5 (Izacard & Grave, 2020a; Lewis et al., 2020). Besides, some works (Guu et al., 2020; Lewis

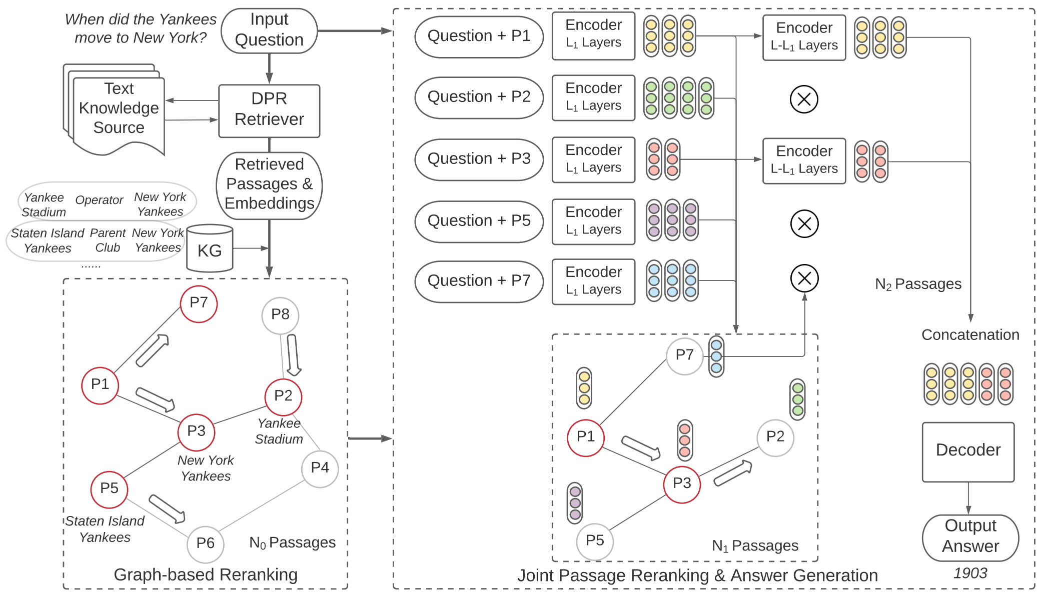


Figure 1: Overall Model Framework. Pi indicates the node of the passage originally ranked the i-th by the DPR retriever, with the article title below it. The left part shows passage retrieval by DPR, passage graph construction based on KG (Section 3.1) and graph based retriever reranking (Section 3.4). The right part shows how we improve the FiD reader by joint passage reranking and answer generation (Section 3.2 and 3.3).

et al., 2020; Sachan et al., 2021a) have shown that additional unsupervised pre-training on retrieval-related language modeling tasks can further improve ODQA performance. However, none of them studied the noise and inefficiency issues of the current best-performed reader FiD (Izacard & Grave, 2020a).

ODQA with knowledge graph Besides the unstructured text corpus, world knowledge also exists in knowledge graphs (KGs), which represent entities and relations in a structural way. Some works (Berant et al., 2013; Sun et al., 2018; 2019; Xiong et al., 2019) restrict the answer to be entities in the knowledge graph and focus on the modeling of entity and relation embeddings for answer prediction. While our work focus on more general ODQA setting where the answer can be any words or phrases. Under this setting, some recent efforts have been made to leverage knowledge graphs for ODQA. For example, UniK-QA (Oguz et al., 2020) transforms KG triplets into text sentences and combine them into text corpus, which loses structure information of KG. Other works use KG to build relationship among passages similar to ours. KAQA (Zhou et al., 2020) use passage graph to propagate passage retrieve scores and answer span scores. Graph-Retriever (Min et al., 2019) iteratively retrieve passages based on the relationship between passages, and also use passage graph to improve passage selection in an extractive reader. However, none of them studied applying KG to improve the FiD model.

3 METHOD

The overview of our framework is illustrated in Figure 1. Given an input question, we first apply DPR (Karpukhin et al., 2020), a BERT-based dual-encoder framework, to retrieve candidate passages from the large corpus. Then the reading module read the retrieved passages, along with the question, to generate an answer. In this work, we use Fusion-in-Decoder (FiD) model for the reading module which achieves the state-of-the-art reader performance.

As the focus in this work is to improve FiD via knowledge graph (KG), we first introduce how to apply KG to build a graph structure among the retrieved passages (Section 3.1). Then we illustrate improving the effectiveness of FiD by graph based reranking (Section 3.2) and improving the efficiency of FiD by using intermediate layer representation (Section 3.3). Finally we show how we adopt the graph based reranking method with DPR retriever to further boost model performance (Section 3.4).

