SemPool: Simple, robust, and interpretable KG pooling for enhancing language models

Costas Mavromatis[1], Petros Karypis[2], and George Karypis[1]

[1] University of Minnesota
{mavro016, karypis}@umn.edu
[2] University of California San Diego
pkarypis@ucsd.edu

Abstract. Knowledge Graph (KG) powered question answering (QA) performs complex reasoning over language semantics as well as knowledge facts. Graph Neural Networks (GNNs) learn to aggregate information from the underlying KG, which is combined with Language Models (LMs) for effective reasoning with the given question. However, GNN-based methods for QA rely on the graph information of the candidate answer nodes, which limits their effectiveness in more challenging settings where critical answer information is not included in the KG. We propose a simple graph pooling approach that learns useful semantics of the KG that can aid the LM's reasoning and that its effectiveness is robust under graph perturbations. Our method, termed SemPool, represents KG facts with pre-trained LMs, learns to aggregate their semantic information, and fuses it at different layers of the LM. Our experimental results show that SemPool outperforms state-of-the-art GNN-based methods by 2.27% accuracy points on average when answer information is missing from the KG. In addition, SemPool offers interpretability on what type of graph information is fused at different LM layers.

1 Introduction

Question answering (QA) is a complex reasoning task that requires understanding of a given natural language query, as well as domain-specific knowledge. For instance, answering biomedical questions requires understanding of biomedical terms as well as knowledge about biomedicine. Language models (LMs) [3,22] are pre-trained on large corpora to understand their underlying semantics. Thus, fine-tuning LMs for the given reasoning tasks [14,9] is the dominant paradigm in NLP for QA.

Despite their success, LMs struggle on intensive reasoning tasks that require good in-domain knowledge [15]. As a result, recent methods incorporate Knowledge Graphs (KGs) during the QA task [19,4], which are graphs that capture factual knowledge explicitly as triplets. Each triplet consists of two entities and their corresponding relation. Most successful KG-based methods [34] leverage Graph Neural Networks (GNNs) [28], which have shown remarkable performance at reasoning tasks with graph information [17,37].

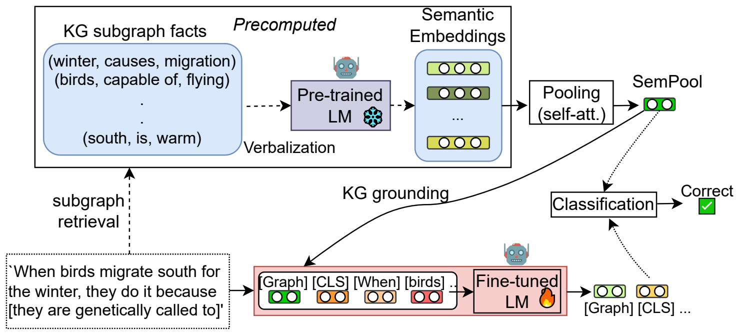


Fig. 1: Our SemPool method performs simple graph pooling to enhance the LM's reasoning. Facts of the KG are represented by their semantic information with pre-trained LMs. SemPool aggregates the graph's semantic information into a single representation that is fed into the LM for QA.

Nevertheless, GNNs operate on graph data while LMs use natural language sequences, which makes information exchange between the two modalities challenging. In fact, our empirical findings (Section 4) suggest that GNNs mainly provide graph statistical information for the QA task [29] rather than information that grounds the LM's reasoning and is robust under graph perturbations. In addition, the representation space mismatch between graph (KGs are usually represented with external node embeddings) and language (represented with pre-trained LMs) does not aid the information exchange between the two modalities.

In this work, we present SemPool, a simple graph pooling method that enhances the LM's reasoning with KG textual information. As illustrated in Figure 1, SemPool represents each fact in the KG with the pre-trained LM, aiming at semantic alignment between graph and language. SemPool then performs a global graph pooling operation in order to aggregate semantic information from the whole graph into a single representation. The aggregated representation is fused as input into the fine-tuned LM for QA, which grounds the LM's reasoning to the information provided. Moreover, we extend SemPool to fuse different type of semantic information into different LM's layers (Section 5.3), providing more flexibility during learning.

SemPool demonstrates robust performance under different settings. We experiment with standard QA benchmarks (OpenbookQA, RiddleSense, MedQA-USMLE), (i) when complemented by complete in-domain KGs, and (ii) when complemented by in-domain KGs where critical information about the candidate answers is missing. SemPool outperforms the *best* performing GNN-based approach by 2.27% accuracy points in the challenging case, while it is competitive (second-best) in the easier case. In addition, our experiments show that SemPool is effective under different LMs (Section 7.1), highlight the importance of semantic alignment between language and graph, and illustrate SemPool's interpretability (Section 7.2).

2 Related Work

Question Answering with KGs. Many KGs have been employed to improve QA for different domains, such as ConceptNet [24] for commonsense QA. Graph neural networks [28,23] have been widely used to combine KG information with language models [10,4,34,35,25,26] leading to SOTA QA systems. In this work, we give new insights on the sensitivity of GNN-based methods with respect to the provided graph and propose a simple approach that improves robustness for QA. Other methods have explored to provide the verbalization of the retrieved KG facts [1,31] or their embeddings [21] as input sequences to the LM, which, however, considerably increases the inference cost due to the extended context. SemPool integrates graph information as a special input token to the LM, offering low computational cost.

