Exploiting Abstract Meaning Representation for Open-Domain Question Answering

Cunxiang Wang♠♣[*], Zhikun Xu♡, Qipeng Guo♢, Xiangkun Hu♢,

Xuefeng Bai♣, Zheng Zhang♢ and Yue Zhang♣[†]

♠Zhejiang University, China
♣School of Engineering, Westlake University, China
♡Fudan University, China; ♢Amazon AWS AI
{wangcunxiang, zhangyue}@westlake.edu.cn

Abstract

The Open-Domain Question Answering (ODQA) task involves retrieving and subsequently generating answers from fine-grained relevant passages within a database. Current systems leverage Pretrained Language Models (PLMs) to model the relationship between questions and passages. However, the diversity in surface form expressions can hinder the model's ability to capture accurate correlations, especially within complex contexts. Therefore, we utilize Abstract Meaning Representation (AMR) graphs to assist the model in understanding complex semantic information. We introduce a method known as Graph-as-Token (GST) to incorporate AMRs into PLMs. Results from Natural Questions (NQ) and TriviaQA (TQ) demonstrate that our GST method can significantly improve performance, resulting in up to 2.44/3.17 Exact Match score improvements on NQ/TQ respectively. Furthermore, our method enhances robustness and outperforms alternative Graph Neural Network (GNN) methods for integrating AMRs. To the best of our knowledge, we are the first to employ semantic graphs in ODQA. [1]

1 Introduction

Question Answering (QA) is a significant task in Natural Language Processing (NLP) (Rajpurkar et al., 2016). Open-domain QA (ODQA) (Chen et al., 2017), particularly, requires models to output a singular answer in response to a given question using a set of passages that can total in the millions. ODQA presents two technical challenges: the first is *retrieving* (Karpukhin et al., 2020) and *reranking* (Fajcik et al., 2021) relevant passages from the dataset, and the second is generating an answer for the question using the selected passages. In this work, we focus on the *reranking* and *reading* processes, which necessitate fine-grained interaction between the question and passages.

Existing work attempts to address these challenges using Pretrained Language Models (PLMs) (Glass et al., 2022). However, the diverse surface form expressions often make it challenging for the model to capture accurate correlations, especially when the context is lengthy and complex. We present an example from our experiments in Figure 1. In response to the question, the reranker incorrectly ranks a confusing passage first, and the reader generates the answer *"2015–16"*. The error arises from the PLMs' inability to effectively handle the complex semantic structure. Despite *"MVP"*, *"Stephen Curry"* and *"won the award"* appearing together, they are not semantically related. In contrast, in the AMR graph, it is clear that *"Stephen Curry"* wins over *"international players"*, not the *"MVP"*, which helps the model avoid the mistake. The baseline model may fail to associate "Most Valuable Player" in the passage with "MVP" in the question, which may be why the baseline does not rank it in the Top10. To address this issue, we adopt structured semantics (i.e., Abstract Meaning Representation (Banarescu et al., 2013) graphs shown on the right of Figure 1) to enhance Open-Domain QA.

While previous work has integrated graphs into neural models for NLP tasks, adding additional neural architectures to PLMs can be non-trivial, as training a graph network without compromising the original architecture of PLMs can be challenging (Ribeiro et al., 2021). Converting AMR graphs directly into text sequences and appending them can be natural, but leads to excessively long sequences, exceeding the maximum process-

[*] Done during the internship at Amazon AWS AI.
[†] The correponding author.
[1] We release our code and data at https://github.com/wangcunxiang/Graph-aS-Tokens

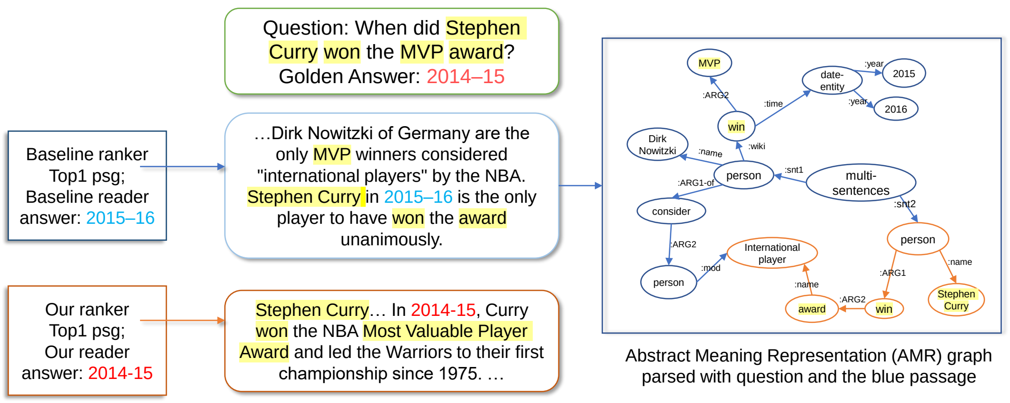

Question: When did Stephen Curry won the MVP award?
Golden Answer: 2014–15
Baseline ranker Top1 psg; Baseline reader answer: 2015–16
…Dirk Nowitzki of Germany are the only MVP winners considered "international players" by the NBA. Stephen Curry in 2015–16 is the only player to have won the award unanimously.
Our ranker Top1 psg; Our reader answer: 2014-15
Stephen Curry… In 2014-15, Curry won the NBA Most Valuable Player Award and led the Warriors to their first championship since 1975. …
MVP
:ARG2
win
:time
date-entity
:year
2015
:year
2016
Dirk Nowitzki
:name
:wiki
person
:snt1
multi-sentences
:ARG1-of
consider
:snt2
:ARG2
person
:mod
International player
person
:name
:ARG1
:name
award
:ARG2
win
Stephen Curry
Abstract Meaning Representation (AMR) graph parsed with question and the blue passage

Figure 1: An example from our experiments. The top-middle square contains the question and the gold standard answer. The middle section shows a confusing passage with an incorrect answer generated by the baseline model and ranked first by the baseline reranker. The bottom-middle section presents a passage with the gold standard answer, which is ranked within the top ten by our reranker but not by the baseline. Important information is highlighted.

ing length of the transformer. To integrate AMR into PLMs without altering the transformer architecture and at a manageable cost, we treat nodes and edges of AMR Graphs aS Tokens (GST) in PLMs. This is achieved by projecting the embeddings of each node/edge, which consist of multiple tokens, into a single token embedding and appending them to the textual sequence embeddings. This allows for integration into PLMs without altering the main model architecture. This method does not need to integrate a Graph Neural Network into the transformer architecture of PLMs, which is commonly used in integrating graph information into PLMs Yu et al. (2022); Ju et al. (2022). The GST method is inspired by Kim et al. (2022) in the graph learning domain, who uses token embeddings to represent nodes and edges for the transformer architecture in graph learning tasks. However, their method is not tailored for NLP tasks, does not consider the textual sequence embeddings, and only handles a certain types of nodes/edges, whereas we address unlimited types of nodes/edges consisting of various tokens.

