ReaRev: Adaptive Reasoning for Question Answering over Knowledge Graphs

Costas Mavromatis and **George Karypis**
University of Minnesota, USA
{mavro016, karypis}@umn.edu

Abstract

Knowledge Graph Question Answering (KGQA) involves retrieving entities as answers from a Knowledge Graph (KG) using natural language queries. The challenge is to learn to reason over question-relevant KG facts that traverse KG entities and lead to the question answers. To facilitate reasoning, the question is decoded into instructions, which are dense question representations used to guide the KG traversals. However, if the derived instructions do not exactly match the underlying KG information, they may lead to reasoning under irrelevant context. Our method, termed REAREV, introduces a new way to KGQA reasoning with respect to both instruction decoding and execution. To improve instruction decoding, we perform reasoning in an adaptive manner, where KG-aware information is used to iteratively update the initial instructions. To improve instruction execution, we emulate breadth-first search (BFS) with graph neural networks (GNNs). The BFS strategy treats the instructions as a set and allows our method to decide on their execution order on the fly. Experimental results on three KGQA benchmarks demonstrate the REAREV's effectiveness compared with previous state-of-the-art, especially when the KG is incomplete or when we tackle complex questions. Our code is publicly available at https://github.com/cmavro/ReaRev_KGQA.

1 Introduction

A knowledge graph (KG) is a relational graph that contains a set of known facts. These facts are represented as tuples *(subject, relation, object)*, where the subject and object are KG entities that link with the given relation. The knowledge graph question-answering (KGQA) task takes as input a natural language question and returns a set of KG entities as the answer. In the weakly-supervised KGQA setting (Berant et al., 2013), the only available supervisions during learning are question-answer pairs, e.g., "Q: *Who is the director of Pulp Fiction*? A: *Q. Tarantino*". Due to labelling costs, the ground-truth KG traversals, such as the path *Pulp Fiction* $\xrightarrow{\text{director}}$ *Q. Tarantino*, whose execution leads to the answers, are seldom available.

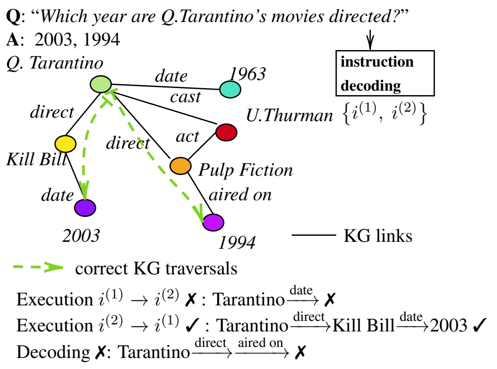
Q: "*Which year are Q.Tarantino's movies directed?*"
A: 2003, 1994
Q. Tarantino
date
1963
cast
direct
U.Thurman
instruction decoding
$\{i^{(1)},\ i^{(2)}\}$
direct
act
Kill Bill
Pulp Fiction
date
aired on
2003
1994
KG links
correct KG traversals
Execution $i^{(1)} \rightarrow i^{(2)}$ ✗ : Tarantino $\xrightarrow{\text{date}}$ ✗
Execution $i^{(2)} \rightarrow i^{(1)}$ ✓ : Tarantino $\xrightarrow{\text{direct}}$ Kill Bill $\xrightarrow{\text{date}}$ 2003 ✓
Decoding ✗: Tarantino $\xrightarrow{\text{direct}}\xrightarrow{\text{aired on}}$ ✗

Figure 1: The given question is decomposed into instructions $\{i^{(1)}, i^{(2)}\}$ that are matched with relations *date* and *direct*, respectively. If the instructions are executed in an incorrect order ($i^{(1)} \rightarrow i^{(2)}$), we cannot arrive at the answer. Moreover, if the instructions cannot be matched with the relation *aired on* (right KG traversal), we cannot arrive at the second answer.

The KGQA problem involves two modules: (i) retrieving question-relevant KG facts, and (ii) reasoning over them to arrive at the answers. For the reasoning module, a general approach (Lan et al., 2021) is to decode the question into dense representations (instructions) that guide the reasoning over the KG. The instructions are matched with one-hop KG facts (or relations) in an iterative manner, which induces KG traversals that lead to the answers. Instructions are usually generated by attending to different question's parts, e.g., "*...is the director...*" (Qiu et al., 2020a). KG traversals are typically performed by utilizing powerful graph reasoners, such as graph neural networks (GNNs) (Kipf and Welling, 2016; Sun et al., 2018).

The main challenge is that question answering is performed over rich graph structures with possibly complex semantics, and thus, instruction decoding and execution is a key factor for KGQA. Figure 1 shows an example where suboptimal initial instructions lead to incorrect KG traversals. While some methods (Miller et al., 2016; Zhou et al., 2018; Sun et al., 2018; Xu et al., 2019) attempt to improve the quality of the instructions, they are mainly designed to tackle specific question types, such as 2-hop or 3-hop questions, or show poor performance for complex questions (Sun et al., 2019).

Our method termed REAREV (Reason & Revise) introduces a new way to KGQA reasoning with respect to both instruction execution and decoding. To improve instruction execution, we do not use instructions in a pre-defined (possibly incorrect) order, but allow our method to decide on the execution order on the fly. We achieve this by emulating breadth-first search (BFS) with GNN reasoners. The BFS strategy treats the instructions as a set and the GNN decides which instructions to accept. To improve instruction decoding, we reason over the KG to obtain KG-aware information and use this information to adapt the initial instructions. Then, we restart the reasoning with the new instructions that are conditioned to the underlying KG semantics. To the best of our knowledge, adaptive reasoning with emulating graph search algorithms with GNNs has not been previously proposed for KGQA.