Graph-augmented LMs. Combining LMs with graphs that include textual features is an emerging research area [20,13]. Recent methods have explored fine-tuning LMs on graph data [16,36] as well as aiding the pre-training of LMs with graph information [33,7,32,30]. SemPool provides a new simple approach to fuse graph information into the LM, easily integrated to existing approaches.

3 Problem Statement & Preliminaries

Multi-choice QA. We study the problem of multiple-choice question answering (QA), where given an optional context c, a question q, and a set of candidate answers $\mathcal{A}$, the goal is to select the correct answer $a^* \in \mathcal{A}$. Multiple-choice QA is transformed into a classification problem by (i) concatenating the question's context with each of the candidate answer $a \in \mathcal{A}$ into a statement q_a, e.g., $q_a = [c, q, a] =$ "*When birds migrate south for the winter, they do it because (A) they are genetically called to*", and (ii) selecting the most probable statement from $\{[c, q, a] : a \in \mathcal{A}\}$. Given each input $[c, q, a]$, a fine-tuned LM is used to determine whether the textual input is plausible. The output token representations $\boldsymbol{q}^{(L)} = \mathrm{LM}([c, q, a])$ are used for classifying the statement as the correct one, usually followed by a pooling operation. We point the readers for more details to the corresponding papers [3,14].

Knowledge Graphs (KGs). Knowledge Graph (KG) powered QA aims at leveraging external factual information from a KG in order to improve the LM's reasoning ability. For instance, the KG might contain the factual information (*winter, causes, bird migration*), which is relevant for the question $q =$ "*When birds migrate south for the winter, they do it because?*". Formally, KG is a multi-relational graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ that contains a set nodes (entities) $\mathcal{V}$ and a set of edges (facts) $\mathcal{E}$. Set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ contains facts in the tuple form (h, r, t), where $h, t \in \mathcal{V}$ and $r \in \mathcal{R}$ is the relation between nodes h and t ($\mathcal{R}$ denotes the relation set).

Subgraph Retrieval. For each statement $[c, q, a]$, a subgraph $\mathcal{G}_{q_a} \subseteq \mathcal{G}$ is retrieved based on the input's context, which may include nodes that correspond to question entities or answer entities. For example, [34] performs entity linking between the question's and the KG's entities, and extracts the two-hop nodes

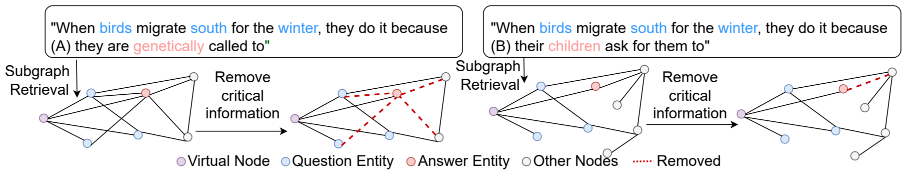

"When birds migrate south for the winter, they do it because (A) they are genetically called to"
"When birds migrate south for the winter, they do it because (B) their children ask for them to"
Subgraph Retrieval
Remove critical information
Subgraph Retrieval
Remove critical information
Virtual Node
Question Entity
Answer Entity
Other Nodes
Removed

Fig. 2: Setting when critical answer information is removed from the KG. Originally, GNNs propagate information from the answer nodes (light-red color) to other nodes of the graph, and the candidate answer with more links is more likely to be the correct answer [29]. If we remove the answer node's edges, information propagation becomes challenging and GNNs struggle to discriminate between correct and incorrect answers (Section 4).

between the linked entities, filtering out question-irrelevant edges. The context-specific subgraph $\mathcal{G}_{q_a} = (\mathcal{V}_{q_a}, \mathcal{E}_{q_a})$ contains a set of nodes $\mathcal{V}_{q_a}$, the set of relations $\mathcal{R}$ and a set of facts $\mathcal{E}_{q_a} \subseteq \mathcal{V}_{q_a} \times \mathcal{R} \times \mathcal{V}_{q_a}$. Note that different candidate answers $a \in \mathcal{A}$ for question q lead to different subgraphs $\mathcal{G}_{q_a}$ as the context changes. Similar to previous works [35], a virtual question node is added to each subgraph and is linked to question and answer entities, by adding *(question, entity, birds)* and *(question, a_entity, children)* to the edge set $\mathcal{E}_{q_a}$, for example.

Graph Neural Networks (GNNs). GNNs learn to update the representation of a node v by aggregating representations of its neighbors, set $\mathcal{N}(v)$, in a recursive manner. Following the message passing strategy [5], GNNs update the representation $\boldsymbol{h}_v^{(l)}$ of node v at layer l as

$$\boldsymbol{h}_v^{(l)} = \psi\Big(\boldsymbol{h}_v^{(l-1)}, \phi\big(\{\boldsymbol{m}_{(v,v')}^{(l)} : v' \in \mathcal{N}(v)\}\big)\Big), \qquad (1)$$

where $\boldsymbol{m}_{(v,v')}$ is the message between two entities v and v', linked with a relation r, and depends on their corresponding representations. Function $\phi(\cdot)$ is an aggregation, e.g., sum, of all neighboring messages and function $\psi(\cdot)$ is a neural network. In order to enable language to graph information fusion, many QA GNN-based approaches [34,35] set the question node's embedding to the question representation obtained by the LM.