Specifically, we select BART and FiD as baselines for the reranking and reading tasks, respectively. To integrate AMR information, we initially embed each question-passage pair into text embeddings. Next, we parse the pair into a single AMR graph using AMRBART (Bai et al., 2022a). We then employ the GST method to embed the graph nodes and graph edges into graph token embeddings and concatenate them with the text embeddings. Lastly, we feed the concatenated text-graph embeddings as the input embeddings to a BART-based (Lewis et al., 2020a) reranker to rerank or a FiD-based (Izacard and Grave, 2020b) reader to generate answers.

We validate the effectiveness of our GST approach using two datasets – Natural Question (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). Results indicate that AMR enhances the models' ability to understand complex semantics and improves robustness. BART-GST-reranker and FiD-GST outperform BART-reranker and FiD on the reranking and reading tasks, respectively, achieving up to 5.9 in Top5 scores, 3.4 in Top10 score improvements, and a 2.44 increase in Exact Match on NQ. When the test questions are paraphrased, models equipped with GST prove more robust than the baselines. Additionally, GST outperforms alternative GNN methods, such as Graph-transformer and Relational Graph Convolution Network (RGCN) (Schlichtkrull et al., 2018), for integrating AMR.

To the best of our knowledge, we are the first to incorporate semantic graphs into ODQA, thereby achieving better results than the baselines.

2 Related Work

Open-domain QA. Open-Domain Question Answering (ODQA) (Chen et al., 2017) aims to answer one factual question given a large-scale text database, such as Wikipedia. It consists of two steps. The first is *dense passage retrieval* (Karpukhin et al., 2020) , which retrieves a certain number of passages that match the question. In this process, a *reranking* step can be used to filter out the most matching passages (Fajcik et al., 2021; Glass et al., 2022). The second is *reading*, which finds answer by reading most matching passages (Izacard and Grave, 2020b; Lewis et al., 2020b). We focus on the reranking and reading, and integrate AMR into those models.

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a formalism for representing the semantics of a text as a rooted, directed graph. In this graph, where nodes represent basic semantic units such as entities and predicates, and edges represent the relationships between them. Compared with free-form natural language, AMR graphs are more semantically stable as sentences with same semantics but different expressions can be expressed as the same AMR graph (Bai et al., 2021; Naseem et al., 2021). In addition, AMR graphs are believed to have more structure semantic information than pure text (Naseem et al., 2021).

Previous work has implemented AMR graphs into neural network models. For example, (Bai et al., 2021) adopts Graph-transformer (Yun et al., 2019) to integrate AMRs into the transformer architecture for the dialogue understanding and generation. AMR-DA (Shou et al., 2022) uses AMRs as an data augmentation approach which first feeds the text into AMRs and regenerates the text from the AMRs. Bai et al. (2022b) uses AMR graphs with rich semantic information to redesign the pre-training tasks which results in improvement on downstream dialogue understanding tasks. However, none of them is used for Open-domain QA or applied with the GST technique. which does not need to implement extra architectures in the PLMs, avoiding the incompatibility of different model architectures.

Integrating Structures into PLMs for ODQA Some work also tries to integrate structure information into PLMs for ODQA. For example, GRAPE (Ju et al., 2022) insert a Relation-aware Graph Neural Network into the T5 encoders of FiD to encode knowledge graphs to enhance the output embeddings of encoders; KG-FiD (Yu et al., 2022) uses the knowledge graph to link different but correlated passages, reranks them before and during the reading, and only feeds the output embeddings of most correlated passages into the decoder. However, existing work concentrates on the knowledge graph as the source of structure information and no previous work has considered AMRs for ODQA.

LLMs in Open-Domain Question Answering (ODQA) Research has been conducted that utilizes pre-trained language models (PLMs) to directly answer open-domain questions without retrieval (Yu et al., 2023; Wang et al., 2021; Ye et al., 2021; Rosset et al., 2021). The results, however, have traditionally not been as effective as those achieved by the combined application of DPR and FiD. It was not until the emergence of ChatGPT that direct answer generation via internal parameters appeared to be a promising approach.

In a study conducted by Wang et al. (2023), the performances of Large Language Models (LLMs), such as ChatGPT (versions 3.5 and 4), GPT-3.5, and Bing Chat, were manually evaluated and compared with that of DPR+FiD across NQ and TQ test sets. The findings demonstrated that FiD surpassed ChatGPT-3.5 and GPT-3.5 on the NQ test set and outperformed GPT-3.5 on the TQ test set, affirming the relevance and effectiveness of the DPR+FiD approach even in the era of LLMs.

3 Method

We introduce the Retrieval and Reading of Open-Domain QA and their baselines in Section 3.1, AMR graph generation in Section 3.2 and our method Graph-aS-Token (GST) in Section 3.3.

3.1 Baseline

Retrieval. The retrieval model aims to retrieve N_1 passages from M reference passages ($N_1 << M$) given the question q. Only fast algorithms, such as BM25 and DPR (Karpukhin et al., 2020), can be used to retrieve from the large-scale database, and complex but accurate PLMs cannot be directly adopted. So, retrieval algorithm is often not very accurate. One commonly used method is applying a reranking process to fine-grain the retrieval results, and we can use PLMs to encode the correlations, which is usually more accurate. Formally, reranking requires model to sort out the most correlated N_2 passages with q from N_1 passages ($N_2 < N_1$). For each passage p in the retrieved passage P_{N_1}, we concatenate the q p together and embed them into text sequence embeddings $X_{qp} \in \mathbb{R}^{L\times H}$, where L is the max token length of the question and passage pair and H is the dimension.