We empirically show that REAREV performs effective reasoning over KGs and outperforms other state-of-the-art. For KGQA with complex questions, REAREV achieves improvement for 4.1 percentage points at Hits@1 over the best competing approach.

Our contributions are summarized below:

- We improve instruction decoding via adaptive reasoning, which updates the instructions with KG-aware information.
- We improve instruction execution by emulating the breadth-first search algorithm with graph neural networks, which provides robustness to the instruction ordering.
- We achieve state-of-the-art (or nearly) performance on three widely used KGQA datasets: WebQuestions (Yih et al., 2015), Complex WebQuestions (Talmor and Berant, 2018), and MetaQA (Zhang et al., 2018).

2 Related Work

There are two mainstream approaches to solve KGQA: (i) parsing the question to executable KG queries like SPARQL, and (ii) grounding question and KG representations to a common space for reasoning.

Regarding the first case, early methods (Berant et al., 2013; Reddy et al., 2014; Bast and Haussmann, 2015) rely on pre-defined question templates to synthesize queries, which requires strong domain knowledge. Recent methods (Yih et al., 2015; Abujabal et al., 2017; Luo et al., 2018; Bhutani et al., 2019; Lan et al., 2019; Lan and Jiang, 2020; Qiu et al., 2020b; Sun et al., 2021; Das et al., 2021) use deep learning techniques to automatically generate such executable queries. However, they need ground-truth executable queries as supervisions (which are costly to obtain) and their performance is limited when the KG has missing links (non-executable queries).

Methods in the second category alleviate the need for ground-truth queries by learning natural language and KG representations to reason in a common space. These methods match question representations to KG facts (Miller et al., 2016; Xu et al., 2019; Atzeni et al., 2021) or to KG structure representations (Zhang et al., 2018; Han et al., 2021; Qiu et al., 2020a). More related to our work, GraftNet (Sun et al., 2018), PullNet (Sun et al., 2019), and NSM (He et al., 2021) enhance the reasoning with graph-based reasoners. Our approach aims at improving the graph-based reasoning via adaptive instruction decoding and execution.

Researchers have also considered the problem of performing KGQA over incomplete graphs (Min et al., 2013), where important information is missing. These methods either rely on KG embeddings (Saxena et al., 2020; Ren et al., 2021) or on side information, such as text corpus (Sun et al., 2018, 2019; Xiong et al., 2019; Han et al., 2020), to infer missing information. However, they offer marginal improvement over other methods when the KG is mostly complete.

KGQA has also been adapted to specific domains, such as QA over temporal (Mavromatis et al., 2021; Saxena et al., 2021) and commonsense (Speer et al., 2017; Talmor et al., 2019; Lin et al., 2019; Feng et al., 2020; Yasunaga et al., 2021) KGs.

3 Background

3.1 KG

A KG $\mathcal{G} := (\mathcal{V}, \mathcal{R}, \mathcal{F})$ contains a set of entities $\mathcal{V}$, a set of relations $\mathcal{R}$, and a set of facts $\mathcal{F}$. Each fact $(v, r, v') \in \mathcal{F}$ is a tuple where $v, v' \in \mathcal{V}$ denote the subject and object entities, respectively, and $r \in \mathcal{R}$ denotes the relation that holds between them. A KG is represented as a directed graph with $|\mathcal{R}|$ relation types, where nodes v and v' connect with a directed relation r if $(v, r, v') \in \mathcal{F}$. Nodes and relations are usually initialized with d-dimensional vectors (representations).

We denote $\boldsymbol{h}_v \in \mathbb{R}^d$ and $\boldsymbol{r} \in \mathbb{R}^d$ the representations for node v and relation r, respectively. We denote $\mathcal{N}_e(v)$ and $\mathcal{N}_r(v)$ the set of node v's neighboring entities and relations (including self-links), respectively. For example, $v' \in \mathcal{N}_e(v)$ and $r \in \mathcal{N}_r(v)$ if a fact (v', r, v) exists. We use the terms nodes and entities interchangeably.

3.2 KGQA

Given a KG $\mathcal{G} := (\mathcal{V}, \mathcal{R}, \mathcal{F})$ and a natural language question q, the task of KGQA is to extract a set of entities $\{a\} \in \mathcal{V}$ that correctly answer q. Following the standard setting in KGQA (Sun et al., 2018), we assume that the entities referred in the question are given and linked to nodes of $\mathcal{G}$ via entity linking algorithms (Yih et al., 2015). We denote these entities as $\{e\}_q \in \mathcal{V}$ (seed entities), e.g., *Q. Tarantino* in Figure 1.

The problem complexity is further reduced by extracting a question-specific subgraph $\mathcal{G}_q := (\mathcal{V}_q, \mathcal{R}_q, \mathcal{F}_q) \subset \mathcal{G}$ which is likely to contain the answers (more details in Section 5). Each question q and its answers $\{a\}_q \in \mathcal{V}_q$, referred to as question-answer pair, induces a question-specific labeling of the nodes. Node $v \in \mathcal{G}_q$ has label $y_v = 1$ if $v \in \{a\}_q$ and $y_v = 0$ otherwise. The task can be thus reduced to performing binary node classification over $\mathcal{G}_q$.

The KGQA problem involves two modules: (i) retrieving a question-specific $\mathcal{G}_q$ and (ii) reasoning over $\mathcal{G}_q$ to perform answer classification. In this work, we introduce a new way to advance KGQA reasoning capabilities.

3.3 GNNs

GNNs (Kipf and Welling, 2016; Schlichtkrull et al., 2018) are well-established graph representation learners suited for tasks such as node classification. Following the message passing strategy (Gilmer et al.), the core idea of GNNs is to update the representation of each node by aggregating itself and its neighbors' representations.

