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
♢ 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 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 (Fajjik 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-
When did Stephen Curry win the MVP award?

Golden Answer: 2014–15

Stephen Curry… In 2014-15, Curry won the MVP award unanimously.
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 $X_{qp}$ and a classification head to calculate a
correlation score between $q$ and $p$:

$$s_{qp} = PLM(X_{qp})$$  \hspace{1cm} (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,

$$L = \frac{1}{N_q} \sum_q \left[ \frac{1}{N_{pos} + N_{neg}} \sum_p l_{qp} \right]$$

$$= \frac{1}{N_q} \left( N_{pos} + N_{neg} \right) \sum_q \sum_p - \left[ (y_{qp} \ast log(s_{qp}) + (1 - y_{qp}) \ast log(1 - s_{qp})) \right],$$  \hspace{1cm} (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 $X_{qp}$ \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

$$a = FiD([X_{qp1}, \ldots, X_{qpn}, X_{qpn2}])$$  \hspace{1cm} (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 $x^n$ or edge embedding $x^e$. 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 $X^N = [x^n_1, \ldots, x^n_{n_2}]$ and edge embeddings $X^E = [x^e_1, \ldots, x^e_{e_2}]$ to the corresponding text sequence embeddings $X^T = [x^t_1, \ldots, x^t_{n_2}]$. So, the result sequence embedding is in the following notation:

$$X = [X^T, X^N, X^E]$$  \hspace{1cm} (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), $n = [t_1, \ldots, t_n]$ and $r = [t_1, \ldots, t_r]$, and each edge $e$ contains two nodes and one relation, we have $e = [[t_1, \ldots, t_u], [t_1, \ldots, t_r], [t_1, \ldots, t_v]]$.

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

For edges, we have
\[
x^e = \left[ [x^u_1, \ldots, x^u_{n_u}], [x^r_1, \ldots, x^r_{n_r}], [x^v_1, \ldots, x^v_{n_v}] \right]
\] (5)

For nodes, we have
\[
x^n = [x^n_1, \ldots, x^n_n]
\] (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 $u$, $x^u = \text{AVE}([x^u_1, \ldots, x^u_{n_u}]) \in \mathbb{R}^{d_H}$. The same is done for the relation.

Then, we concatenate the two node embeddings and one relation embedding together as the edge embedding,
\[
x^e = [x^u, x^r, x^v] \in \mathbb{R}^{3d_H}
\] (7)

Next, we use a $\mathbb{R}^{3d_H \times d_H}$ MLP layer to project the $x^e \in \mathbb{R}^{d_H}$ into $x^e \in \mathbb{R}^{d_H}$, and the final edge embedding
\[
x^e = MLP(x^e^2)
\] (8)

Similarly, we average the node tokens embeddings first $x^n = \text{AVE}([x^n_1, \ldots, x^n_n])$. To reuse the MLP layer, we copy the node embedding two times and concatenate, so, $x^{n2} = [x^n_1, x^n_1, x^n_1] \in \mathbb{R}^{3d_H}$. Last, We adopt an MLP layer to obtain final node embedding
\[
x^n = MLP(x^{n2}) \in \mathbb{R}^{d_H}
\] (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
\[
x^e = \text{Att}_E([x^n_1, \ldots, x^n_{n_u}], [x^1_1, \ldots, x^1_{n_u}, x^1_1, \ldots, x^1_{n_v}])
\] (10)

Similarly, the node embedding is calculated
\[
x^n = \text{Att}_N([x^n_1, \ldots, x^n_{n}],
\] (11)

where $\text{Att}_E$ and $\text{Att}_N$ 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 $X = X^T$ to $X = [X^T, X^N, X^E]$. 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) 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

\[\text{https://dl.fbaipublicfiles.com/FiD/pretrained_models/nq_retriever.tar.gz}\]

\[\text{https://dl.fbaipublicfiles.com/FiD/pretrained_models/tqa_retriever.tar.gz}\]

\[\text{https://huggingface.co/xfbai/AMRBART-large-finetuned-AMR3.0-AMRParsing}\]
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.