We use a pretrained language model to encode each $\mathbf{X_{qp}}$ and a classification head to calculate a

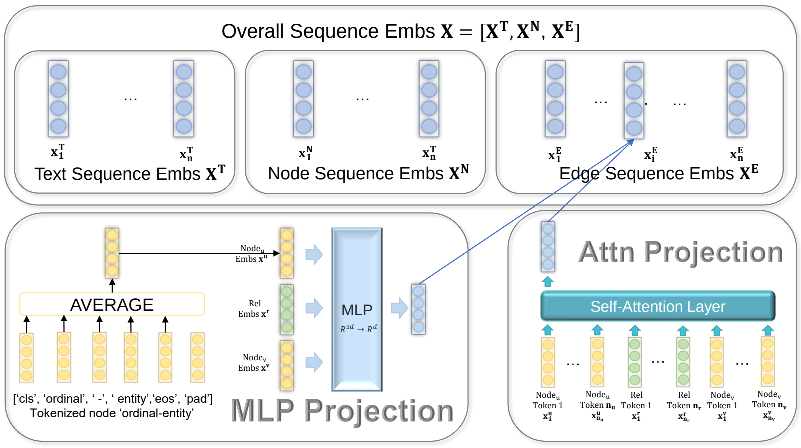

Overall Sequence Embs $\mathbf{X} = [\mathbf{X^T}, \mathbf{X^N}, \mathbf{X^E}]$
$\mathbf{x_1^T}$
$\mathbf{x_n^T}$
Text Sequence Embs $\mathbf{X^T}$
$\mathbf{x_1^N}$
$\mathbf{x_n^T}$
Node Sequence Embs $\mathbf{X^N}$
$\mathbf{x_1^E}$
$\mathbf{x_i^E}$
$\mathbf{x_n^E}$
Edge Sequence Embs $\mathbf{X^E}$
AVERAGE
['cls', 'ordinal', ' -', ' entity','eos', 'pad']
Tokenized node 'ordinal-entity'
Node$_u$ Embs $\mathbf{x^u}$
Rel Embs $\mathbf{x^r}$
Node$_v$ Embs $\mathbf{x^v}$
MLP
$R^{3d} \rightarrow R^d$
MLP Projection
Attn Projection
Self-Attention Layer
Node$_u$ Token 1 $\mathbf{x_1^u}$
Node$_u$ Token $\mathbf{n_u}$ $\mathbf{x_{n_u}^u}$
Rel Token 1 $\mathbf{x_1^r}$
Rel Token $\mathbf{n_r}$ $\mathbf{x_{n_r}^r}$
Node$_v$ Token 1 $\mathbf{x_1^v}$
Node$_v$ Token $\mathbf{n_v}$ $\mathbf{x_{n_v}^v}$

Figure 2: The structure of our Graph-aS-Token method. The input consists of the text and the AMR graph of one passage; The output is a united embedding.

correlation score between q and p:

$$s_{qp} = PLM(\mathbf{X_{qp}}) \tag{1}$$

where PLM denotes the pretrained language model and the commonly used Multi-Layer Perceptron (MLP) is used as as the classification head.

We use the cross entropy as the loss function,

$$\begin{aligned}\mathcal{L} &= \frac{1}{N_q}\sum_q [\frac{1}{N_{pos}+N_{neg}}\sum_p l_{qp}] \\ &= \frac{1}{N_q*(N_{pos}+N_{neg})}\sum_q\sum_p - \\ &[(y_{qp}*log(s_{qp})+(1-y_{qp})*log(1-s_{qp}))],\end{aligned} \tag{2}$$

where N_{pos} and N_{neg} are the numbers of positive and negative passages for training one question, respectively. To identify positive/negative label of each passage to the question, we follow Karpukhin et al. (2020), checking whether at least one answer appears in the passage.

We choose the N_2 passages which have reranked among Top-N_2 for the reading process.

Reading. The reader needs to generate an answer a given the question q and N_2 passages. In this work, we choose the Fusion-in-Decoder (FiD) model (Izacard and Grave, 2020b) as the baseline reader model. The FiD model uses N_2 separate T5 encoders (Raffel et al., 2020) to encode N_2 passages and concatenate the encoder hidden states to feed in one T5 decoder to generate answer.

Similar to reranking, we embed the question q and each passage p to text sequence embeddings $\mathbf{X_{qp}} \in \mathbb{R}^{L\times d_H}$, where L is the max token length of the question and passage pair and d_H is the dimension. Next, we feed the embeddings in the FiD model to generate the answer

$$a = FiD([\mathbf{X_{qp_1}}, \dots, \mathbf{X_{qp_i}}, \mathbf{X_{qp_{N_2}}}]) \tag{3}$$

where a is a text sequence.

3.2 AMR

We concatenate each question q and passage p, parse the result sequence into an AMR graph $G_{qp} = \{V, E\}$, where V, E are nodes and edges, respectively. Each edge is equipped with types, so $e = \{(u, r, v)\}$ where u, r, v represent the head node, relation and the tail node, respectively.

3.3 Graph aS Token (GST)

As shown in Figure 2, we project each node n or edge e in one AMR graph G into node embedding $\mathbf{x^n}$ or edge embedding $\mathbf{x^e}$. We adopt two types of methods to project each node and edge embeddings to one token embedding, which are MLP projection and Attention projection. After the projection, we append the node embeddings $\mathbf{X^N} = [\mathbf{x_1^n}, \dots, \mathbf{x_{n_n}^n}]$ and edge embeddings $\mathbf{X^E} = [\mathbf{x_1^e}, \dots, \mathbf{x_{n_e}^e}]$ to the corresponding text sequence embeddings $\mathbf{X^T} = [\mathbf{x_1^t}, \dots, \mathbf{x_{n_t}^t}]$. So, the result sequence embedding is in the following notation:

$$\mathbf{X} = [\mathbf{X^T}, \mathbf{X^N}, \mathbf{X^E}] \tag{4}$$

Initialization We explain how we initialize embeddings of nodes and edges here.