The GNN updates node representation $\boldsymbol{h}_v^{(l)}$ at layer l as

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

where $\boldsymbol{m}_{v'v}^{(l)}$ is the message between two neighbors v and v', and $\phi(\cdot)$ is an aggregation function of all neighboring messages. Function $\psi(\cdot)$ combines representations of consecutive layers. At each layer, GNNs capture 1-hop information (neighboring messages). An L-layer GNN model captures the neighborhood structure and semantics within L hops.

3.4 GNNs for KGQA

To better reason over multiple facts (graphs), successful KGQA methods utilize GNNs (Sun et al., 2018; He et al., 2021). The idea is to condition the message passing of Eq.(1) to the given question q. For example, if a question refers to movies, then 1-hop movie entities are more important. It is common practice (Qiu et al., 2020a; He et al., 2021; Shi et al., 2021; Lan et al., 2021) to decompose q into L representations $\{\boldsymbol{i}^{(l)}\}_{l=1}^{L}$ (instructions), where each one may refer to a specific question's context, e.g., movies or actors.

The instructions are used to guide different reasoning steps over $\mathcal{G}_q$ by writing the GNN updates as

$$\boldsymbol{h}_v^{(l)} = \psi\Big(\boldsymbol{h}_v^{(l-1)}, \phi\big(\{\boldsymbol{m}_{v'v}^{(l)} : v' \in \mathcal{N}_e(v) | \boldsymbol{i}^{(l)}\}\big)\Big), \quad (2)$$

where each GNN layer l is now conditioned to a different instruction $\boldsymbol{i}^{(l)}$. Message $\boldsymbol{m}_{v'v}^{(l)}$ usually depends on the representations of the corresponding fact (v', r, v).

The goal of GNNs is to selectively aggregate information from the question-relevant facts. Via Eq.(2), GNNs learn to match each $\boldsymbol{i}^{(l)}$ with 1-hop neighboring facts. Using the instructions $\{\boldsymbol{i}^{(l)}\}_{l=1}^{L}$ recursively, GNNs learn the sequence of facts (KG traversal) that leads to the final answers.

4 ReaRev Approach

ReaRev (Reason & Revise) enhances instruction decoding and execution for effective KGQA reasoning. Our contributions across these two dimensions are described below.

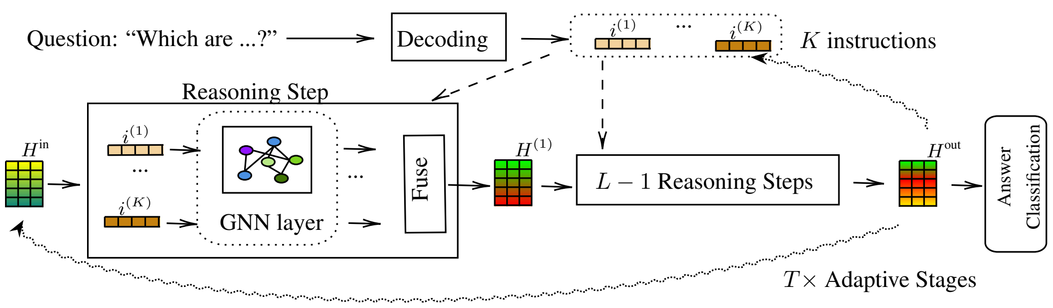

Figure 2: REAREV's adaptive reasoning. The question is decoded to K instructions. At L reasoning steps, we perform a BFS execution of the instructions. The procedure is repeated for T adaptive stages that enhance the initial instruction decoding.

4.1 REAREV's BFS Instruction Execution

GNN updates in Eq.(2) execute the instructions in a pre-defined order, which assumes that the generated instruction sequence matches exactly the information that is present in the KG. REAREV does not impose this assumption. Instead, it improves instruction execution by selecting the order by which to process the instructions on the fly, e.g., based on the KG semantics.

To achieve this, we emulate the breadth-first search strategy (BFS) by modifying the GNN updates in Eq.(2). The idea is to reason with all instructions at each step (breadh-first), before we decide which execution results to accept. We decompose the question into K instructions $\{\boldsymbol{i}^{(k)}\}_{k=1}^{K}$, but the number of instructions K is now decoupled form the number of GNN layers L. We derive instruction-specific representations $\tilde{\boldsymbol{h}}_v^{(k,l)}$ at each GNN layer as

$$\tilde{\boldsymbol{h}}_v^{(k,l)} = \phi\big(\{\boldsymbol{m}_{v'v}^{(l)} : v' \in \mathcal{N}_e(v) | \boldsymbol{i}^{(k)}\}\big). \tag{3}$$

To allow the model select which instruction-specific representations are useful, we fuse them with a learnable function $\psi(\cdot)$ as

$$\boldsymbol{h}_v^{(l)} = \psi\big(\boldsymbol{h}_v^{(l-1)}, \{\tilde{\boldsymbol{h}}_v^{(k,l)}\}_{k=1}^{K}\big). \tag{4}$$

As a result, the reasoning module has the capability to decide on the final execution plan, i.e., which instructions are accepted at which GNN layers.

Specifically, we compute message $\boldsymbol{m}_{v'v}^{(l)}$ between nodes v' and v as the representation of their corresponding relation $\boldsymbol{r}_{v'v} \in \mathbb{R}^d$ followed by a learnable projection matrix $\boldsymbol{W}_R^{(l)} \in \mathbb{R}^{d\times d}$. We condition $\boldsymbol{m}_{v'v}^{(l)}$ to the underlying instruction $\boldsymbol{i}^{(k)}$ by an element-wise multiplication followed by ReLU$(\cdot)$ nonlinerarity. In summary, we obtain

$$\boldsymbol{c}_{v'v}^{(k,l)} = \text{ReLU}(\boldsymbol{i}^{(k)} \odot \boldsymbol{W}_R^{(l)} \boldsymbol{r}_{v'v}), \tag{5}$$

where $\boldsymbol{c}_{v'v}^{(k,l)}$ denotes the question-relevant message from node v' at layer l and for instruction k.