4 Empirical Findings on Robustness

As discussed in Section 3, GNNs leverage the graph information of the retrieved KG to update the node embeddings. However, current QA GNN-based approaches use external node embeddings to represent the nodes' information. The representation space of these embeddings is not necessarily aligned with the representation space of the LM, which limits the effectiveness of fusing semantic information between natural language and graph. For example, [29] and [21] show that replacing the node embeddings with simple node features, such as node types (node coloring in Figure 2), leads to better QA performance. These findings indicate that GNNs rely on the underlying graph statistics, e.g., the number of connections between answer nodes and other graph nodes [29], to discriminate between correct and incorrect answers.

Table 1: QA performance comparison when complete information about candidate answers are in the graph (w/ ans) and when their edges are removed (w/o ans). Δ_{acc} denotes the relative performance degradation.

	OBQA			RiddleQA		
	w/ ans	w/o ans	Δ_{acc}	w/ ans	w/o ans	Δ_{acc}
LM (w/o KG)	64.8	64.8	0.0	60.7	60.7	0.0
LM + GNN*	68.3	65.0	-4.8%	66.7	64.8	-2.8%

*Results are averaged over three representative QA-GNNs (Section 7.1). The seed LM is `RoBERTa-Large`.

To test our hypothesis, we experiment with a setting for multiple-choice QA, where critical information is missing from the KG. For each retrieved KG subgraph, we remove the facts (edges) that include the candidate answer a, similar to the case where answer entities are not linked in the graph. The setting becomes challenging as GNNs cannot easily propagate answer-specific information and need to leverage information about the remaining entities to improve the LM's reasoning. The studied setting is illustrated in Figure 2.

We present the results for OBQA and RiddleQA datasets in Table 1. When removing answer information from the KG, GNNs show significant performance degradation and cannot effectively discriminate between correct and incorrect answers. The relative performance degradation is up to 4.8% for OBQA and using the external KG improves over the LM (w/o KG) by only 0.2% accuracy points. The experiment suggests that the message passing of GNNs learns to propagate answer-specific information, depends on the connectivity of the answer nodes, and is limited when this information is not present or is removed from the graph.

5 SemPool: Semantic Graph Pooling

We present a graph-based pooling method, termed SemPool, that aims at robustness during QA with KGs. Unlike message passing methods that depend on *local graph* information around the answer nodes, SemPool leverages *global textual* information from the KG to represent its semantics. Global information is more robust under local graph perturbations, e.g., incomplete edges around nodes, while the textual representation of the KG aids its integration to the LM. SemPool's overall framework is depicted in Figure 1. Next, we provide SemPool's components in detail.

5.1 KG Initialization

SemPool retrieves a subgraph $\mathcal{G}_{q_a}$ for context $[c, q, a]$ following existing works [34] (Section 3). From now on, we denote the retrieved subgraph as $\mathcal{G}_q$ (instead of $\mathcal{G}_{q_a}$) for better readability. SemPool uses the LM to encode the textual information of each fact $(h, r, t) \in \mathcal{E}_q$ in the retrieved subgraph, aiming at semantic alignment between language and graph, as follows. We transform each edge in

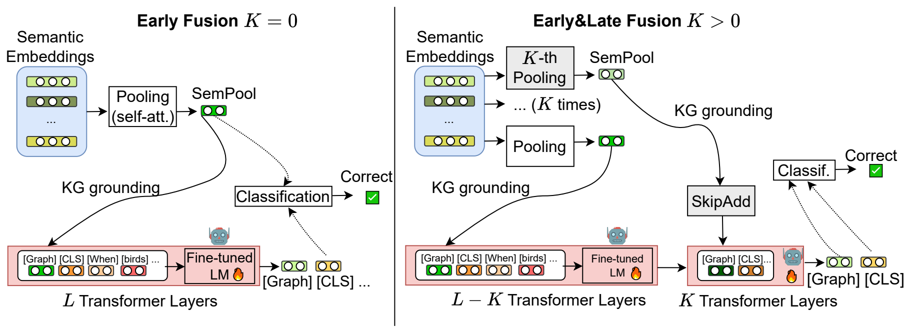


Fig. 3: SemPool architecture with early (left) and late (right) fusion. Number K represents the number of late fusion layers.

the subgraph, $(h, r, t) \in \mathcal{E}_q$, into natural language based on predefined templates for each relation $r \in \mathcal{R}$. For instance, (*winter, causes, bird migration*) is transformed to "winter causes bird migration". Then the verbalized fact, e, for edge (h, r, t) is encoded by the *pre-trained* LM. In order to compute a single edge embedding $\boldsymbol{h}_e$ for each edge e, we use mean-pooling or cls-pooling of the computed token embeddings.

5.2 Pooling

SemPool performs global pooling over the edge embeddings $\{\boldsymbol{h}_e : e \in \mathcal{E}_q\}$ of the retrieved subgraph $\mathcal{G}_q$. However, $\mathcal{G}_q$ is determined based on the linked nodes and their neighbors (Section 3) and as a result, it may contain some noisy facts in the set $\mathcal{E}_q$. Thus, we propose a self-attention pooling layer that weights the importance of each fact with respect to the semantics of the subgraph. Specifically, we compute a global graph representation $\boldsymbol{g}_q$ with a weighted aggregation by

$$\boldsymbol{g}_q = \sum_{e \in \mathcal{E}_q} a_e f_v(\boldsymbol{h}_e), \tag{2}$$

where $f_v : \mathbb{R}^d \to \mathbb{R}^d$ is a linear projection and $a_e \in [0, 1]$ measures the importance of each fact $e \in \mathcal{E}_q$. Weight a_e is computed by a softmax$(\cdot)$ operation as

$$a_e = \text{softmax}_{e \in \mathcal{E}_q}\big(f_k(\boldsymbol{h}_e)\big), \tag{3}$$

where $f_k : \mathbb{R}^d \to \mathbb{R}^d$ is a neural network. Learning the importance of each fact facilitates the identification of facts that provide new information to the LM for the QA task.