As each node n and relation r contain plural tokens (example of node 'ordinal-entity' is shown the left and bottom of Figure 2), $n = [t1, .., t_n]$ and $r = [t1, \ldots, t_r]$, and each edge e contains two nodes and one relation, we have $e = [[t1, .., t_u], [t1, \ldots, t_r], [t1, .., t_v]]$.

For edges and nodes, we first embed their internal tokens into token embedding.

For edges, we have

$$\begin{aligned} \mathbf{x^{e1}} =& [[\mathbf{x^u_1}, \ldots, \mathbf{x^u_{n_u}}], \\ & [\mathbf{x^r_1}, \ldots, \mathbf{x^r_{n_r}}], \\ & [\mathbf{x^v_1}, \ldots, \mathbf{x^v_{n_v}}]] \end{aligned} \tag{5}$$

For nodes, we have

$$\mathbf{x^{n1}} = [\mathbf{x^n_1}, \ldots, \mathbf{x^n_n}] \tag{6}$$

MLP Projection The process is illustrated in the MLP Projection part of Figure 2. As each AMR node can have more than one tokens, we first average its token embeddings. For example, for a head node u, $\mathbf{x^u} = AVE([\mathbf{x^u_1}, \ldots, \mathbf{x^u_{n_u}}]) \in \mathbb{R}^{d_H}$. The same is done for the relation.

Then, we concatenate the two node embeddings and one relation embedding together as the edge embedding,

$$\mathbf{x^{e2}} = [\mathbf{x^u}, \mathbf{x^r}, \mathbf{x^v}] \in \mathbb{R}^{3d_H} \tag{7}$$

Next, we use a $\mathbb{R}^{3d_H \times d_H}$ MLP layer to project the $\mathbf{x^{e2}} \in \mathbb{R}^{d_H}$ into $\mathbf{x^e} \in \mathbb{R}^{d_H}$, and the final edge embedding

$$\begin{aligned} \mathbf{x^e} &= MLP(\mathbf{x^{e2}}) \\ &= MLP([\mathbf{x^u}, \mathbf{x^r}, \mathbf{x^v}]) \end{aligned} \tag{8}$$

Similarly, we average the node tokens embeddings first $\mathbf{x^{n1}} = AVE([\mathbf{x^n_1}, \ldots, \mathbf{x^n_n}])$. To reuse the MLP layer, we copy the node embedding two times and concatenate, so, $\mathbf{x^{n2}} = [\mathbf{x^{n1}}, \mathbf{x^{n1}}, \mathbf{x^{n1}}] \in \mathbb{R}^{3d_H}$. Last, We adopt an MLP layer to obtain final node embedding

$$\mathbf{x^n} = MLP(\mathbf{x^{n2}}) \in \mathbb{R}^{d_H} \tag{9}$$

We have also tried to assign separate MLP layers to nodes and edges, but preliminary experiments show that it does not improve the results.

Attention Projection We use one-layer self-attention to project nodes and edges into embeddings, which is shown in the Attn Projection part in Figure 2. The edge embedding is calculated

$$\begin{aligned} \mathbf{x^e} = Att_E([&\mathbf{x^u_1}, \ldots, \mathbf{x^u_{n_u}}, \\ &\mathbf{x^r_1}, \ldots, \mathbf{x^r_{n_r}}, \mathbf{x^v_1}, \ldots, \mathbf{x^v_{n_v}}]) \end{aligned} \tag{10}$$

Similarly, the node embedding is calculated

$$\mathbf{x^n} = Att_N([\mathbf{x^n_1}, \ldots, \mathbf{x^n_n}]), \tag{11}$$

where Att_E and Att_N both denote one self-attention layer for edges and nodes, respectively. We take the first token (additional token) embedding from the self-attention output as the final embedding.

We only modify the input embeddings from $\mathbf{X} = \mathbf{X^T}$ to $\mathbf{X} = [\mathbf{X^T}, \mathbf{X^N}, \mathbf{X^E}]$. The rest details of models, such as the transformer architecture and the training paradigm, are kept the same with the baselines. Our model can directly use the PLMs to encode AMR graphs, without incompatibility between GNN's parameters and PLMs' parameters.

4 Experiments

4.1 Data

We choose two representative Open-Domain QA datasets, namely Natural Questions (NQ) and TriviaQA (TQ), for experiments. Data details are in presented in Appendix Table 9.

Since retrieval results have a large impact on the performance of downstream reranking and reading, we follow Izacard and Grave (2020b) and (Yu et al., 2022) to fix retrieval results for each experiment to make the reranking and reading results comparable for different models. In particular, we use the DPR model initialized with parameters in Izacard and Grave (2020a) [2] to retrieve 100 passages for each question. Then we rerank them into 10 passages, which means $N_1 = 100, N_2 = 10$.

We generate the amr graphs using AMRBART (Bai et al., 2022a) (the AMRBART-large-finetuned-AMR3.0-AMRParsing checkpoint) [3].

4.2 Models Details

We choose the BART model as the reranker baseline and the FiD model (implemented on T5 model(Raffel et al., 2020)) as the reader baseline, and adopt the GST method on them. For each model in this work, we use its Large checkpoint, such as BART-large and FiD-large, for reranking and reading, respectively. In the reranking process, we evaluate the model using the dev set per

[2]https://dl.fbaipublicfiles.com/FiD/pretrained_models/nq_retriever.tar.gz
https://dl.fbaipublicfiles.com/FiD/pretrained_models/tqa_retriever.tar.gz