To measure the importance of $\boldsymbol{c}_{v'v}^{(k,l)}$ from node v', we multiply it with $p_{v'}^{(l-1)}$ of the previous layer, where $\boldsymbol{p}^{(l)} \in [0,1]^{|\mathcal{V}_q|}$ is a probability vector (we shortly describe how it is computed). We aggregate neighboring facts for node v with a sum-operation and Eq.(3) becomes

$$\tilde{\boldsymbol{h}}_v^{(k,l)} = \sum_{v'\in\mathcal{N}(v)} p_{v'}^{(l-1)} \boldsymbol{c}_{v'v}^{(k,l)}. \tag{6}$$

To combine node representations from different instructions, we use the column-wise concatenation operation $||$ followed by a learnable projection matrix $\boldsymbol{W}_h^{(l)} \in \mathbb{R}^{d\times(K+1)d}$ (we observe similar performance with other fusion mechanisms, such as self-attention). Eq.(4) becomes

$$\tilde{\boldsymbol{h}}_v^{(l)} = ||_{k=1}^{K} \tilde{\boldsymbol{h}}_v^{(k,l)} \tag{7}$$

$$\boldsymbol{h}_v^{(l)} = \text{ReLU}\big(\boldsymbol{W}_h^{(l)}(\boldsymbol{h}_v^{(l-1)} || \tilde{\boldsymbol{h}}_v^{(l)})\big). \tag{8}$$

Finally, we collect $\boldsymbol{h}_v^{(l)}$ for all nodes to $\boldsymbol{H}^{(l)}$ and compute the probability vector $\boldsymbol{p}^{(l)}$ as

$$\boldsymbol{p}^{(l)} = \text{softmax}(\boldsymbol{H}^{(l)}\boldsymbol{w}). \tag{9}$$

Initially, we set $p_v^{(0)} = 0$ if $v \notin \{e\}_q$ and $p_v^{(0)} = 1$ otherwise, so that we start reasoning from the seed entities.

4.2 REAREV's Adaptive Instruction Decoding

If the instructions are computed based solely on the question's context, they are not conditioned with respect to the underlying KG. This is important when we need to reason over multiple facts (complex questions), over KGs with rich semantics (see

Figure 1) or over missing information (Sun et al., 2018).

Figure 2 shows how our adaptive reasoning works. After reasoning with L steps via Eq.(3) and Eq.(4), we keep the reasoning output (node representations $\boldsymbol{H}^{\text{out}}$) that contains KG-aware information. We use it to update the initial instruction decomposition $\{\boldsymbol{i}^{(k)}\}_{k=1}^{K}$ as well as the initial node representations $\boldsymbol{H}^{\text{in}}$. This procedure is repeated for T adaptation stages. Our goal is twofold; (i) to ground the instruction decomposition to underlying KG semantics, and (ii) to guide the reasoning process in an iterative manner. For example, some instructions may correspond to a part of the question that needs to be answered first, before reasoning with the rest instructions.

To update each $\boldsymbol{i}^{(k)}$ at every stage $t \in \{1, \dots, T\}$, we use the seed entities' final representations as the KG-aware information, which are computed by

$$\boldsymbol{h}_e = \sum_{v \in \{e\}_q} \boldsymbol{h}_v^{(L)}, \tag{10}$$

and compute the adapted instructions $\boldsymbol{i}^{(k)}$ as

$$\begin{aligned}\boldsymbol{i}^{(k)} =& (\boldsymbol{1} - \boldsymbol{g}^{(k)}) \odot \boldsymbol{i}^{(k)} + \\ & \boldsymbol{g}^{(k)} \odot \boldsymbol{W}_q(\boldsymbol{i}^{(k)} || \boldsymbol{h}_e || \boldsymbol{i}^{(k)} - \boldsymbol{h}_e || \boldsymbol{i}^{(k)} \odot \boldsymbol{h}_e).\end{aligned} \tag{11}$$

Here, $\boldsymbol{W}_q \in \mathbb{R}^{d \times 4d}$ are learnable parameters and $\boldsymbol{g}^{(k)} \in [0,1]^d$ is the output gate vector computed by a standard GRU (Cho et al., 2014). At every reasoning stage t, we set $\boldsymbol{H}^{\text{in}} = \boldsymbol{H}^{(L)}$ to encode information of the previous stage. However, we reset the probability vector $\boldsymbol{p}^{(0)}$ to the seed entities.

The algorithmic procedure of our ReaRev method is summarized in Algorithm 1. It takes as input a question q with seed entities $\{e\}_q$, and the corresponding question-specific KG subgraph $\mathcal{G}_q$ and classifies nodes as answers or non-answers. The number of adaptation stages T, instructions K, and reasoning steps L are hyper-parameters.