3.1 Construct Passage Graph using KG

The intuition behind using KG is that there exists the structural relationship among the retrieved passages which can be captured by the KG. Similar to Min et al. (2019), we construct the passage graph where vertices are passages of text and the edges represent the relationships that are derived from the external KGs as $\mathcal{KG} = \{(e_h, r, e_t)\}$, where e_h, r, e_t are the head entity, relation and tail entity of a triplet respectively.

First, we formalize the definition of a *passage*. Following previous works (Wang et al., 2019; Karpukhin et al., 2020), each article in the text corpus is split into multiple disjoint text blocks of 100 words called *passages*, which serve as the basic retrieval units.

We assume there is a one-one mapping between the KG entities and articles in the text corpus. Specifically, we use English Wikipedia as the text corpus and English Wikidata (Vrandečić & Krötzsch, 2014) as the knowledge graph, since there exists an alignment between the two resources[1]. For example, for the article titled with "New York Yankees", it contains passages such as "The New York Yankees are an American professional baseball team ...". The article also corresponds to a KG entity with the same name as "New York Yankees".

Then we define the mapping function $e = f(p)$, where the KG entity e corresponds to the article which p belongs to. Note that one passage can only be mapped to one entity, but multiple passages could be mapped to the same entity. The final passage graph is defined as $\mathcal{G} = \{(p_i, p_j)\}$, where passages p_i and p_j are connected if and only if their mapped entities are directly connected in the KG, i.e., $(f(p_i), r, f(p_j)) \in \mathcal{KG}$.

Since the total number of passages is very large, e.g., more than 20M in Wikipedia, constructing and maintaining a graph over all the passages is inefficient and memory-consuming. Thus, we build a passage graph on the fly for each question, based on the retrieved passages.

3.2 Improving FiD Effectiveness via Graph-based Reranking

In this section, we briefly introduce the vanilla FiD reading module before illustrating our graph-based passage reranking method. By default, we suppose the reader takes $N_1 = 100$ retrieved passages $\{p_{r_1}, p_{r_2}, \cdots, p_{r_{N_1}}\}$ as input.

Vanilla Reading Module: We denote the hidden dimension as H and number of encoder layers and decoder layers as L, FiD first separately encodes each passage p_{r_i} concatenated with question q:

$$\mathbf{P}_i^{(0)} = \text{T5-Embed}(q + p_{r_i}) \in \mathbb{R}^{T_p \times H},\ \mathbf{P}_i^{(l)} = \text{T5-Encoder}_l(\mathbf{P}_i^{(l-1)}) \in \mathbb{R}^{T_p \times H} \tag{1}$$

where T_p is the sequence length of a passage concatenated with the question. T5-Embed$(\cdot)$ is the initial embedding layer of T5 model (Raffel et al., 2019) and T5-Encoder$_l(\cdot)$ is the l-th layer of its encoder module. Then the token embeddings of all passages output from the last layer of the encoder are concatenated and sent to the decoder to generate the answer tokens $\mathbf{A}$:

$$\mathbf{A} = \text{T5-Decoder}([\mathbf{P}_1^{(L)}; \mathbf{P}_2^{(L)}; \cdots; \mathbf{P}_{N_1}^{(L)}] \in \mathbb{R}^{N_1 T_p \times H}) \tag{2}$$

Graph-based Passage Reranking: Since the decoding process of vanilla FiD will jointly model all the retrieved passages, it becomes vulnerable to the noisy irrelevant passages. Thus, we propose to rerank the input N_1 passages during the encoding and only select top-N_2 ($N_2 < N_1$) reranked passages into the decoder. By such reranking, we aim to filter out irrelevant passages so that they'll not go through the decoding process.

Our model is based on both the structural graph information and the textual semantic information of passages. The previous section introduced the construction of passage graph, which we denote

[1]Entity recognition and linking can be used if there is no such alignment.

as $\mathcal{G}_1$. To represent the semantic information of passages, one can use another pre-trained language model to encode the passage texts, but it can incur heavy computational cost as N_1 is large and introduce lots of additional model parameters. Here we propose to reuse the encoder-generated question-aware passage representation from FiD for passage reranking as it is already computed in Equation 1.

Specifically, the initial passage embeddings $Z_i^{(0)}$ comes from the [CLS] token embedding of the final layer in the FiD-Encoder, i.e., $Z_i^{(0)} = \mathbf{P}_i^{(L)}([\text{CLS}])$. In this way, no additional contextual computation is needed for reranking.

