SQALER: Scaling Question Answering by Decoupling Multi-Hop and Logical Reasoning

Mattia Atzeni  
IBM Research, EPFL  
Switzerland  
atz@zurich.ibm.com

Jasmina Bogojeska  
IBM Research  
Switzerland  
jbo@zurich.ibm.com

Andreas Loukas  
EPFL  
Switzerland  
andreas.loukas@epfl.ch

Abstract

State-of-the-art approaches to reasoning and question answering over knowledge graphs (KGs) usually scale with the number of edges and can only be applied effectively on small instance-dependent subgraphs. In this paper, we address this issue by showing that multi-hop and more complex logical reasoning can be accomplished separately without losing expressive power. Motivated by this insight, we propose an approach to multi-hop reasoning that scales linearly with the number of relation types in the graph, which is usually significantly smaller than the number of edges or nodes. This produces a set of candidate solutions that can be provably refined to recover the solution to the original problem. Our experiments on knowledge-based question answering show that our approach solves the multi-hop MetaQA dataset, achieves a new state-of-the-art on the more challenging WebQuestionsSP, is orders of magnitude more scalable than competitive approaches, and can achieve compositional generalization out of the training distribution.

1 Introduction

Reasoning, namely the ability to infer conclusions and draw predictions based on existing knowledge, is a hallmark of human intelligence. Infusing the same ability into machine learning models has been a major challenge [34, 29] and has historically required complex systems made of several hand-crafted or learned components [47, 19]. Recently, the paradigm has shifted to deep learning approaches [43, 42], where neural networks are used to reason over structured knowledge or a text corpus. In this work, we assume that the source of knowledge is a structured knowledge graph (KG) and we tackle the problem of knowledge-based question answering (KBQA), namely finding answers to natural language queries involving multi-hop and logical reasoning over the KG.

Answering queries over a knowledge graph involves many challenges, among which scalability is a major issue. Real-world KGs often contain millions of nodes and even a 2-hop neighborhood of the entities mentioned in the query may comprise tens of thousands of nodes. Many state-of-the-art approaches [41, 42, 39] address the challenge of scalability by building small query-dependent subgraphs. To this end, they usually use simple heuristics [41] or, in some cases, iterative procedures based on learned classifiers [42]. This preprocessing step is usually needed because each forward pass in end-to-end neural networks for KBQA scales at least linearly with the number of edges in the subgraph. Training neural networks involves repeated evaluation, which renders even a linear complexity impractical for graphs of more than a few tens thousands of nodes.

In order to address this issue, we introduce a novel approach called SQALER (Scaling Question Answering by Leveraging Edge Relations). The method first learns a model that generates a set of candidate answers (entities in the KG) by multi-hop reasoning: the candidate solutions are obtained by starting from the set of entities mentioned in the question and seeking those that provide an answer by chained relational following operations. We refer to this module as the relation-level model.
We show that this multi-hop reasoning step can be done efficiently and provably generates a set of candidates including all the actual answers to the original question. SQALER then uses a second-stage edge-level model that recovers the real answers by performing logical reasoning on a subgraph in the vicinity of the candidate solutions. A visual summary of our approach is depicted in Figure 1.

The main contributions and takeaway messages of this work are the following:

1. KBQA can be addressed by first performing multi-hop reasoning on the KG and then refining the result with more sophisticated logical reasoning without losing expressiveness (we will elaborate this claim in more details in Section 2.3).

2. Multi-hop reasoning can be accomplished efficiently with a method that scales linearly with the number of relation types in the KG, which are usually significantly fewer than the number of facts or entities.

In the remainder of the paper, we first provide an extensive overview of our approach and a theoretical analysis of the expressive power and the computational complexity of SQALER. Our experimental results show that SQALER achieves better reasoning performance than state-of-the-art approaches, generalizes compositionally out of the training distribution, and scales to the size of real-world knowledge graphs with millions of entities.

2 Scaling KBQA with relation and edge-level reasoning

This section provides a detailed description of our approach. We start by defining the problem formally and giving an intuitive overview of SQALER. Then, we discuss the approach in more details and analyze its computational complexity and expressive power.