Algorithm 1 The high-level algorithmic procedure of ReaRev (inference).

```
 1: Input: Question $q$, KG subgraph $\mathcal{G}_q$, seed entities $\{e\}_q$, hyper-parameters: $T, K, L$.
 2: Initialize: $K$ instructions $\{\boldsymbol{i}^{(k)}\}_{k=1}^{K}$, node representations $\boldsymbol{H}^{\text{in}}$.
 3: for $t = 1$ to $T$ do
 4:     for $l = 1$ to $L$ do
 5:         for $k = 1$ to $K$ do
 6:             $\tilde{\boldsymbol{h}}_v^{(k,l)} = \phi\big(\{\boldsymbol{m}_{v'v}^{(l)} : v' \in \mathcal{N}(v) | \boldsymbol{i}^{(k)}\}\big)$.
 7:         end for
 8:         $\boldsymbol{h}_v^{(l)} = \psi\big(\boldsymbol{h}_v^{(l-1)}, \{\tilde{\boldsymbol{h}}_v^{(k,l)}\}_{k=1}^{K}\big)$.
 9:     end for
10:     Set $\boldsymbol{H}^{\text{out}}$ and $\boldsymbol{H}^{\text{in}}$ to $\boldsymbol{H}^{(L)}$.
11:     Update $\{\boldsymbol{i}^{(k)}\}_{k=1}^{K}$ using $\boldsymbol{H}^{\text{out}}$.
12: end for
13: Result: Classify node $v$ as answer or non-answer based on $\boldsymbol{h}_v^{\text{out}}$.
```

4.2.1 Optimization and Initialization

After reasoning for T stages, we obtain the final probability vector $\boldsymbol{p}^{\text{out}}$ by applying Eq.(9) to $\boldsymbol{H}^{\text{out}} = \boldsymbol{H}^{(L)}$. Here, we omit the superscript $t \in \{1, \dots, T\}$ for readability, e.g., $\boldsymbol{H}^{(T,L)}$. We optimize the model's parameters with a classification based loss function (we use the KL-divergence (Kullback and Leibler, 1951)), so that p_v^{out} is close to 1 if $v \in \{a\}_q$ and zero otherwise. During inference, since we do not have the answer nodes $\{a\}_q$, we rank the nodes as possible answers based on their final probabilities.

For better generalization to unobserved entities, we initialize node representations $\boldsymbol{h}_v^{\text{in}}$ as a function of their direct relations $r \in \mathcal{N}_r(v)$, as

$$\boldsymbol{h}_v^{\text{in}} = \text{ReLU}\Big(\sum_{r \in \mathcal{N}_r(v)} \boldsymbol{W}_0 \boldsymbol{r}\Big), \tag{12}$$

where $\boldsymbol{r} \in \mathbb{R}^d$ is the representation of relation r and $\boldsymbol{W}_0 \in \mathbb{R}^{d \times d}$ are learnable parameters. We derive the representation $\boldsymbol{r}$ with the same pre-trained language model used for the question based on the relation's surface form, if applicable. Otherwise, we randomly initialize and update $\boldsymbol{r}$ during training.

To capture multiple question's contexts, each instruction is initialized by dynamically attending to different question's tokens (Qiu et al., 2020a; He et al., 2021). First, we derive a representation $\boldsymbol{b}_j$ for token j and a question representation $\boldsymbol{q}$ with pre-trained language models, such as SentenceBERT (Reimers and Gurevych, 2019) (see also Appendix). Each instruction $\boldsymbol{i}^{(k)}$ is computed by

$$\boldsymbol{i}^{(k)} = \sum_j u_j^{(k)} \boldsymbol{b}_j, \tag{13}$$

where $u_j^{(k)} \in [0,1]$ is an attention weight for token j. To ensure different instructions can attend to different tokens, the dynamic attention is computed by

$$u_j^{(k)} = \text{softmax}_j(\boldsymbol{W}_u(\boldsymbol{q}^{(k)} \odot \boldsymbol{b}_j)),$$
$$\boldsymbol{q}^{(k)} = \boldsymbol{W}^{(k)}(\boldsymbol{i}^{(k-1)}||\boldsymbol{q}||\boldsymbol{q} \odot \boldsymbol{i}^{(k-1)}||\boldsymbol{q} - \boldsymbol{i}^{(k-1)}), \tag{14}$$

where $\boldsymbol{W}_u \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times 4d}$ are learnable parameters.

4.3 Complexity

ReaRev's computational complexity for each question is $\mathcal{O}(T|\mathcal{V}_q|L\Delta)$, assuming the KG is sparse with maximum node degree equal to Δ. We also assume that the inner loop in Algorithm 1 (line 6) is parallelized. On the other hand, the computations of traditional GNN-based KGQA methods (Section 3.4) can be achieved in $\mathcal{O}(|\mathcal{V}_q|L\Delta)$ time. However, we find that having $T = 2$ is sufficient for ReaRev in practice, so ReaRev's complexity does not necessarily increase linearly.

5 Experimental Setup

We experiment with three widely used KGQA benchmarks: WebQuestionsSP (Webqsp) (Yih et al., 2015), Complex WebQuestions 1.1 (CWQ) (Talmor and Berant, 2018), and MetaQA-3 (Zhang et al., 2018). We provide the final dataset statistics (see Section 5.2) in Appendix.

5.1 Dataset Details

Webqsp contains 4,737 natural language questions that are answerable using a subset Freebase KG. This KG contains 164.6 million facts and 24.9 million entities. The questions require up to 2-hop reasoning within this KG. Specifically, the model needs to aggregate over two KG facts for 30% of the questions, to reason over constraints for 7% of the questions, and to use a single KG fact for the rest of the questions.

CWQ is generated from Webqsp by extending the question entities or adding constraints to answers, in order to construct more complex multi-hop questions (34,689 in total). There are four types of question: composition (45%), conjunction (45%), comparative (5%), and superlative (5%). The questions require up to 4-hops of reasoning over the KG, which is the same KG with Webqsp.

MetaQA-3 consists of more than 100k 3-hop questions in the domain of movies. The questions were constructed using the KG provided by the WikiMovies (Miller et al., 2016) dataset, with about 43k entities and 135k triples.