We then employ a GAT (Veličković et al., 2017) with L_g layers as the graph neural network (GNN) model to update representations for each node based on the passage graph. The l-th layer of the GNN model updates the embedding of node i as follows:

$$Z_i^{(l)} = h(Z_i^{(l-1)}, \{Z_j^{(l-1)}\}_{(i,j)\in\mathcal{G}_1}) \tag{3}$$

where h is usually a non-linear learnable function which aggregates the embeddings of the node itself and its neighbor nodes. The reranking score of passage i is calculated by $s_i = W^T Z_i^{(L_g)}$ where W is a trainable model parameter. After reranking, the final top-N_2 ($N_2 < N_1$) passages are sent for decoding. Suppose their indices are $\{g_1, g_2, \cdots, g_{N_2}\}$, the decoding process is:

$$\mathbf{A} = \text{T5-Decoder}([\mathbf{P}_{g_1}^{(L)}; \mathbf{P}_{g_2}^{(L)}; \cdots; \mathbf{P}_{g_{N_2}}^{(L)}] \in \mathbb{R}^{N_2 T_p \times H}) \tag{4}$$

The training loss of passage ranking for each question is:

$$\mathcal{L}_r = -\sum_{i=1}^{N_1} y_i \log \hat{y}_i, \text{ where } \hat{y}_i = \frac{\exp(s_i)}{\sum_{j=1}^{N_1}\exp(s_j)} \tag{5}$$

where $y_i = 1$ if p_{r_i} is the gold passage[2] that contains the answer, and 0 otherwise. As a result, the passage reranking and answer generation are jointly trained. We denote the answer generation loss for each question is $\mathcal{L}_a$, then the final training loss of our reader module is $\mathcal{L} = \mathcal{L}_a + \lambda\mathcal{L}_r$, where λ is a hyper-parameter which controls the weight of our proposed reranking task in the total loss.

3.3 Improving FiD Efficiency via Intermediate Representation

Recall that in the section 3.2, we take the passage representation from the last layer of FiD-encoder for passage reranking. In this section, we propose to further reduce the computation cost by taking the intermediate layer representation rather than the last layer. Besides efficiency, another intuition is that answer generation task is more difficult than passage reranking which only needs to predict whether the passage contains the answer or not. Thus we may not need the representation from the whole encoder module for passage reranking.

Suppose we take the representation from the L_1-th layer ($1 \le L_1 < L$), i.e., $Z_i^{(0)} = \mathbf{P}_i^{(L_1)}([\text{CLS}])$ for $i \in \{1, 2, \cdots, N_1\}$, and the reranking method remains the same. Then only the top-N_2 ($N_2 < N_1$) reranked passages will go through the rest layers of FiD-encoder. Suppose their indices are $\{g_1, g_2, \cdots, g_{N_2}\}$, for $l \ge L_1 + 1$:

$$\mathbf{P}_i^{(l)} = \begin{cases} \text{T5-Encoder}_l(\mathbf{P}_i^{(l-1)}), & \text{if } i \in \{g_1, g_2, \cdots g_{N_2}\} \\ \text{Stop-Computing} & \text{else,} \end{cases} \tag{6}$$

Then $\mathbf{P}_{g_1}^{(L)}, \mathbf{P}_{g_2}^{(L)}, \cdots, \mathbf{P}_{g_{N_2}}^{(L)}$ are sent into the decoder for answer generation as in Equation 4. In Section 4.3, we demonstrate this can reduce 60% computation cost than the original FiD while keeping the on-par performance on two benchmark datasets.

Then we analyze the time complexity of our proposed KG-FiD. Suppose the length of answer sequence $\mathbf{A}$ is denoted as T_a. For vanilla FiD reader, the time complexity of the encoder module is $O(L \cdot N_1 \cdot T_p^2)$, where L, N_1 denote the number of encoder layers and the retrieved passages for reading and T_p is the average length of the passage. The square comes from the self-attention

[2]We follow (Karpukhin et al., 2020) on the definition of gold passages.

mechanism. The decoder time complexity is $O(L \cdot (N_1 \cdot T_p \cdot T_a + T_a^2))$, where $N_1 \cdot T_p \cdot T_a$ comes from the cross-attention mechanism. For our reading module, all the N_1 candidate passages are processed by the first L_1 layers of encoder. But only N_2 passages are processed by the remaining $L - L_1$ encoder layers and sent into the decoder. Thus, the encoder computation complexity becomes $O((L_1 \cdot N_1 + (L - L_1) \cdot N_2) \cdot T_p^2)$, and the decoder computation takes $O(L \cdot (N_2 \cdot T_p \cdot T_a + T_a^2))$. Because $L_1 < L, N_2 < N_1$, both the encoding and decoding of our method is more efficient than vanilla FiD.