[3]https://huggingface.co/xfbai/AMRBART-large-finetuned-AMR3.0-AMRParsing

Reranker + Reader \ Dataset	Natural Questions			TriviaQA		
	Reranking		Reading	Reranking		Reading
	Top5	Top10	EM	Top5	Top10	EM
w/o reranker + FiD-reader	73.7/74.6	79.5/80.3	49.47/50.66	78.0/78.1	81.5/81.8	69.02/69.50
w/o reranker + FiD-GST-A			50.12/51.11			70.17/70.39
w/o reranker + FiD-GST-M			50.06/50.97			69.98/70.10
BART-reranker + FiD-reader	78.7/78.6	83.0/83.3	50.33/51.33	83.2/83.2	85.2/85.1	71.16/71.33
BART-reranker + FiD-GST-A			50.80/52.38			71.93/72.05
BART-reranker + FiD-GST-M			50.76/52.24			72.12/72.24
BART-GST-A + FiD-reader	79.3/79.3	83.3/83.3	50.68/52.18	**83.5/83.3**	**85.3/85.3**	71.54/71.71
BART-GST-A + FiD-GST-A			51.05/52.80			**72.63/72.67**
BART-GST-M + FiD-reader	**79.6/80.0**	83.3/**83.7**	51.11/52.13	83.1/82.9	85.0/85.1	71.47/71.62
BART-GST-M + FiD-GST-M			**51.40/53.10**			72.58/72.61

Table 1: Reranking and reading results on the dev/test set of NQ and TQ. In each cell, the left is on the dev while the right is on the test. For the BART/FiD with GST-M/A in the first column, they are equipped AMR graphs with the GST method, -M indicates the MLP projection while -A is the attention projection.

Rearnker \ Dataset	Natural Questions		TriviaQA	
	MRR	MH@10	MRR	MH@10
w/o reranker	20.2/18.0	37.9/34.6	12.1/12.3	25.5/25.9
BART-reranker	25.7/23.3	49.3/45.8	16.9/17.0	37.7/38.0
BART-GST-A	28.1/24.7	52.7/48.2	**17.7/17.8**	**39.3/39.9**
BART-GST-M	**28.4/25.0**	**53.2/48.7**	17.5/17.6	39.1/39.5

Table 2: Overall reranking results on NQ and TQ. In each cell, the left is dev and the right is test.

epoch, and use Top10 as the pivot metric to select the best-performed checkpoint for the test. For the reading, we evaluate the model per 10000 steps, and use Exact Match as the pivot metric. For training rerankers, we set number of positive passages as 1 and number of negative passages as 7. We run experiments on 2 Tesla A100 80G GPUs.

4.3 Metric

Following Glass et al. (2022) and Izacard and Grave (2020b), we use Top-N to indicate the reranking performance and Exact Match for the reading performance.

However, TopN is unsuitable for indicating the overall reranking performance for all positive passages, so we also adopt two metrics, namely Mean Reciprocal Rank (MRR) and Mean Hits@10 (MHits@10). The MRR score is the Mean Reciprocal Rank of all positive passages. Higher scores indicate that the positive passages are ranked higher overall. The MHits@10 indicates the percentage of positive passages are ranked in Top10. Higher scores indicate that more positive passages are ranked in Top10. Their formulations are in Appendix Section A.5. Note that, only when the retrieved data is exactly the same, the MRR and MHits@10 metrics are comparable.

4.4 Preliminary Experiments

We present the reranking performance of four baseline PLMs, including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020) and BART (Lewis et al., 2020a) on the NQ and TQ in Appendix Table 8. BART outperforms other three models in every metric on both NQ and TQ. So, we choose it as the reranker baseline and apply our Graph-aS-Token method to it in following reranking experiments.

4.5 Main Results

The Main results are presented in Table 1. Our method can effectively boost the performance on both reranking and reading.

Reading. As shown in the reading columns of Table 1, our method can boost the FiD performance, no matter whether there is reranker and whether the reranker is with AMR or not. Without reranking, FiD-GST-A achieves 51.11/70.39 EM on NQ/TQ test , which are 0.45/0.89 EM higher over the baseline FiD; With reranking, 'BART-GST-M + FiD-GST-M ' achieves 53.10/72.61 EM on NQ/TQ test, 1.77/1.27 EM better than 'BART-reranker + FiD'. With the same reranker, FiD-GST is better than the baseline FiD, for example, 'BART-reranker + FiD-GST-A' achieves 52.38/72.05 on NQ/TQ test, which is 1.05/0.72 higher than the 51.33/71.33 of 'BART-reranker + FiD'.

Overall, our GST models have achieved up to

	Orig Test	New Test	Drop
BART-reranker	78.6/83.3 23.3/45.8	76.2/81.8 21.5/43.6	-2.6/-1.5 -1.8/-2.2
BART-GST-A	79.3/83.3 24.7/48.2	77.4/82.0 23.2/46.1	**-1.9**/-1.3 **-1.4/-2.1**
BART-GST-M	80.0/83.7 25.0/48.7	78.0/82.4 23.4/46.3	-2.0/-1.3 -1.6/-2.4

A: Robustness of rerankers. Each cell contains Top5/Top10/MRR/MHits@10 as the metrics.

	Orig Test	New Test	Drop
FiD-reader	50.66	46.76	-3.90
FiD-GST-A	51.11	47.84	-3.27
FiD-GST-M	50.97	47.76	**-3.21**

B: Robustness of readers. Exact Match as the Metric. To avoid the influence of different reranking results, we use the same DPR results to train and eval.

Table 3: Robustness on rerankers and readers. We conduct experiments on NQ. *Orig Test* is the original test questions while *New Test* means the paraphased test questions. *Drop* is the difference from the original test to the paraphrased test, the smaller absolute number indicates better robustness.

2.44 EM (53.10 vs 50.66) on NQ test and 3.17 (72.67 vs 69.50) on TQ test.

Reranking Shown in the reranking columns of Table 1, BART-GST-M can achieve 80.0/83.7 scores in Top5/Top10, which improve 5.4/3.4 on NQ-test compared to DPR and 1.4/0.4 compared to BART-reranker. BART-GST-M achieves 79.3/83.3 scores in Top5/Top10, which outperform DPR by 4.7/3.0 on NQ-test, showing that our GST method is effective.

We present results of the MRR and MHits@10 metrics in Table 2. Our GST method can help positive passages rank higher in Top10. In NQ, BART-GST-M has 7.0/14.1 advantages on MRR/MHits@10 over DPR while 1.7/2.9 advantages over BART-reranker; In TQ, BART-GST-A has 5.5/14.0 advantages on MRR/MHits@10 over DPR and 0.8/1.9 advantages on MRR, MHits@10 over BART-reranker.