5.3 KG Grounding

We ground the LM to the subgraph's semantic information $\boldsymbol{g}_q$ by inserting a special `[Graph]` token in the beginning of the question q. The embedding of the `[Graph]` token is set to $\boldsymbol{g}_q$. During training, the LM's transformer layers [27] learn to fuse information between the question's tokens and the graph token.

Early Fusion. In the early fusion approach, the graph representation $\boldsymbol{g}_q$ is prepended to the query. After L transformer layers, the LM outputs the final hidden states $[\boldsymbol{h}_{\texttt{Graph}}^{(L)}, \boldsymbol{h}_{\texttt{CLS}}^{(L)}, \dots, \boldsymbol{h}_T^{(L)}]$. We use the `[CLS]` token as the final question representation $\boldsymbol{q}^{(L)} := \boldsymbol{h}_{\texttt{CLS}}^{(L)}$ for answer classification. Moreover, we use $\boldsymbol{g}_q$ for the answer classification loss so that the pooling module gives more attention to facts useful for the QA task. Given an answer candidate $a \in \mathcal{A}$, its probability $p(a|q)$ of being the correct answer for question q is computed by

$$p(a|q) = \exp\big(f_q(\boldsymbol{q}^{(L)}) + f_g(\boldsymbol{g}_q)\big), \tag{4}$$

where $f_q, f_g : \mathbb{R}^d \to \mathbb{R}^1$ are MLP networks. During training, we optimize the parameters via the cross entropy loss.

Early&Late Fusion. The late fusion approach has skip connections [6] that fuse graph information into deeper layers of the LM. This encourages the LM to mix useful graph semantics with language before predictions. The hyperparameter K denotes the K last transformer layers where graph information is fused. For each transformer layer, we have a dedicated pooling module that computes $\boldsymbol{g}_q^{(k)}$, where $k \in \{0, \dots, K\}$. Similar to Equation 2 and Equation 3, each $\boldsymbol{g}_q^{(k)}$ is obtained via

$$\boldsymbol{g}_q^{(k)} = \sum_{e \in \mathcal{E}_q} a_e^{(k)} f_v^{(k)}(\boldsymbol{h}_e), \quad a_e^{(k)} = \mathrm{softmax}_{e \in \mathcal{E}_q}(f_k^{(k)}(\boldsymbol{h}_e)). \tag{5}$$

In the beginning, we set the embedding of the graph token $\boldsymbol{h}_g^{(0)}$ to $\boldsymbol{g}_q^{(0)}$. At the $(L-k)$-th layer of the LM, $\boldsymbol{h}_{\texttt{GRAPH}}^{(L-k)}$ is updated via a skip connection as

$$\boldsymbol{h}_{\texttt{GRAPH}}^{(L-k)} = \boldsymbol{h}_{\texttt{GRAPH}}^{(L-k)} + \boldsymbol{h}_g^{(k)}. \tag{6}$$

We compute the final answer probabilities as

$$p(a|q) = \exp\big(f_q(\boldsymbol{q}^{(L)}) + f_g(\boldsymbol{h}_{\texttt{GRAPH}}^{(L)})\big), \tag{7}$$

where $f_q, f_g : \mathbb{R}^d \to \mathbb{R}^1$ are MLP networks. Note that the final representation $\boldsymbol{g}_q^{(L)}$ used for answer classification depends on the previous states of $\{\boldsymbol{g}_q^{(k)}\}_{k=1}^K$, which are optimized altogether during training.

6 Experimental Setting

QA Datasets. We evaluate SemPool on three multiple-choice question-answering datasets across two domains. OpenBookQA (**OBQA**; [18]) dataset is a 4-way multiple-choice QA dataset that requires reasoning with elementary science knowledge. It contains 5,957 questions along with an optional open book of scientific facts. We use the official data split. RiddleSense (**RiddleQA**; [11]) dataset is a 5-way multiple-choice task testing complex riddle-style commonsense reasoning. It has 5,715 questions and we split the dev set in half to make in-house dev/test sets. MedQA-USMLE (**MedQA**; [8]) dataset is 4-way multiple-choice

Table 2: Test performance comparison on QA datasets. Purple color denotes performance degradation at the adversarial setting, while teal color denotes improvement.