Furthermore, suppose the answer is much shorter than the passage (which is the case in our experiments), i.e., $T_a \ll T_p$. Then the decoding computation is negligible compared to the encoding. In this case, the approximated ratio of saved computation cost brought by our proposed method is:

$$1 - \frac{(L_1 \cdot N_1 + (L - L_1) \cdot N_2) \cdot T_p^2}{L \cdot N_1 \cdot T_p^2} = (1 - \frac{L_1}{L})(1 - \frac{N_2}{N_1})$$

This shows that we can reduce more computation cost by decreasing L_1 or N_2. For example, if setting $L_1 = L/2, N_2 = N_1/5$, we can reduce 40% of computation cost. More empirical results and discussions will be presented in Section 4.3.

3.4 Applying Graph-based Reranking for Retriever

Our framework applies DPR (Karpukhin et al., 2020) as the retriever, which applies a BERT based passage encoder to encode all the N passages in the text corpus $\{p_1, p_2, \cdots, p_N\}$. In this section, we apply the proposed graph-based reranking for improving passage retrieval, since DPR independently retrieve N_1 passages based on the similarity with input question without considering inter-passage relationship. Besides, we note that there still exists valuable information out of the N_1 passages. Thus we propose to retrieve N_0 ($N_0 > N_1$) passages, then rerank them and finally input top-N_1 reranked passages into the reader.

Suppose all the passage embeddings are fixed and stored in memory as $M \in \mathbb{R}^{N \times D}$ where D is the hidden dimension. For an input question q, DPR applies a BERT based question encoder to obtain its representation Q:

$$M_i = \text{BERT}_{\text{passage}}(p_i) \text{ for } i \in \{1, 2, \cdots N\}, \; Q = \text{BERT}_{\text{question}}(q) \tag{7}$$

Then it applies FAISS (Johnson et al., 2019) to conduct fast dot-product similarity search between Q and M, and returns N_0 ($N_0 \ll N$) passages with the highest similarity scores.

Following Section 3.1, we construct a graph among the N_0 passages denoted as $\mathcal{G}_1'$. To avoid additional computation cost, we propose to reuse the offline passage embeddings M generated from the DPR retriever in Equation 7 as the initial passage representation: $E_i^{(0)} = M_{r_i}$ for $i \in \{1, 2, \cdots, N_0\}$. We then employ an L_g-layer GAT (Veličković et al., 2017) model to update passage node embeddings: $E^{(L_g)} = \text{GAT}(E^{(0)}, \mathcal{G}_1')$, which are used for reranking. Specifically, the reranking score for each passage p_{r_i} is calculated by $s_i = Q^T E_i^{(L)}$, where Q is the question embedding also generated by the retriever in Equation 7. Then we sort the retrieved passages by the reranking scores, and input the top-N_1 passages into the reader. To train the reranking, we adopt the same loss function as Equation 5.

As we only add a lightweight graph neural network and reuse the pre-computed and static DPR passage embeddings, our reranking module can process a large number of candidate passages efficiently for each question. This is different with the reader module, where the passage representation generation requires on-the-fly processing of a large pre-trained language model (T5-encoder). In experiments, we set $N_0 = 1000, N_1 = 100$. In Section 4.4, we demonstrate that such reranking can further boost the performance.

4 Experiment

In this section, we conduct extensive experiments on two most commonly-used ODQA benchmark datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019), which is based on Google Search Queries, and TriviaQA (Joshi et al., 2017), which contains questions from trivia and quiz-league websites. We follow the same setting as Izacard & Grave (2020a) to preprocess these datasets, which is introduced in Appendix A.1. All our experiments are conducted on 8 Tesla A100 40Gb GPUs.

4.1 Implementation Details

Knowledge Source: Following Karpukhin et al. (2020); Izacard & Grave (2020a), we use the English Wikipedia as the text corpus, and apply the same preprocessing to divide them into disjoint passages with 100 words, which produces 21M passages in total. For the knowledge graph, we use English Wikidata. The number of aligned entities, relations and triplets among these entities are 2.7M, 974 and 14M respectively.