The overall reranking results can also explain the reason why even when the Top10 results are similar and readers are the same, the reranked passages by BART-GST can lead to better reading performance. For example, in NQ test, the reading performance of 'BART-GST-M + FiD' is 0.80 better than 'BART-reranker + FiD'.

	NQ dev	NQ test	TQ dev	TQ test
FiD-10	49.47	50.66	69.02	69.50
FiD-100	**51.60**	52.88	71.61	71.88
FiD-10 w/ BART-reranker	50.33	51.33	71.16	71.33
FiD-GST-A-10 w/ BART-GST-A reranker	51.03	52.80	**72.63**	**72.67**
FiD-GST-M-10 w/ BART-GST-M reranker	51.30	**53.10**	72.58	72.61

Table 4: Reading experiments of with and without reranking. The first two row are trained/evaluated with DPR data while the rest are with reranked data.

4.6 Analysis

Robustness. To evaluate the robustness of the baseline and our models, we paraphrase the test questions of NQ and TQ, evaluate paraphrased test questions and the original ones with the same model checkpoint. We use a widely-used paraphraser, namely *Parrot Paraphraser* (Damodaran, 2021) to paraphrase test questions. The results are shown in Table 3.

The performance drops in reranking and reading of our GST models are lower than the baseline model, despite that our models have better performance. For reranking, the drop of our BART-GST-A is -1.9/-1.3/-1.4/-2.1 for Top5/Top10/MRR/MHits@10, which is lower than the baseline's -2.6/-1.5/-1.8/-2.2. For reading, the -3.21 EM drop of FiD-GST-M is also smaller than the -3.90 of baseline FiD. It shows that our GST method can not only improve performance but also improve robustness, which can prove that adding structural information can help models avoid the erroneous influence of sentence transformation.

Comparison with FiD-100. We also compare the reranking+reading paradigm with the directly-reading paradigm. For the latter, the FiD reader is directly trained and evaluated on 100 retrieved passages without reranking. The results are shown in Table 4.

Without our GST method, the reranking+reading paradigm (FiD-10 w/ BART reranker) is worse than FiD-100 without reranking, which is 71.33 to 71.78 on the test. However, with our GST method, the reranking+reading paradigm outperforms FiD-100. For example, FiD-GST-M-10 w/ BART-GST-M reranker has better performance on NQ test than FiD-100, which is 53.10 vs 52.88, and FiD-GST-A-10 w/ BART-GST-A reranker vs FiD-100 on TQ

	Top5	Top10	MRR	MH@10
BART-reranker	78.7/78.6	83.0/83.3	25.7/23.3	49.3/45.8
BART-GST-M (superior AMRs)	79.6/80.0	83.3/83.7	28.4/25.0	53.2/48.7
BART-GST-M (inferior AMRs)	79.5/79.3	83.5/83.1	28.4/24.7	52.9/47.8

In reranking.

	Exact Match
FiD-reader	48.47/50.66
FiD-GST-A (superior AMRs)	50.12/51.11
FiD-GST-A (inferior AMRs)	49.95/50.83

In reading.

Table 5: Influence of superior AMR graphs which generated by a larger model, and inferior AMR graphs which generated by a smaller model.

	Top5	Top10	MRR	MH@10
BART-reranker	78.7/78.6	83.0/83.3	25.7/23.3	49.3/45.8
BART-GST-M	79.6/80.0	83.3/83.7	28.4/25.0	53.2/48.7
BART-GST-M only nodes	78.5/78.9	82.9/83.1	27.6/24.2	51.8/47.3
BART-GST-M only edges	78.6/79.3	83.0/83.3	27.9/24.7	52.4/47.4

Table 6: Ablation to nodes and edges to our GST methods on NQ. We choose BART-GST-M because it better performs on NQ.

test is 72.67 vs 71.78.

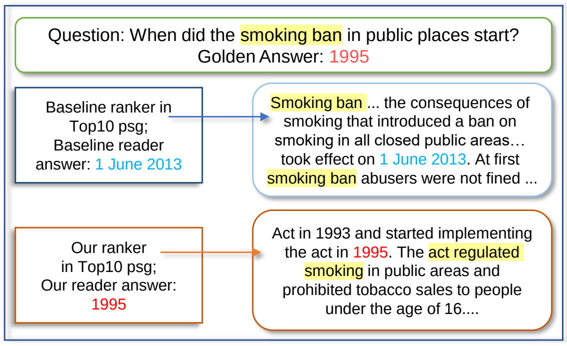
To our knowledge, we are the first make FiD-10 beat FiD-100.

Influence of AMR Quality. We explore how AMR graphs quality influence the performance of our models in this section, by using the AMRBART-base-finetuned-AMR3.0-AMRParsing, [4] which is a smaller version. We compare the reranking performance of BART-GST with either superior or inferior graphs on NQ and TQ. We use the each kind of graphs to train its own reranking models. The results are shown in Table 5.

Our models still work with inferior AMR graphs but the performance is not good as the superior ones in both reranking and reading. This indicates that when the quality of AMR graphs is higher, the GST models can potentially achieve better performance.

Ablation to Nodes/Edges We ablate nodes and edges in our models to explore whether nodes or edges contribute more to the results. We conduct reranking experiments on NQ. The results are shown in Table 6. As can be seen, nodes are edges are both useful for the GST method, where 'BART-GST-M (only nodes)' and 'BART-GST-M (only edges)' both outperform the baseline BART-reranker in MRR/MHits@10 on NQ test, which are 24.2/48.7 vs 24.7/47.4 vs 23.3/45.8, respectively. However, 'BART-GST-M (only edges)' are better than 'BART-GST-M (only nodes)' in four metrics on NQ, partly due to the fact that edges also contain nodes information.