5.2 Implementation and Evaluation Details

Recall that our ReaRev takes as input the question's seed entities $\{e\}_q$ and a question-specific subgraph $\mathcal{G}_q$. We use the seed entities provided by (Yih et al., 2015) for Webqsp, by (Talmor and Berant, 2018) for CWQ, and by (Miller et al., 2016) for MetaQA-3. We obtain subgraphs by (He et al., 2021). It runs the PageRank-Nibble (Andersen et al., 2006) (PRN) method from the seed entities to select the top-m entiites to be included in the subgraph, as in (Sun et al., 2018). We have $m = 2,000$ for Webqsp (full KG) and CWQ, and $m = 500$ for MetaQA-3 and Webqsp (incomplete KG).

We tune the hyper-parameters T (number of iterations), K (number of instructions), and L (number of GNN layers) amongst $T \in \{2, 3\}$, $K \in \{2, 3\}$, and $L \in \{2, 3, 4\}$. We perform model selection based on the best validation scores (more implementation details in the Appendix). For evaluation, we adopt two widely used metrics, Hits@1, which is the accuracy of the top-predicted answer, and the F1 score. To compute the F1 score we set a threshold equal to 0.95. For the competing approaches, we reuse the evaluation results reported in the corresponding papers, unless otherwise stated.

5.3 Competing Approaches

We compare with methods that focus on improving KGQA reasoning capabilities. KV-Mem (Miller et al., 2016) is a key-value memory network (Sukhbaatar et al., 2015) for KGQA. EmbedKGQA (Saxena et al., 2020) utilizes KG pre-trained embeddings (Trouillon et al., 2016) to improve multi-hop reasoning. GraftNet (Sun et al., 2018), HGCN (Han et al., 2020) and SGReader (Xiong et al., 2019) are GNN-based approaches, where GraftNet and HGCN use a convolution-based GNNs (Kipf and Welling, 2016), while SGReader uses attention-based GNNs (Veličković et al., 2017).

NSM (He et al., 2021) is the adaptation of Neural State Machines (Hudson and Manning, 2019) to KGQA and performs a GNN-based reasoning. NSM-distill (He et al., 2021) improves NSM for multi-hop reasoning by learning which intermediate nodes to visit via distillation (Hinton et al., 2015). TransferNet (Shi et al., 2021) improves multi-hop reasoning over the relation set. EmQL (Sun et al., 2020) and Rigel (Sen et al.,

Table 1: Performance comparison of different methods (Hits@1 or F1 scores in %). Bold fonts denote the best methods.

Method	**Webqsp** H@1 / F1	**CWQ** H@1
KV-Mem (Miller et al., 2016)	46.7 / 38.6	21.1
SGReader (Xiong et al., 2019)	67.2 / 57.3	–
EmbedKGQA (Saxena et al., 2020)	66.6 / –	*
GraftNet (Sun et al., 2018)	66.7 / 62.4	32.8
PullNet (Sun et al., 2019)	68.1 / –	45.9
TransferNet (Shi et al., 2021)	71.4 / –	48.6
Rigel (Sen et al., 2021)	73.3 / –	48.7
NSM (He et al., 2021)	68.7 / 62.8	47.6
NSM-distill (He et al., 2021)	74.3 / 67.4	48.8
EmQL (Sun et al., 2020)	75.5 / –	*
SQALER (Atzeni et al., 2021)	70.6 / –	*
SQALER+GNN (Atzeni et al., 2021)	76.1 / –	*
ReaRev	**76.4 / 70.9**	**52.9**

–: Result not reported.
*: Method cannot inherently tackle this setting.

2021) improve the ReifiedKB (Cohen et al., 2020) scalable baseline for deductive reasoning and reasoning with complex questions, respectively.

In addition, we compare with methods that focus on improving the question-specific input subgraph $\mathcal{G}_q$. PullNet (Sun et al., 2019) is built on top of GraftNet, but learns which nodes to retrieve via selecting shortest paths to the answers. SQALER (Atzeni et al., 2021) learns which relations (facts) to retrieve during KGQA by reconstructing KG traversals to answers.

6 Experimental Results

6.1 Main Results

We present the KGQA performance for the compared methods in Table 1. ReaRev outperforms the best performing method by 0.3% and 4.1% points at H@1 for Webqsp and CWQ, respectively.

For Webqsp, although most questions involve one-hop or two-hop reasoning, few training examples are given. Methods such as NSM-distill and SQALER+GNN tackle this challenge with additional supervision signals (compare NSM with NSM-distill and SQALER with SQALER-GNN), while EmQL uses pre-defined question templates to facilitate instruction execution. In contrast, ReaRev relies on its adaptive reasoning. If we compare ReaRev with other reasoning-based approaches (NSM, GraftNet, and SGReader), ReaRev performs better by 5.7-9.7% (H@1 points) and 8.1-13.6% (F1 points).

Table 2: H@1 / F1 results in % for Webqsp with incomplete KGs. "% KG completeness" denotes the percent of the remaining KG facts.

% of KG completeness	10%	30%	50%
GraftNet (Sun et al., 2018)	15.5 / 6.5	34.9 / 20.4	47.7 / 34.3
SGReader (Xiong et al., 2019)	17.1 / 7.0	35.9 / 20.2	49.2 / 33.5
HGCN (Han et al., 2020)	18.3 / 7.9	35.2 / 21.0	49.3 / 34.3
ReaRev	**19.4 / 8.6**	**37.9 / 23.6**	**53.4 / 39.9**

In CWQ, many questions include multiple seed entities and require both composition (sequential) and conjunction (parallel) reasoning, which makes the instruction execution challenging. ReaRev's breadth-first strategy and adaptive updates are designed to benefit such challenging cases. Note that some methods cannot inherently tackle this setting: EmbedKGQA requires single-entity questions, EmQL requires pre-defined question templates that are hard to derive for complex questions, and SQALER assumes that only composition question types are involved. ReaRev outperforms all other methods by more than 4% points at H@1.