Model Details: For the retrieving module, we use the DPR retriever (Karpukhin et al., 2020) which contains two BERT (base) models for encoding question and passage separately. For the GNN reranking models, we adopt 3-layer Graph Attention Networks (GAT) (Veličković et al., 2017). We also try different GNN model type and number of layers, which will be illustrated in Section 4.4. For the reading module, same as Izacard & Grave (2020a), we initialize it with the pretrained T5-base and T5-large models (Raffel et al., 2019). Our implementation is based on the HuggingFace Transformers library (Wolf et al., 2019). For number of passages, we set $N_0 = 1000, N_1 = 100, N_2 = 20$. We include the results of hyper-parameter search in Appendix A.3.

Training Process: For training our framework, we adopt the separate-training strategy: we first train the DPR model following its original paper, then freeze the DPR model to train the reranking in the retriever module, and finally the KG-FiD . For the training of retriever module, the optimizer is AdamW with learning rate as 1e-3 and linear-decay scheduler. The weight decay rate is 0.01. Batch size is set as 64. The number of total training steps is 15k, and the model is evaluated every 500 steps and the model with best validation results is saved as the final model. For the training of KG-FiD , we adopt the same training setting except that the learning rate is 1e-4 for the base model and 5e-5 for the large model. We also adopt learning rate warm up with 1000 steps.

Evaluation: We follow the standard evaluation metric of answer prediction in ODQA, which is the exact match score (EM) (Rajpurkar et al., 2016). A generated answer is considered correct if it matches any answer in the list of acceptable answers after normalization[3]. For all the experiments, we conduct 5 runs with different random seeds and report the averaged scores.

4.2 Baseline Methods

We mainly compare KG-FiD with the baseline model FiD (Izacard & Grave, 2020a). We also experiment with its variants FiD-KD (Izacard & Grave, 2020b), which trains a better retriever by performing knowledge distillation from the FiD reader. We adopt the retriever from FiD-KD to replace the DPR retriever , which we name as KG-FiD-KD. For other baselines, we compare with the state-of-the-art methods from each category: (1) leveraging knowledge graphs: Graph-Retriever (Min et al., 2019), KAQA (Zhou et al., 2020), and UniK-QA (Oguz et al., 2020); (2) performing additional unsupervised pre-training on a large corpus: REALM (Guu et al., 2020), RAG (Lewis et al., 2020), Joint Top-K (Sachan et al., 2021a) and EMDR[2] (Sachan et al., 2021b); (3) hybrid methods which contains multiple readers: R2-D2 (Fajcik et al., 2021), UnitedQA (Cheng et al., 2021).

4.3 Main Results

Comparison with Baselines: Table 1 shows the results of our method and all baselines. We see that our proposed model KG-FiD consistently and significantly improves FiD on both NQ and TriviaQA datasets over both base and large model. Specifically, for large model, KG-FiD improves FiD by 1.5% and 1.1% on two datasets respectively. When equipped with the retriever of FiD-KD, our model (denoted as KG-FiD-KD) also brings consistent improvements, such as 0.8% and 0.5% on base model over two datasets respectively. From the table, we see that our model KG-FiD-KD (large) outperforms all the baseline methods except two hybrid models R2-D2 and UnitedQA, which contain much more model parameters and complicated pipelines such as the combination of multiple

[3]The normalization includes lowercasing and removing articles, punctuation and duplicated whitespace.

readers. Even compared with the two hybrid models, our method significantly outperforms them on the TriviaQA dataset.

Model	#params	NQ	TriviaQA
Graph-Retriever (Min et al., 2019)	110M	34.7	55.8
KAQA (Zhou et al., 2020)	110M	-	66.6
UniK-QA (Oguz et al., 2020)*	990M	54.0	64.1
REALM (Guu et al., 2020)	330M	40.4	-
RAG (Lewis et al., 2020)	626M	44.5	56.1
Joint Top-k (Sachan et al., 2021a)	440M	49.2	64.8
EMDR^2 (Sachan et al., 2021b)	440M	52.5	71.4
R2-D2 (Fajcik et al., 2021)†	1.4B	**55.0**	69.9
UnitedQA (Cheng et al., 2021)†	2.1B	54.7	70.5
FiD (base) (Izacard & Grave, 2020a)	440M	48.2	65.0
FiD (large) (Izacard & Grave, 2020a)	990M	51.4	67.6
FiD-KD (base) (Izacard & Grave, 2020b)	440M	49.6	68.8
FiD-KD (large) (Izacard & Grave, 2020b)	990M	53.7	72.1
Our Implementation			
FiD (base)	440M	48.8	66.2
KG-FiD (base)	443M	49.6	66.7
FiD (large)	990M	51.9	68.7
KG-FiD (large)	994M	53.4	69.8
FiD-KD (base)	440M	50.0	68.5
KG-FiD-KD (base)	443M	50.8	69.0
FiD-KD (large)	990M	53.7	72.1
KG-FiD-KD (large)	994M	**54.2**	**72.3**