	(w/ ans.) OBQA	RiddleQA	MedQA	Avg.	(w/o ans.) OBQA	RiddleQA	MedQA	Avg.	Avg.
message passing									
LM + QAGNN	67.8 (±2.8)*	67.0*	38.0*	57.60	67.0 (±0.7)	65.2 (±0.4)	36.8	56.33	56.97
LM + GreaseLM	66.9*	67.2*	38.5*	57.53	64.4 (±3.2)	63.7 (±1.7)	39.0	55.70	56.61
LM + GSC	70.3 (±0.8)*	66.0 (±1.5)	38.0	**58.10**	63.6 (±2.6)	65.6 (±0.7)	37.8	55.66	56.88
no message passing									
LM (w/o KG)	64.8 (±2.4)*	60.7*	37.2*	54.23	64.8 (±2.4)*	60.7*	37.2*	54.23	54.23
LM + SemPool	67.7 (±1.2)	67.3 (±0.4)	38.9	57.96	69.5 (±0.5)	67.2 (±1.2)	39.4	**58.70**	**58.33**

*Published results. We use the `RoBERTa-Large` LM for OBQA and RiddleQA (commonsense). We use the `SapBERT-Base` LM for MedQA (biomedical).

task that originates from the USMLE practice sets, requiring biomedical and clinical knowledge. The dataset has 12,723 questions and we use the original data splits.

Knowledge Graphs. Following prior works, we use ConceptNet [24], a general-domain knowledge graph, as our external knowledge source $\mathcal{G}$ for OBQA and RiddleQA. ConceptNet has 799,273 nodes and 2,487,810 edges in total. For MedQA-USMLE, we use the KG provided by [8]. This KG contains 9,958 nodes and 44,561 edges. For each question, we retrieve subgraphs following the algorithm of [34]. We set the default subgraph size to 32 nodes, which empirically performs well in all datasets. In addition, we study the setting in Figure 2.

Language & Graph Encoding. We use (i) RoBERTa-Large [14] and AristoRoBERTa [2] for the experiments on OBQA and RiddleQA, and (ii) SapBERT [12] and BioLinkBERT-Base [33] for MedQA, demonstrating SemPool's effectiveness with respect to different LM initializations. We encode KG facts via the respective pretrained LMs for each case: (i) RoBERTa-Large for OBQA and RiddleQA, (ii) SapBERT and BioLinkBERT-Base for MedQA.

SemPool Implementation. We follow prior work's implementation for the QA task [35]. We use RAdam optimizer with learning rates selected from $\{1, 2, 5\} \times \text{e}{-5}$ for the LM and set to 1e−3 for our pooling encoder with a batch size of 32, training epochs are selected from $\{20, 30, 70\}$. For SemPool, we tune K from $\{0, 2, 3, 5\}$ and select cls or mean pooling for the KG edge representation based on the dev set. Experiments are conducted on a Nvidia GeForce RTX 3090 24GB machine.

Compared Methods. We compare SemPool with representative LM+KG GNN methods: QAGNN [34], GreaseLM [35], and GSC [29]. QAGNN uses the question's representation to guide the GNN updates. GreaseLM fuses information from both the language and the graph into the last interaction layers of the LM. QAGNN and GreaseLM use external node embeddings for the KG. GSC treats language and graph separately, and relies on the node/edge types for discriminating between correct and incorrect answers. For a fair comparison, we

Table 3: Performance comparison on QA datasets with different LMs at the adversarial setting (w/o ans).

	OBQA		RiddleQA		MedQA		Avg.
	RoBERTa	AristoRoBERTa	RoBERTa	AristoRoBERTa	SapBERT	BioLinkBERT	
	(dev / test)	(dev / test)	(dev / test)	(dev / test)	(dev / test)	(dev / test)	(dev / test)
LM + GNN$^{\heartsuit}$	71.4 / 67.0	71.4 / 74.0	65.7 / 66.5	66.1 / 69.0	38.9 / 39.0	40.7 / 40.9	59.03 / 59.40
LM + SemPool	70.4 / 69.6	73.0 / 75.2	66.2 / 67.7	68.2 / 69.2	37.9 / 39.4	42.3 / 41.6	**59.66** / **60.45**

$^{\heartsuit}$ We use the best performing GNN from Table 2: QAGNN for OBQA, GSC for RiddleQA, GreaseLM for MedQA.

use the same LM for all compared methods. In addition, we report performance of fine-tuning the LM without using any KG information, 'LM (w/o KG)'.

7 Results

7.1 Main Results

We present the results when comparing SemPool with existing GNN-based (message passing) approaches. Table 2 shows that SemPool is the most robust method under different configurations and datasets, although it does not involve any complex message passing. SemPool improves over GNNs by 1.45-1.72% accuracy points on average, while GNNs struggle on the setting when answer node facts are removed from the KG (w/o ans.). Moreover, the benefit of grounding the LM's reasoning to the KG becomes clear when comparing SemPool with LM (w/o KG), where SemPool significantly improves performance by 4.1% points.

In Table 3, we present results when using different LMs for the QA task. SemPool outperforms the *best* performing GNN by 0.63% and 1.05% accuracy points for the dev and test set, respectively, at the critical setting, when answer information is missing. We observe that SemPool's improvements increase with more powerful LMs: AristoRoBERTa for OBQA and RiddleQA, and BioLinkBERT for MedQA. In these cases, SemPool outperforms the GNNs by 1.6% (OBQA), 2.1% (RiddleQA), and 1.7% (MedQA) accuracy points at the dev set.

7.2 Ablation Studies & Analysis

Table 4: Dev set performance of different embedding models, averaged over OBQA and RiddleQA datasets.