<table>
<thead>
<tr>
<th>Reranker + Reader \ Dataset</th>
<th>Natural Questions</th>
<th>TriviaQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reranking</td>
<td>Reading</td>
</tr>
<tr>
<td></td>
<td>Top5</td>
<td>Top10</td>
</tr>
<tr>
<td>w/o reranker + FiD-reader</td>
<td>73.7/74.6</td>
<td>79.5/80.3</td>
</tr>
<tr>
<td>w/o reranker + FiD-GST-A</td>
<td>78.7/78.6</td>
<td>83.0/83.3</td>
</tr>
<tr>
<td>w/o reranker + FiD-GST-M</td>
<td>79.9/79.3</td>
<td>83.3/83.3</td>
</tr>
<tr>
<td>BART-reranker + FiD-reader</td>
<td>78.7/78.6</td>
<td>83.0/83.3</td>
</tr>
<tr>
<td>BART-reranker + FiD-GST-A</td>
<td>79.4/79.3</td>
<td>83.3/83.3</td>
</tr>
<tr>
<td>BART-GST-A + FiD-reader</td>
<td>79.6/80.0</td>
<td>83.3/83.7</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Reranker \ Dataset</th>
<th>Natural Questions</th>
<th>TriviaQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>MH@10</td>
</tr>
<tr>
<td>w/o reranker</td>
<td>20.2/18.0</td>
<td>37.9/34.6</td>
</tr>
<tr>
<td>BART-reranker</td>
<td>25.7/23.3</td>
<td>49.3/45.8</td>
</tr>
<tr>
<td>BART-GST-A</td>
<td>28.1/24.7</td>
<td>52.7/48.2</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th></th>
<th>Orig Test</th>
<th>New Test</th>
<th>Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART-reranker</td>
<td>78.6/83.3</td>
<td>76.2/81.8</td>
<td>-2.6/-1.5</td>
</tr>
<tr>
<td></td>
<td>23.3/45.8</td>
<td>21.5/43.6</td>
<td>-1.8/-2.2</td>
</tr>
<tr>
<td>BART-GST-A</td>
<td>79.3/83.3</td>
<td>77.4/82.0</td>
<td>-1.9/-1.3</td>
</tr>
<tr>
<td></td>
<td>24.7/48.2</td>
<td>23.2/46.1</td>
<td>-1.4/-2.1</td>
</tr>
<tr>
<td>BART-GST-M</td>
<td>80.0/83.7</td>
<td>78.0/82.4</td>
<td>-2.0/-1.3</td>
</tr>
<tr>
<td></td>
<td>25.0/49.7</td>
<td>23.4/46.3</td>
<td>-1.6/-2.4</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Orig Test</th>
<th>New Test</th>
<th>Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiD-reader</td>
<td>50.66</td>
<td>46.76</td>
<td>-3.90</td>
</tr>
<tr>
<td>FiD-GST-A</td>
<td>51.11</td>
<td>47.84</td>
<td>-3.27</td>
</tr>
<tr>
<td>FiD-GST-M</td>
<td>50.97</td>
<td>47.76</td>
<td>-3.21</td>
</tr>
</tbody>
</table>

**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 paraphrased 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’.

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

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

In reranking.

<table>
<thead>
<tr>
<th></th>
<th>Exact Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiD-reader</td>
<td>48.47/50.66</td>
</tr>
</tbody>
</table>

FiD-GST-A (superior AMRs) | 50.12/51.11 |

FiD-GST-A (inferior AMRs) | 49.95/50.83 |

In reading.

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

<table>
<thead>
<tr>
<th></th>
<th>Top5</th>
<th>Top10</th>
<th>MRR</th>
<th>MH@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART-reranker</td>
<td>78.7/78.6</td>
<td>83.0/83.3</td>
<td>25.7/23.3</td>
<td>49.3/45.8</td>
</tr>
<tr>
<td>BART-GST-M only nodes</td>
<td>78.5/78.9</td>
<td>82.9/83.1</td>
<td>27.6/24.2</td>
<td>51.8/47.3</td>
</tr>
<tr>
<td>BART-GST-M only edges</td>
<td>78.6/79.3</td>
<td>83.0/83.3</td>
<td>27.9/24.7</td>
<td>52.4/47.4</td>
</tr>
</tbody>
</table>

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, 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.

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

<table>
<thead>
<tr>
<th></th>
<th>Top5</th>
<th>Top10</th>
<th>MRR</th>
<th>MH@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART-reranker</td>
<td>78.7/68.6</td>
<td>83.0/83.3</td>
<td>25.7/23.3</td>
<td>49.3/45.8</td>
</tr>
<tr>
<td>BART-GST-M</td>
<td>79.6/80.0</td>
<td>83.3/83.7</td>
<td>28.4/25.0</td>
<td>53.2/48.7</td>
</tr>
<tr>
<td>RGCN-Stacking</td>
<td>78.6/78.2</td>
<td>82.3/83.0</td>
<td>26.1/23.1</td>
<td>49.5/46.0</td>
</tr>
</tbody>
</table>

In 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

\[ H = \text{Encoder}(X_{qp}) \]

For one node \( n \), its initial embedding

\[ h^0 = \text{MeanPooling}(H_{\text{start:end}}) \]

where \( \text{start} \) and \( \text{end} \) are the start and end positions of the text span aligned with the node.

The update of node embedding for each layer \( l \) is

\[ h_{l+1} = \sigma(W_0^l h_l^l + \sum_{r \in R} \sum_{j \in N_r^i} \frac{1}{c_{i,r}} W_r^j h_r^j) \]

\[ c_{i,r} = \|N_r^i\| \]

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

so the correlation score of \( q \) and \( p \):

\[ s_{qp} = \text{ClsHead}(h_{[CLS]}^L) \]

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 slight 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


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


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.


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


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: <PassageTitle>. Context: <Passage
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.

Table 9: Details of each dataset.

Contexts”. 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

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

Table 10: Hyper-parameters Setting

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.
ACL 2023 Responsible NLP Checklist

A For every submission:

✓ A1. Did you describe the limitations of your work?  
  Limitations section

✓ A2. Did you discuss any potential risks of your work?  
  No potential risks

✓ A3. Do the abstract and introduction summarize the paper’s main claims?  
  Abstract and Introduction sections

✗ A4. Have you used AI writing assistants when working on this paper?  
  Left blank.

B ✓ Did you use or create scientific artifacts?

  Section 4

✓ B1. Did you cite the creators of artifacts you used?  
  Section 4

✓ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?  
  Section 4

✓ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?  
  Section 4

✗ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?  
  Not applicable. Left blank.

✗ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?  
  Not applicable. Left blank.

✓ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.  
  Section 4

C ✓ Did you run computational experiments?

  Section 4

✓ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?  
  Section 4

---

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
   Section 4 and appendix

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
   Section 4

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
   Not applicable. Left blank.

D Did you use human annotators (e.g., crowdworkers) or research with human participants?
   Left blank.

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
   No response.

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
   No response.

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
   No response.

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
   No response.

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
   No response.