Table 1: Exact match score of different models over the test sets of NQ and TriviaQA datasets. $\star$ means that additional knowledge source like tables is used in this method. † stands for hybrid models where multiple readers are used. All the baseline results are directly taken from the original papers except the ones under Our Implementation.

Efficiency & Accuracy: Table 2 show the comparison between our method and FiD in the large version. The results of base version is shown in Appendix A.3. We see that for KG-FiD, decreasing L_1 can improve the computation efficiency as analyzed in Section 3.3, while increasing L_1 can improve the model performance. We think the performance improvement comes from the noise reduction of passage filtering. For a larger L_1, the passage embeddings for reranking will have a better quality so that the gold passages are less likely to be filtered out. It's important to note that our model can achieve the performance on par with FiD with only 40% of computation cost. When consuming the same amount of computations ($L_1 = 24$), our model significantly outperforms FiD on both NQ and TriviaQA datasets respectively. These experiments demonstrate that our model is very flexible and can improve both the efficiency and effectiveness by changing L_1.

4.4 Ablation Study

Since our proposed graph-based reranking method can be applied in both reading and retrieving modules: joint passage reranking and answer generation (Section 3.2) and graph based reranking after retrieving (Section 3.4). We conduct ablation study to validate the effectiveness of each one. For simplicity, we name the former one as Reader-Reranking and the latter one as Retriever-Reranking. Table 3 shows the experiment results by removing each module. We see the performance of KG-FiD drops when removing any of the two modules, demonstrating both of them can improve model performance. Another thing we observe is that retriever-reranking is more effective in base model while reader-reranking is more effective in large model. This is reasonable since reader-reranking relies on the effectiveness of reader encoder module, where large model is better than base model.

Effectiveness of KG: Table 4 shows the comparison results of using KG or not in the retriever-reranking and reader-reranking modules. If not using KG, we apply a Multi-layer Perceptron (MLP) as the reranking model instead of GNN. We see that in the retriever-reranking part, using KG can significantly improve the model performance. For reader-reranking part, we see that the effect of

Model	Computation Cost	NQ		TriviaQA	
		dev	test	dev	test
FiD (large)	100%	50.1	51.9	68.1	68.7
KG-FiD (large, L_1=6)	40%	50.0	52.0	68.5	68.9
KG-FiD (large, L_1=12)	60%	50.3	52.3	68.8	69.2
KG-FiD (large, L_1=18)	80%	50.9	52.6	69.1	69.8
KG-FiD (large, L_1=24)	100%	51.3	53.4	69.2	69.8

Table 2: Exact match score of FiD (large) and KG-FiD (large) with different computation cost.

Model	NQ		TriviaQA	
	base	large	base	large
FiD	48.8	51.9	66.2	68.7
KG-FiD	49.6	53.4	66.7	69.8
w/o Retriever-Reranking	49.3	53.1	66.2	69.5
w/o Reader-Reranking	49.4	52.3	66.5	69.2

Table 3: Ablation study of our graph-based reranking method in retriever and reader modules. EM scores are reported.

Model	NQ		TriviaQA	
	w/ KG	w/o KG	w/ KG	w/o KG
Retriever-Reranking	49.4	49.0	66.5	66.2
Reader-Reranking (base, L_1=3)	48.2	47.1	65.4	65.0
Reader-Reranking (base, L_1=6)	48.8	48.4	65.6	65.3
Reader-Reranking (base, L_1=9)	48.9	48.8	66.0	66.0
Reader-Reranking (base, L_1=12)	49.3	49.3	66.2	66.2

Table 4: Ablation study on the effectiveness of KG in our proposed reranking method in two modules. w/ KG refers to using GNN for passage reranking as our current model while w/o KG refers to using MLP instead of GNN. EM scores are reported.

Model	NQ				TriviaQA			
	H@10	H@20	H@50	H@100	H@10	H@20	H@50	H@100
DPR	74.8	79.2	83.7	86.3	76.3	79.7	83.2	85.2
w/ Retriever-Reranking	77.1	81.4	85.4	87.5	77.3	80.7	83.9	85.8

Table 5: Passage Retrieval Results. H@K is the Hits@K metric, measuring the percentage of top-K retrieved passages that contain the ground-truth passage.