	Language Encoder	
	RoBERTa	AristoRoBERTa
Graph Encoder		
RoBERTa	68.3	70.6
SBERT	52.3	70.8

Semantic Alignment. Table 4 shows the importance of semantic alignment between language and graph representations. When using RoBERTa for QA and SBERT for computing graph embeddings, language and graphs semantics are not aligned, which leads to poor performance. On the other hand, using RoBERTa as the graph encoder improves performance by up to 17% accuracy points. AristoRoBERTa is pre-tuned for QA tasks and thus, it benefits from both RoBERTa and SBERT graph embeddings.

Graph Fusion. Figure 4 shows the importance of graph to language fusion. In most cases, late fusion ($K > 0$) outperforms early fusion ($K = 0$) as it injects

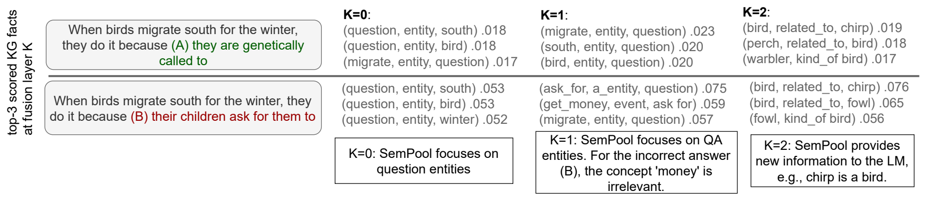

top-3 scored KG facts at fusion layer K
When birds migrate south for the winter, they do it because (A) they are genetically called to
When birds migrate south for the winter, they do it because (B) their children ask for them to
K=0:
(question, entity, south) .018
(question, entity, bird) .018
(migrate, entity, question) .017
K=1:
(migrate, entity, question) .023
(south, entity, question) .020
(bird, entity, question) .020
K=2:
(bird, related_to, chirp) .019
(perch, related_to, bird) .018
(warbler, kind_of bird) .017
(question, entity, south) .053
(question, entity, bird) .053
(question, entity, winter) .052
(ask_for, a_entity, question) .075
(get_money, event, ask for) .059
(migrate, entity, question) .057
(bird, related_to, chirp) .076
(bird, related_to, fowl) .065
(fowl, kind_of bird) .056
K=0: SemPool focuses on question entities
K=1: SemPool focuses on QA entities. For the incorrect answer (B), the concept 'money' is irrelevant.
K=2: SemPool provides new information to the LM, e.g., chirp is a bird.

Fig. 5: Working mechanism of SemPool: Top-3 scored facts at each $K \in \{0, 1, 2\}$, along with their attention weights.

graph information in multiple LM's layers. The optimal number of fusion layers K is model and task specific, but can be tuned based on the dev set. For example, RiddleQA has more complex questions and requires $K = 5$ fusion layers for the RoBERTa LM.

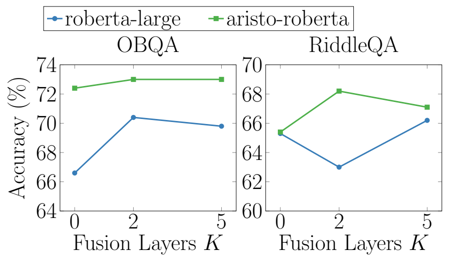


Fig. 4: Dev set performance with respect to the number K of fusion layers, using two different LMs.

Interpretability. Figure 5 illustrates the working mechanism of SemPool in one examples case from the OBQA dataset. We observe that different layers of SemPool extract different semantics. At $K = 0$, SemPool focuses on question entities (relation: 'entity'), which helps the LM give additional importance to the linked entities during its first layers of reasoning. At $K = 1$, SemPool focuses on both question and answer entities (relation: 'a_entity'). Thus, the LM uses additional semantics for the candidate answers. At the last fusion layer, SemPool learns to aggreagate new information for the LM, e.g., *(bird, related_to, chirp)*. This helps the LM to ground its predictions based on the global KG semantics. For the incorrect answer (B), SemPool identifies the irrelevant concepts 'ask_for' and 'get_money' that do not provide any useful information to the LM.

8 Conclusions

We study a critical setting for KG-based QA, where information about the candidate answer entities is missing from the KG. Our empirical results showed that graph-based (message passing) approaches struggle on the QA task under answer-based graph perturbations (Section 4). We propose SemPool, a graph pooling approach, that is more robust on KG-based QA task as it treats the graph as a set of facts. Experimental results show that SemPool outperforms competing methods by 2.27% accuracy points, while offering interpretability during inference (Section 7.2).

Acknowledgements. This work was supported in part by NSF (1447788, 1704074, 1757916, 1834251, 1834332), Army Research Office (W911NF1810344), Intel

Corp, and Amazon Web Services. Access to research and computing facilities was provided by the Minnesota Supercomputing Institute.