[4]https://huggingface.co/xfbai/AMRBART-base-finetuned-AMR3.0-AMRParsing


Question: When did the smoking ban in public places start?
Golden Answer: 1995

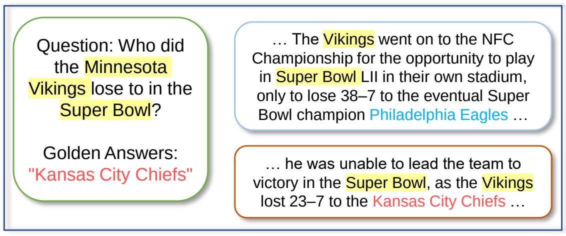
Baseline ranker in Top10 psg; Baseline reader answer: 1 June 2013

Smoking ban ... the consequences of smoking that introduced a ban on smoking in all closed public areas… took effect on 1 June 2013. At first smoking ban abusers were not fined ...

Our ranker in Top10 psg; Our reader answer: 1995

Act in 1993 and started implementing the act in 1995. The act regulated smoking in public areas and prohibited tobacco sales to people under the age of 16....

A: A case for reranking, where the baseline ranker does not rank the positive psg into Top10 while our model does.


Question: Who did the Minnesota Vikings lose to in the Super Bowl?

Golden Answers: "Kansas City Chiefs"

… The Vikings went on to the NFC Championship for the opportunity to play in Super Bowl LII in their own stadium, only to lose 38–7 to the eventual Super Bowl champion Philadelphia Eagles …

… he was unable to lead the team to victory in the Super Bowl, as the Vikings lost 23–7 to the Kansas City Chiefs …

B: A case for reading, where the negative and positive psgs are both ranked into Top10. The baseline reader finds the wrong psg to answer while our model answer correctly.

Figure 3: Two cases from our experiments for reranking and reading, respectively. We highlight important information over questions and passages.

Case Study We present two cases from our in Figure 3. In the upper one, for the negative passage, the baseline may consider *"a ban on smoking in all closed public areas"* same as *"the smoking ban in public places"*, which are actually different; For the positive passage, the baseline may not take *"act regulated smoking in public area"* as *"the smoking ban in public places"* while our model does.

In the lower one, the baseline reader ignores the competition is *" for the opportunity to play in Super Bow"* rather than *"in the Super Bowl"* , and because the number of similar passages with *"Philadelphia Eagle"* are more than the positive passage's, the baseline reader finds the incorrect passage which leads to the incorrect answer. In

	Top5	Top10	MRR	MH@10
BART-reranker	78.7/78.6	83.0/83.3	25.7/23.3	49.3/45.8
BART-GST-M	79.6/80.0	83.3/83.7	28.4/25.0	53.2/48.7
RGCN-Stacking	78.6/78.2	82.3/83.0	26.1/23.1	49.5/46.0

Table 7: Comparison between the baseline, GST and RGCN-Stacking in reranking on NQ.

contrast, our model focuses on the only positive passage and answers the question correctly.

4.7 Alternative Graph Methods

We have also tried several methods to integrate AMRs into PLMs, but their performance is worse than our Graph-aS-Token method. Here we take two representative examples, which are Relational Graph Convolution Network (RGCN) (Schlichtkrull et al., 2018) for the reranker and Graph-transformer (Yun et al., 2019) for FiD. All those methods require alignments between text tokens and graph nodes, for which only some nodes can be successfully aligned.

Stacking RGCN above Transformer The model architecture consists of a transformer encoder and a RGCN model where RGCN is stacked on top of the transformer. After the vanilla forward by transformer encoder, AMR graphs abstracted from queries and passages in advance are constructed with node embeddings initialized from transformer output. Then they are fed into the RGCN model and the final output of the [CLS] node is used for scoring.

For the text embeddings of one question-passage pair, its encoder hidden states

$$\mathbf{H} = Encoder(X_{qp})$$

For one node n, its initial embedding

$$\mathbf{h^0} = MeanPooling(\mathbf{H_{start:end}})$$

where $start$ and end are the start and end positions of the text span aligned with the node.

The update of node embedding for each layer l is

$$\mathbf{h_i^{l+1}} = \sigma(W_0^l \mathbf{h_i^l} + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^l \mathbf{h_i^l})$$

$$c_{i,r} = \|N_i^r\|$$

where R is the set of edge types, N_i^r stands for the group of nodes which connect with node i in relation r.

so the correlation score of q and p:

$$s_{qp} = ClsHead(h^L_{[CLS]})$$

The results are presented in Table 7, which is clear that the RGCN-stacking method is inferior to the GST method. Some metrics, including Top5, Top10 and MRR, of RGCN-stacking are worse than the baseline, meaning the RGCN method is not feasible for integrating AMRs into PLMs though it looks like reasonable and practical.

Graph-transformer We apply the graph-transformer architecture to FiD model for reading. We follow the graph-transformer architecture in Bai et al. (2021), whose main idea is using AMR information to modify the self-attention scores between text tokens. However, we find stucking challenging for PLMs because the new-initialized graph architectures are not compatible with architectures of PLMs, lead to non-convergence during training. Despite that, tricks such as incrementally training and separate tuning can lead to convergence, results are still below the baseline model, let alone GST.

Flattening AMR Graphs We have also tried to directly flatten AMR graphs into text sequences, but the result sequences are always beyond the maximum processing length (1024) of the transformer. So, we have to cut off some nodes and edges to fit in the transformer, but the results show that it does not work well and has only a very sight improvement while the computational cost is tens times over the baseline.

5 Conclusion

In this study, we successfully incorporated Abstract Meaning Representation (AMR) into Open-Domain Question Answering (ODQA) by innovatively employing a Graph-aS-Token (GST) method to assimilate AMRs with pretrained language models. The reranking and reading experiments conducted on the Natural Questions and TriviaQA datasets have demonstrated that our novel approach can notably enhance the performance and resilience of Pretrained Language Models (PLMs) within the realm of ODQA.

Acknowledgement

This publication has emanated from research conducted with the financial support of the Pioneer and

“Leading Goose" R&D Program of Zhejiang under Grant Number 2022SDXHDX0003.

Limitations

Our Graph-aS-Token (GST) method can increase the time and GPU memory cost, we set an quantitative analysis in Appendix Section A.4. We train the models with only one random seed. We do not conduct a large number of hyper-parameter tuning experiments, but use a fixed set of hyper-parameters to make the baseline and our models comparable.