KG varies with the choice of L_1, number of layers to compute passage representation for reranking. The larger improvement comes from the smaller L_1. This is reasonable since when L_1 is small, the quality of initial node embeddings generated by L_1 layers of encoder is sub-optimal, hence the graph structure information becomes more important. In summary, for the reader module, KG is more useful when we want to reduce computation cost by setting a small L_1.

Passage Reranking Results: We additionally show that our proposed GNN reranking method can improve the passage retrieval results. This is demonstrated in Table 5, where we report Hits@K metric (H@K), measuring the percentage of top-K retrieved passages that contain the gold passages (passages that contain the answer). We see that DPR w/ KG-Reranking consistently outperforms DPR for all the $K \in \{10, 20, 50, 100\}$, which shows that such reranking can increase the rank of gold passages which are previously ranked lower by DPR.

5 Conclusion

This work tackles the task of Open-Domain Question Answering. We focus on the current best performed reader model FiD and propose a novel KG-based reranking method to improve both its effectiveness and efficiency. Our reranking model reuses the passage representation generated by the reader encoder and apply graph neural networks to compute reranking scores. We propose to use the intermediate layer of encoder to reduce computation cost while maintaining good performance. We further show that our GNN based reranking method can also be applied to retriever to further boost model performance. Experiments on Natural Question and TriviaQA show that our model can significantly improve original FiD by 1.5% and achieve on-par performance with FiD but reducing 60% of computation cost.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1533–1544, 2013.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Unitedqa: A hybrid approach for open domain question answering. *arXiv preprint arXiv:2101.00178*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. Pruning the index contents for memory efficient open-domain qa. *arXiv preprint arXiv:2102.10697*, 2021.

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

Srinivasan Iyer, Sewon Min, Yashar Mehdad, and Wen-tau Yih. Reconsider: Improved re-ranking using span-focused cross-attention for open domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1280–1287, 2021.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020a.

Gautier Izacard and Edouard Grave. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*, 2020b.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

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

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. Reader-guided passage reranking for open-domain question answering. *arXiv preprint arXiv:2101.00294*, 2021.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*, 2019.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. Unified open-domain question answering with structured and unstructured knowledge. *arXiv preprint arXiv:2012.14610*, 2020.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5835–5847, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

Devendra Singh Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L Hamilton, and Bryan Catanzaro. End-to-end training of neural retrievers for open-domain question answering. *arXiv preprint arXiv:2101.00408*, 2021a.

Devendra Singh Sachan, Siva Reddy, William Hamilton, Chris Dyer, and Dani Yogatama. End-to-end training of multi-document reader and retriever for open-domain question answering. *arXiv preprint arXiv:2106.05346*, 2021b.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*, 2018.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*, 2019.

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

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage bert: A globally normalized bert model for open-domain question answering. *arXiv preprint arXiv:1908.08167*, 2019.

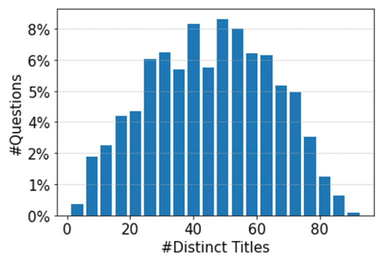
Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. Improving question answering over incomplete kbs with knowledge-aware reader. *arXiv preprint arXiv:1905.07098*, 2019.

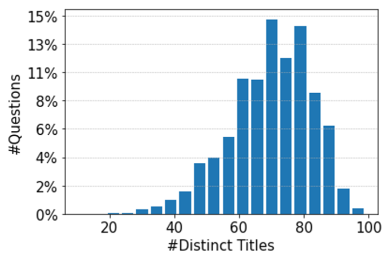
Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

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

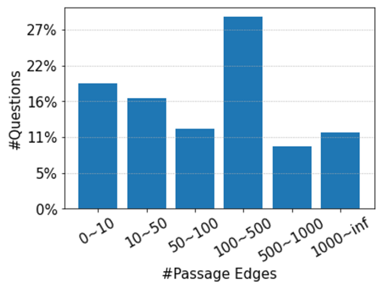
Mantong Zhou, Zhouxing Shi, Minlie Huang, and Xiaoyan Zhu. Knowledge-aided open-domain question answering. *arXiv preprint arXiv:2006.05244*, 2020.