References

1. Agarwal, O., Ge, H., Shakeri, S., Al-Rfou, R.: Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In: NAACL (2021)
2. Clark, P., Etzioni, O., Khot, T., Khashabi, D., Mishra, B., Richardson, K., Sabharwal, A., Schoenick, C., Tafjord, O., Tandon, N., et al.: From 'f'to 'a'on the ny regents science exams: An overview of the aristo project. AI Magazine (2020)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Feng, Y., Chen, X., Lin, B.Y., Wang, P., Yan, J., Ren, X.: Scalable multi-hop relational reasoning for knowledge-aware question answering. In: EMNLP (2020)
5. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML (2017)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE CVPR (2016)
7. Ioannidis, V.N., Song, X., Zheng, D., Zhang, H., Ma, J., Xu, Y., Zeng, B., Chilimbi, T., Karypis, G.: Efficient and effective training of language and graph neural network models. arXiv preprint arXiv:2206.10781 (2022)
8. Jin, D., Pan, E., Oufattole, N., Weng, W.H., Fang, H., Szolovits, P.: What disease does this patient have? a large-scale open domain question answering dataset from medical exams. Applied Sciences (2021)
9. Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., Hajishirzi, H.: UNIFIEDQA: Crossing format boundaries with a single QA system. In: EMNLP Findings (2020)
10. Lin, B.Y., Chen, X., Chen, J., Ren, X.: Kagnet: Knowledge-aware graph networks for commonsense reasoning. In: EMNLP-IJCNLP (2019)
11. Lin, B.Y., Wu, Z., Yang, Y., Lee, D.H., Ren, X.: Riddlesense: Reasoning about riddle questions featuring linguistic creativity and commonsense knowledge. In: ACL Findings (2021)
12. Liu, F., Shareghi, E., Meng, Z., Basaldella, M., Collier, N.: Self-alignment pretraining for biomedical entity representations. In: NAACL-HLT (2021)
13. Liu, J., Yang, C., Lu, Z., Chen, J., Li, Y., Zhang, M., Bai, T., Fang, Y., Sun, L., Yu, P.S., et al.: Towards graph foundation models: A survey and beyond. arXiv preprint arXiv:2310.11829 (2023)
14. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
15. Mallen, A., Asai, A., Zhong, V., Das, R., Hajishirzi, H., Khashabi, D.: When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. In: ACL (2023)
16. Mavromatis, C., Ioannidis, V.N., Wang, S., Zheng, D., Adeshina, S., Ma, J., Zhao, H., Faloutsos, C., Karypis, G.: Train your own gnn teacher: Graph-aware distillation on textual graphs. In: ECML-PKDD (2023)

17. Mavromatis, C., Karypis, G.: Rearev: Adaptive reasoning for question answering over knowledge graphs. In: EMNLP Findings (2022)
18. Mihaylov, T., Clark, P., Khot, T., Sabharwal, A.: Can a suit of armor conduct electricity? a new dataset for open book question answering. In: EMNLP (2018)
19. Mihaylov, T., Frank, A.: Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In: ACL (2018)
20. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying large language models and knowledge graphs: A roadmap. arXiv preprint arXiv:2306.08302 (2023)
21. Park, J., Choi, H.K., Ko, J., Park, H., Kim, J.H., Jeong, J., Kim, K., Kim, H.: Relation-aware language-graph transformer for question answering. In: AAAI (2023)
22. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR (2020)
23. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC (2018)
24. Speer, R., Chin, J., Havasi, C.: Conceptnet 5.5: An open multilingual graph of general knowledge. In: AAAI (2017)
25. Sun, Y., Shi, Q., Qi, L., Zhang, Y.: Jointlk: Joint reasoning with language models and knowledge graphs for commonsense question answering. In: NAACL-HLT (2022)
26. Tian, Y., Song, H., Wang, Z., Wang, H., Hu, Z., Wang, F., Chawla, N.V., Xu, P.: Graph neural prompting with large language models. arXiv preprint arXiv:2309.15427 (2023)
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
28. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
29. Wang, K., Zhang, Y., Yang, D., Song, L., Qin, T.: Gnn is a counter? revisiting gnn for question answering. In: ICLR (2022)
30. Xie, H., Zheng, D., Ma, J., Zhang, H., Ioannidis, V.N., Song, X., Ping, Q., Wang, S., Yang, C., Xu, Y., et al.: Graph-aware language model pre-training on a large graph corpus can help multiple graph applications (2023)
31. Xie, T., Wu, C.H., Shi, P., Zhong, R., Scholak, T., Yasunaga, M., Wu, C.S., Zhong, M., Yin, P., Wang, S.I., et al.: Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In: EMNLP (2022)
32. Yasunaga, M., Bosselut, A., Ren, H., Zhang, X., Manning, C.D., Liang, P., Leskovec, J.: Deep bidirectional language-knowledge graph pretraining. In: NeurIPS (2022)
33. Yasunaga, M., Leskovec, J., Liang, P.: Linkbert: Pretraining language models with document links. In: ACL (2022)
34. Yasunaga, M., Ren, H., Bosselut, A., Liang, P., Leskovec, J.: Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In: NAACL (2021)
35. Zhang, X., Bosselut, A., Yasunaga, M., Ren, H., Liang, P., Manning, C.D., Leskovec, J.: Greaselm: Graph reasoning enhanced language models for question answering. In: ICLR (2022)
36. Zhao, J., Qu, M., Li, C., Yan, H., Liu, Q., Li, R., Xie, X., Tang, J.: Learning on large-scale text-attributed graphs via variational inference. In: ICLR (2023)
37. Zhu, Z., Galkin, M., Zhang, Z., Tang, J.: Neural-symbolic models for logical queries on knowledge graphs. In: ICML (2022)

Appendix

A SemPool

Table 5: QA examples from OBQA, RiddleQA, and MedQA datasets.