6.2 Low-Data Regime Results

Webqsp contains mostly simple questions which are easily answerable over a full KG. Table 2 shows the performance for a more challenging task, when keeping 10%, 30%, and 50% of the KG's total facts. We compare against GraftNet, SGReader, and HGCN, which are GNN-based approaches especially designed for reasoning over incomplete KG subgraphs. ReaRev outperforms competing methods by 1.1-5.7% points at H@1 and by 0.7-5.6% points at F1. The best improvement is obtained for KG-50%, since there is more KG information that ReaRev can leverage during its adaptive reasoning.

MetaQA-3 has more than 100k train questions which involve only few KG relations. For a more challenging setting, we experiment with MetaQA-3 when we decrease the ratio of the KG completeness as well as the number of training question-answer pairs. We compare against NSM and NSM-distill that also rely on subgraph extraction with PRN (Section 5.2). We provide additional results in the Appendix. Table 3 shows that the more challenging the setting is, the better the improvement ReaRev achieves over NSM and NSM-distill. When the KG is 50% complete and we only use 1% of the training questions, ReaRev improves over NSM-distill by more than 10% points at H@1.

Table 3: H@1 results in % for MetaQA-3 under different settings. "% KG completeness" denotes the percent of the remaining KG facts and "% Train QAs" the percent of training question-answer pairs used.

% of KG completeness	100%	100%	50%	50%
% of Train QAs	10%	1%	10%	1%
NSM	98.8	89.6	71.8	49.7
NSM-distill	**98.9**	98.2	72.3	51.5
REAREV	**98.9**	**98.6**	**75.4**	**62.7**

6.3 Ablation Studies

Table 4 verifies that KGQA improvements stem from the algorithmic design of our method. For complex questions (CWQ), deriving the correct execution order of the instructions becomes challenging. By treating the instructions as a set, our BFS execution provides a performance gain of 4.7% points. When there are missing facts (Webqsp-50%), grounding the instruction decoding to the available information becomes crucial. Our adaptive decoding leverages this KG-aware information and provides a performance gain of 2.2% points.

Moreover, we experiment with hyper-parameter sensitivity. We intentionally set $L = 2$ for MetaQA-3, although it requires 3-hop reasoning, and gradually increase $T \in \{1, \dots, 5\}$ to evaluate whether REAREV can reach the answers. For $T = 4$ and $K \in \{2, 3\}$, REAREV achieves 84.5%-88.9% at H@1. When we set $T = 5$, REAREV further improves and achieves 98.7%-98.8% at H@1. In Table 5, we provide two case studies that show how the number T of adaptive stages impacts answer retrieval.

In addition, we have performed the following ablation study at MetaQA-3 as motivated by Figure 1. The idea is to switch some KG relations with semantically similar ones during inferece to evaluate REAREV's adaptiveness. We switch the KG relations {*directed by, written by, starred actors, release year*} (out of total 9 relations) to {*has executive, plot by, has cast, air on*} respectively. During training, we switch them with 5% probability, but during testing, we switch them with a 50% or 100% probability, which enforces a distribution shift over the underlying relations. REAREV (T=2) with adaptive stages outperforms REAREV (T=1) without adaptive stages in both cases. It performs 87.3% and 86.9% at F1, while REAREV (T=1) performs 84.5% and 81.1% at F1, respectively. This experiment also suggests that it is REAREV's algorithmic design that leads to its improvements.

Table 4: H@1 results in % (with performance drop) under different REAREV's modifications for Webqsp (50% complete) and CWQ. BFS Execution is described in Section 4.1 and Adaptive Decoding in Section 4.2.

Modification	**Webqsp-50%**	**CWQ**
REAREV	**53.4**	**52.9**
without BFS Execution	52.6 (-0.8)	48.2 (-4.7)
without Adaptive Decoding	51.2 (-2.2)	50.5 (-2.4)

Table 5: Question-answer pairs and predicted answers (with probabilities) with respect to the number T of adaptive stages.

Q: Who wrote movies that share directors with the movie The Comebacks? **A**: R. Schneider, T. Brady
$T = 3$: 2007 (0.99)
$T = 4$: R. Schneider (0.99)
$T = 5$: R. Schneider (0.5), T. Brady (0.5)
Q: When did the movies release whose writers also wrote Birdy? **A**: 1989
$T = 3$: 1989 (0.60), GD. Goldberg (0.11)
$T = 4$: 1989 (0.99)
$T = 5$: 1989 (1.0)

7 Conclusion

Our method (REAREV) introduces a new way to KGQA reasoning with respect to instruction execution and decoding. We improve instruction decoding via adaptive reasoning, which updates the instructions with KG-aware information. We improve instruction execution by emulating the breadth-first search algorithm, which provides robustness to the initial instruction ordering. Experimental results on three KGQA benchmarks demonstrate the REAREV 's effectiveness compared with previous state-of-the-art, especially when the KG is incomplete or when we tackle complex questions.

8 Limitations

Our contributions are on the reasoning part, and our method assumes that we have the linked entities and a question-specific subgraph as input. Although improving entity linking is out of our scope, our approach cannot recover from entity linking errors. However, just like REAREV, all the methods that we compare against make the same assumption and rely on external entity linking tools. The linked entities are obtained as explained in Section 5.2.

Moreover, our method assumes that using a single GNN layer per reasoning step (see Figure 2) is sufficient. This may not be the case when, for example, we need to reason for multiple steps with the same instruction. Our method could benefit from designing a multistep reasoning module.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th international conference on world wide web*.

Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*.

Mattia Atzeni, Jasmina Bogojeska, and Andreas Loukas. 2021. Sqaler: Scaling question answering by decoupling multi-hop and logical reasoning. *Advances in Neural Information Processing Systems*.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*.

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

Nikita Bhutani, Xinyi Zheng, and HV Jagadish. 2019. Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

William W Cohen, Haitian Sun, R Alex Hofer, and Matthew Siegler. 2020. Scalable neural methods for reasoning with a symbolic knowledge base. *arXiv preprint arXiv:2002.06115*.

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

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

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR.

Jiale Han, Bo Cheng, and Xu Wang. 2020. Open domain question answering based on text enhanced knowledge graph with hyperedge infusion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*.

Jiale Han, Bo Cheng, and Xu Wang. 2021. Two-phase hypergraph based reasoning with dynamic relations for multi-hop kbqa. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Drew A Hudson and Christopher D Manning. 2019. Learning by abstraction: The neural state machine. In *Advances in Neural Information Processing Systems*, volume 32.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. *arXiv preprint arXiv:2105.11644*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. Association for Computational Linguistics.

Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019. Knowledge base question answering with topic units. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

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*.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N Ioannidis, Soji Adeshina, Phillip R Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. 2021. Tempoqr: Temporal question reasoning over knowledge graphs. *arXiv preprint arXiv:2112.05785*.

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

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

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

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

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. 2021. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *International Conference on Machine Learning*. PMLR.

Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. 2021. Question answering over temporal knowledge graphs. *arXiv preprint arXiv:2106.01515*.

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

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 *European semantic web conference*. Springer.

Priyanka Sen, Amir Saffari, and Armin Oliya. 2021. Expanding end-to-end question answering on differentiable knowledge graphs with intersection. *arXiv preprint arXiv:2109.05808*.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. *arXiv preprint arXiv:2104.07302*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Advances in neural information processing systems*, 28.

Haitian Sun, Andrew Arnold, Tania Bedrax Weiss, Fernando Pereira, and William W Cohen. 2020. Faithful embeddings for knowledge base queries. *Advances in Neural Information Processing Systems*, 33:22505–22516.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

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

Yawei Sun, Pengwei Li, Gong Cheng, and Yuzhong Qu. 2021. Skeleton parsing for complex question answering over knowledge bases. *Journal of Web Semantics*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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)*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*. PMLR.

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

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Improving question answering over incomplete kbs with knowledge-aware reader. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

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

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qagnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

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

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

A Appendix

A.1 Dataset Statistics

In Table 6, we provide the dataset statistics with the subgraph extraction algorithm described in Section 5.2. We also include the 2-hop MetaQA-2 (Zhang et al., 2018) dataset, which is easier than MetaQA-3.

Table 6: Datasets statistics. “avg.$|\mathcal{V}_q|$” denotes average number of entities in subgraph, and “coverage” denotes the ratio of at least one answer in subgraph.

Datasets	Train	Dev	Test	avg. $\|\mathcal{V}_q\|$	coverage (%)
Webqsp	2,848	250	1,639	1,429.8	94.9
CWQ	27,639	3,519	3,531	1,305.8	64.4
MetaQA-2	118,980	14,872	14,872	469.8	100.0
MetaQA-3	114,196	14,274	14,274	497.9	99.0

A.2 Implementation Details

To encode questions, we use SentenceBERT (Reimers and Gurevych, 2019); although we observe similar results if we use BERT (Devlin et al., 2018) or RoBERTa (Liu et al., 2019). For MetaQA, we use an LSTM encoder (Hochreiter and Schmidhuber, 1997) and initialize tokens with Glove (Pennington et al., 2014).

The number of hidden dimensions d is tuned amongst $\{50, 100\}$. We optimize the model with Adam optimizer (Kingma and Ba, 2014), where the learning rate is set tuned amongst $\{1e^{-4}, 5e^{-4}, 1e^{-4}\}$ and the batch size is tuned amongst $\{8, 16, 40\}$. We tune the number of epochs amongst $\{10, 30, 50, 100, 200\}$. We apply dropout regularization (Srivastava et al., 2014) with probability tuned amongst $\{0.1, 0.2, 0.3\}$. We perform model selection based on the best validation scores. For Webqsp, REAREV achieves a validation score of 78.4% at H@1, and for CWQ, a validation score of 57.4% at H@1.

We implemented REAREV using PyTorch (Paszke et al., 2017), reusing the source code of (Sun et al., 2018; He et al., 2021). Experiments were performed on a Nvidia Geforce RTX-2070 and on a Nvidia Geforce RTX-3090 GPU over 32GB and 128GB RAM machines. Our code is publicly available at https://github.com/cmavro/ReaRev_KGQA.

A.3 MetaQA Results

We provide MetaQA full results in Table 7. We devide methods in two categories: (i) subgraph-based that may not be able to access all possible answers, and (ii) full-graph/retrieval-based that can access all KG facts. Subgraph-based methods performs worse when answers are missing (see MetaQA-3 and Table 6).

Table 7: H@1 performance comparison for MetaQA.

	MetaQA-2	**MetaQA-3**
Subgraph-based with PRN algorithm (Section 5.2)		
GraftNet	94.8	77.7
NSM	**99.9**	**98.9**
NSM-distill	**99.9**	**98.9**
REAREV	**99.9**	**98.9**
Full graph/ Retrieval-based		
KV-Mem	82.7	48.9
EmbedKGQA	98.8	94.8
PullNet	99.9	91.4
EmQL	98.6	99.1
SQALER	99.9	99.9
SQALER+GNN	99.9	99.9
TransferNet	**100**	**100**