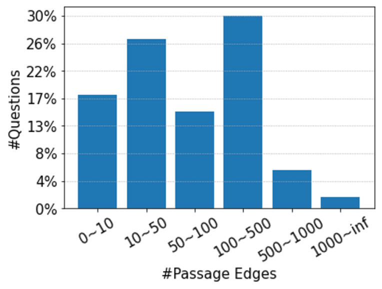
(a) Results on NQ



(b) Results on TriviaQA



(c) Results on NQ



(d) Results on TriviaQA

Figure 2: Preliminary Analysis on the retrieved passages by DPR.

A Appendix

A.1 Dataset

The datasets we use are Natural Questions (NQ) and TriviaQA. The open-domain version of NQ is obtained by discarding answers with more than 5 tokens. For TriviaQA, its *unfiltered* version is used for ODQA. We also convert all letters of answers in lowercase except the first letter of each word on TriviaQA. When training on NQ, we sample the answer target among the given list of answers, while for TriviaQA, we use the unique human-generated answer as generation target. For both datasets, we use the original validation data as test data, and keep 10% of the training set for validation.

A.2 Preliminary Analysis

We conduct preliminary analysis on the graph constructed among passages. Note that for each question, we first apply the retriever to retrieve a few candidate passages, then build edge connection only among the retrieved passages, which means that the passage graph is question-specific. Since the passage graph depends on the retrieved passages, before further utilizing the graph, we need avoid two trivia situations: (1) all the retrieved passages come from the same article (2) The number of graph edges is very small. Thus we conduct statistics of the passage graphs on two ODQA benchmark datasets, which is shown in Figure 2. For each question, the number of retrieved passages is 100. We see that the two trivia situations only happen for a small portion of questions.

A.3 Additional Experiment Results

We show additional experiment results in this section, which includes the efficiency and performance comparison between FiD (base) and KG-FiD (base) shown in Table 6, and hyper-parameter search results listed below:

GNN Model Design: We conduct search on the model type and number of layers of our GNN based reranking model. For efficiency, we rerank 100 passages returned by DPR retriever and search them based on the passage retrieval results. Table 8 shows the Hits scores for different choices. We see that GAT outperforms vanilla GCN model (Kipf & Welling, 2016) which is reasonable since GAT leverage attention to reweight neighbor passages by their embeddings. The best choice

Model	Computation Cost	NQ		TriviaQA	
		dev	test	dev	test
FiD (base)	100%	47.0	48.8	65.4	66.2
KG-FiD (base, L_1=3)	40%	46.7	48.4	64.9	65.6
KG-FiD (base, L_1=6)	60%	47.2	49.0	65.2	66.1
KG-FiD (base, L_1=9)	80%	47.4	49.3	65.7	66.3
KG-FiD (base, L_1=12)	100%	48.0	49.6	66.0	66.7

Table 6: Exact match score of FiD (large) and KG-FiD (large) with different computation cost.

Model	N_2=10	N_2=20	N_2=30
KG-FiD	47.6	48.0	48.0
	λ=0.01	λ=0.1	λ=1.0
KG-FiD	47.7	48.0	46.6

Table 7: EM scores on NQ dev data of our model under different choices of filtered passage numbers and weights of reranking loss.

Model	H@1	H@5	H@10	H@20
GCN	49.1	69.7	75.7	79.9
GAT	50.1	70.1	76.1	80.2
#Layers				
1	49.0	69.7	75.8	79.8
2	49.6	70.0	76.0	80.2
3	50.1	70.1	76.1	80.2
4	49.5	69.9	76.1	80.1

Table 8: Passage Retrieval Results on NQ dev data of our model under different GNN types and number of layers.

for the number of GNN layers is 3. Note that other GNN models such as GIN (Xu et al., 2018), DGI (Veličković et al., 2018) can also be applied here and we leave the further exploration of GNN models as future work.

N_2 **and** λ. For the reader-reranking part, we also conduct hyper-parameter search on the number of passages after filtering: $N_2 \in \{10, 20, 30\}$ and the weight of reranking loss when training the reading module: $\lambda \in \{0.01, 0.1, 1.0\}$. As shown in Table 7, $N_2 = 20$ achieves better results than $N_2 = 10$, but further increasing N_2 does not bring performance gain while decreasing the efficiency of model since the number of passages to be processed by the decoder is increased. Thus we choose $N_2 = 20$. For the loss weight λ, we found that with its increment, the performance first increases then significantly drops. This shows that it's important to balance the weight of two training losses, as we want the model to learn better passage reranking while not overwhelming the training signal of answer generation.