OBQA	When birds migrate south for the winter, they do it because (A) they are genetically called to (B) their children ask for them to (C) it is important to their happiness (D) they decide to each year
OBQA+fact	Migration is an instinctive behavior. When birds migrate south for the winter, they do it because (A) they are genetically called to (B) their children ask for them to (C) it is important to their happiness (D) they decide to each year
RiddleQA	What turns everything around, but does not move? (A) side (B) mirror (C) street corner (D) drive (E) corner
MedQA	A 51-year-old female presents with intermittent right upper quadrant discomfort. The physician suspects she is suffering from biliary colic and recommends surgery. Following surgery, brown stones are removed from the gallbladder specimen. What is the most likely cause of the gallstone coloring? (A) E. coli infection; beta-glucoronidase release (B) Shigella infection; HMG-CoA reductase release (C) Shigella infection; beta-glucoronidase release (D) Bile supersaturated with cholesterol; beta-glucoronidase release

A.1 Comparison with Existing Approaches

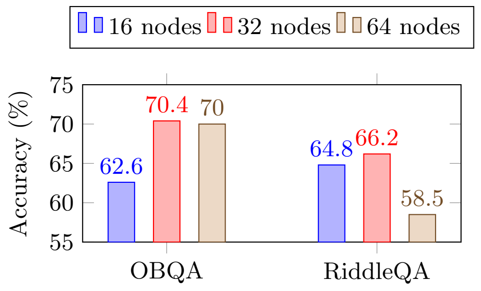
KG Grounding. One benefit of SemPool is that KG information is inserted into the LM, grounding its reasoning at different layers. Most existing approaches do not fuse information into the LM's layers [34,29] or fuse information at last layers only [35,25,21]. The goal of these methods is to use the question representation to guide the GNN updates. On the other hand, SemPool aims at improving the LM's reasoning, by conditioning its transformer layers to KG information.
Semantic Alignment. SemPool uses the same seed LM to encode textual information from the KG facts as well as for initializing the QA-finetuned LM. As a result, SemPool represents both the graph and the language in the same semantic space, which can facilitate knowledge exchange between the two modalities. In contrast, most existing approaches [4] use external embeddings to represent the graph, which might require additional learning for aligning the graph and the language representations.

A.2 Complexity

GNNs are *node-centric* and they recursively update the node embeddings at different layers (Section 3). For each node in $\mathcal{V}_q$, GNNs perform $\mathcal{O}(K)$ aggregations, where K is the number of GNN layers, requiring $\mathcal{O}(|\mathcal{V}_q|K)$ total aggregations. At each aggregation, GNNs aggregate $\mathcal{O}(\Delta)$ messages, where Δ is the maximum node degree (usually, $\Delta \ll |\mathcal{V}_q|$ in sparse graphs). SemPool is *graph-centric* as it pre-computes *edge* embeddings and aggregates them once into a single representation during inference. SemPool only requires $\mathcal{O}(K+1)$ *total* aggregations, where K is the number of fusion layers. Each aggregation involves $|\mathcal{E}_q|$ precomputed messages.

Table 6: Hyperparameter settings for experiments.

Category	Hyperparameter	OBQA / RiddleQA	MedQA
Model	Number of fusion layers	$\{0, 2, 5\}$	$\{0, 3\}$
	Token pooling	{cls, mean}	{cls, mean}
Optimization	LM learning rate	$1e^{-5}$	$\{2e^{-5}, 5e^{-5}\}$
	Graph encoder learning rate	$1e^{-3}$	$1e^{-3}$
	Optimizer	RAdam	RAdam
	Epochs	70 / 30	20
	Batch size	32	32
Data	Max number of nodes	32	32
	Max number of tokens	100	512



(a) Dev set performance with respect to the subgraph size, setting the maximum node number to $\{16, 32, 64\}$.

Fig. 6: Ablation studies of different SemPool's components.

B Datasets & Experimental Setting

We provide example cases of the QA datasets used in Table 5. Hyperparameter settings for the experiments are given in Table 6.

C Results & Ablation Studies

C.1 Subgraph Size

In Figure 6a, we study the effect of the subgraph size retrieved, setting the number of maximum nodes to $\{16, 32, 64\}$. For OBQA, the retrieved subgraphs are sparser with 110-118 nodes on average, while for RiddleQA, the retrieved subgraphs are denser with 167-200 nodes on average. As Figure 6a shows, OBQA benefits from subgraphs with more edges that can provide additional factual information ($|\mathcal{V}_q| \in \{32, 64\}$), while RiddleQA benefits from smaller graphs that include fewer noisy edges ($|\mathcal{V}_q| \in \{16, 32\}$).

C.2 Integrating SemPool to QA systems

We further experiment using AristoRoBERTa with additional text data for OBQA, similar to SOTA systems. Table shows the potential of SemPool: Grounding the LM's reasoning to the KG (SemPool) and combining it with message passing over the KG (GSC), we can achieve higher accuracy than billion-scale LMs or SOTA systems for KG powered QA, such as DRAGON and QAT.

Table 7: SOTA performance on OBQA using additional scientific text.

System (#Params)	Acc.
T5 (3B) [22]	83.2
T5+KB ($\geq$11B)	85.4
UnifiedQA (11B) [9]	87.2
GreaseLM (359M) [35]	84.8
DRAGON (359M) [32]	87.8
AristoRoBERTa (355M) [2]	77.8
+ QAGNN [34]	82.8
+ JointLK [25]	85.6
+ GSC [29]	87.4
+ QAT [21]	87.6
+ SemPool + GSC (**ours**)	**88.2**