Ethics Statement

No consideration.

References

Xuefeng Bai, Yulong Chen, Linfeng Song, and Yue Zhang. 2021. Semantic representation for dialogue modeling. 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 4430–4445, Online. Association for Computational Linguistics.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022a. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Xuefeng Bai, Linfeng Song, and Yue Zhang. 2022b. Semantic-based pre-training for dialogue understanding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 592–607, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Pre-training transformers as energy-based cloze models. In *EMNLP*.

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu.

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

Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. 2021. R2-D2: A modular baseline for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 854–870, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2G: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715, Seattle, United States. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2020a. Distilling knowledge from reader to retriever for question answering.

Gautier Izacard and Edouard Grave. 2020b. Leveraging passage retrieval with generative models for open domain question answering.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Mingxuan Ju, Wenhao Yu, Tong Zhao, Chuxu Zhang, and Yanfang Ye. 2022. Grape: Knowledge graph enhanced passage reader for open-domain question answering. In *Findings of Empirical Methods in Natural Language Processing*.

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 (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure transformers are powerful graph learners. *ArXiv*, abs/2207.02505.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

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. *arXiv preprint arXiv:1907.11692*.

Tahira Naseem, Austin Blodgett, Sadhana Kumaravel, Timothy J. O'Gorman, Young-Suk Lee, Jeffrey Flanigan, Ramón Fernández Astudillo, Radu Florian, Salim Roukos, and Nathan Schneider. 2021. Docamr: Multi-sentence amr representation and evaluation. In *North American Chapter of the Association for Computational Linguistics*.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Corbin L Rosset, Chenyan Xiong, Minh Phan, Xia Song, Paul N. Bennett, and saurabh tiwary. 2021. Pretrain knowledge-aware language models.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Ziyi Shou, Yuxin Jiang, and Fangzhen Lin. 2022. AMR-DA: Data augmentation by Abstract Meaning Representation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3082–3098, Dublin, Ireland. Association for Computational Linguistics.

Cunxiang Wang, Sirui Cheng, Zhikun Xu, Bowen Ding, Yidong Wang, and Yue Zhang. 2023. Evaluating open question answering evaluation.

Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book QA? 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 3241–3251, Online. Association for Computational Linguistics.

Qinyuan Ye, Belinda Z. Li, Sinong Wang, Benjamin Bolte, Hao Ma, Wen tau Yih, Xiang Ren, and Madian Khabsa. 2021. Studying strategically: Learning to mask for closed-book qa.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022. KG-FiD: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4961–4974, Dublin, Ireland. Association for Computational Linguistics.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *International Conference for Learning Representation (ICLR)*.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

A Experimental Details

A.1 Pre-experiment

A.2 Details for Data

For each question and passage pair, we feed it in the generator in such a format "Question: <question>. Title: <Passage Title>. Context: <Passage

	Top5	Top10	MRR	MH@10
w/o reranker	73.7/74.6	79.5/80.3	20.2/18.0	37.9/34.6
BERT	76.5/75.7	81.5/81.4	23.7/20.9	45.5/41.5
RoBERTa	77.1/76.6	82.3/82.3	24.7/21.5	47.7/43.3
ELECTRA	77.3/77.8	82.4/82.5	25.1/22.5	47.9/43.9
BART	**78.7/78.6**	**83.0/83.3**	**25.7/23.3**	**49.3/45.8**

A: On the Natural Questions dataset.

	Top5	Top10	MRR	MH@10
w/o reranker	78.0/78.1	81.5/81.8	12.1/12.3	25.5/25.9
BERT	82.0/82.3	84.5/84.7	16.0/16.2	35.6/35.9
RoBERTa	82.8/82.9	85.0/85.0	16.8/16.8	37.2/37.4
ELECTRA	82.4/82.6	84.8/82.6	16.3/16.4	36.2/36.4
BART	**83.2/83.1**	**85.2/85.1**	**16.9/17.0**	**37.7/38.0**

B: On the TriviaQA dataset.

Table 8: Pre-experiments of four PLMs' reranking performance on NQ and TQ. In each cell, the left is on the dev while the right is on the test. Among four PLMs, BART performs best.

	Train Set	**Dev Set**	**Test Set**
Natural Questions	79168	8757	3610
TriviaQA	78785	8837	11313

Table 9: Details of each dataset.

Context>". Additionally, we link the nodes, which are recognized as entities such as person name and date and have same surfaces, with the ":same" relation because it helps performance. For nodes in one AMR graph, we remove their '-XX', where X is a 0-9 number.

A.3 Hyper-parameters

We set other model-related hyper-parameters in Table 10.

A.4 Cost Increase

We conduct an experiment of the increase of time and GPU memory cost on our GST compared with the baseline. For inference, while keeping other parameters as the same, the time costs of FiD-GST-M, FiD-GST-A are 1.29x and 1.40x, respectively, and the GPU memory costs are 1.11x and 1.40x, respectively, compared with FiD, as shown in Table 11.

A.5 Metrics

$$MRR = \frac{1}{|Q|}\sum_{i\in Q}((\sum_{j\in Pos}\frac{1}{t(j)})\frac{1}{num_{Pos}(i)})$$

	Reranking	**Reading**
Leaning Rate	3e-5	1e-4
Training Epoch	10	5
Node MaxLength	145	145
Edge MaxLength	165	165
Text Maxlength	200	200
Eval Step/Epoch	10k steps	1 epoch

Table 10: Hyper-parameters Setting

	Time cost	GPU Memory Cost
FiD	1.00	1.00
FiD-GST-M	1.29	1.11
FiD-GST-M	1.40	1.40

Table 11: The results of time and GPU memory cost comparing our GST method and the baseline. The experiment is inference on the NQ test set. We take the baseline FiD model cost as 1.00.

where Q is the evaluating dataset; $t(j)$ is the rank of passage j; Pos is the set of positive passages.

$$MHits@10 = \frac{1}{|Q|}\sum_{i\in Q}(\sum_{j\in pos, t(j)<=10}\frac{1}{num_{Pos}(i)})$$

where Q is the evaluating dataset; $t(j)$ is the rank of passage j; Pos is the set of positive passages.