Problem statement. We denote a knowledge graph as \( G = (V, R, E) \), where \( v \in V \) represents an entity or node in \( G \), \( r \in R \) is a relation type, and we write \( v \xrightarrow{r} v' \) to denote an edge in \( E \) labeled with relation type \( r \in R \) between two entities \( v, v' \in V \). We extend the same notation to sets of nodes by writing \( V_i \xrightarrow{r} V_j \) if \( V_j = \{ v_j \in V | v_i \xrightarrow{r} v_j, v_i \in V_i \} \). Given a knowledge graph \( G = (V, R, E) \) and a natural language question \( Q \), expressed as a sequence of tokens \( Q = (q_1, q_2, \ldots, q_{|Q|}) \), in knowledge-based question answering the objective is to identify a set of nodes \( A_Q \subseteq V \) representing the correct answers to \( Q \). Following previous work \[41, 42, 43\], we assume that the set of entities mentioned in the question \( V_Q \subseteq V \) is given. These nodes are also called the anchor nodes of the question and in practice are commonly obtained using an entity-linking module.

Overview. KBQA can be cast as an entity seeking problem on \( G \) by translating \( Q \) into a set of nodes \( V_Q \subseteq V \) (the starting points of the search) and seeking for nodes that provide an answer \[41, 42, 43\]. Attempting to find \( A_Q \) directly on \( G \) is prohibitive in practice, as even the most efficient graph-based neural networks generally scale at least linearly with the number of edges. Our approach mitigates this issue by breaking the problem in two subproblems.

(a) We first utilize a relation-level model \( \phi \) to obtain a set of candidate answers \( \hat{A}_Q \), such that \( A_Q \subseteq \hat{A}_Q \). We refer to \( \phi \) as “relation-level” because, as we will see, it operates on the coalesced graph, a simplified representation of \( G \), where edges of the same relation type are coalesced. The coalesced graph is constructed before training and incurs a one-time linear cost. By exploiting it during training, the relation-level model scales with the number of (distinct) relation types in the KG, which are usually significantly fewer than the number of edges or entities.

(b) The candidate answers are then refined using an edge-level model \( \psi \) applied on a subgraph \( G(\hat{A}_Q) \) of the original knowledge graph in the vicinity of \( \hat{A}_Q \). We should note that the refining step is not always necessary. Indeed, we found that a relation-level model is sufficient to perfectly solve tasks like multi-hop question answering \[50\]. Figure 1 shows an overview of our approach.

2.1 Relational coalescing for efficient knowledge seeking

Our approach relies on a relation-level model \( \phi \) that operates as a knowledge seeker in \( G \). The model identifies a node \( v \) as a candidate \( v \in \hat{A}_Q \) based on the sequence of relations that connect it with \( V_Q \).
This can be achieved by using a neural network $\phi$ to predict how likely it is that the correct answer is reached from $V_Q$ by following a sequence of relations $R$.

**Reachability.** To define how our method works, it will help to formalize the concept of reachability. Let $R = (r_1, \ldots, r_{|R|})$ be a sequence of relations. We say that "$v$ is $R$-reachable from $V_Q$" if there exists a path $P = (v_1, \ldots, v_{|R|}, v)$ in $G$ such that:

$$v_1 \in V_Q \quad \text{and} \quad v_i \xrightarrow{r_i} v_{i+1} \quad \text{for every} \quad i = 1, \ldots, |R|.$$

That is, we can reach $v$ by starting from a node in $V_Q$ and following a sequence of edges with relation types $R$. We also denote by $reach_G(V_Q, R)$ the set of nodes that are $R$-reachable from $V_Q$:

$$reach_G(V_Q, R) = \{ v \in V \mid v \text{ is } R\text{-reachable from } V_Q \}.$$

**Relational coalescing.** Given a knowledge graph $G$, a question $Q$, and a set of entity mentions $V_Q$, we consider a representation of the graph $\tilde{G}_Q = (\tilde{V}_Q, \tilde{R}_Q, \tilde{E}_Q)$, which allows us to efficiently compute sets of nodes that are reachable from $V_Q$. We refer to this representation as the question-dependent coalesced KG, because edges with the same relation type are coalesced, as shown in Figure 1. The nodes of $\tilde{G}_Q$ are sets of nodes of $G$ that are reachable from $V_Q$ by following any possible sequence of relations originating from $V_Q$. The graph $\tilde{G}_Q$ has an edge $\tilde{V}_i \leadsto \tilde{V}_j$ if $V_j$ is the set of nodes that are reachable from $V_i$ by following relation $r$. For convenience, we include a relation type $\text{self} \in \tilde{R}_Q$ to denote self loops. We refer the reader to Appendix A for a formal definition of $\tilde{G}_Q$.

The coalesced graph can be precomputed once as a preprocessing step for each question $Q$ and incurs a one-time linear cost. In practice, however, we do not need to compute and store all the nodes in $\tilde{G}_Q$ but only edge labels. This makes learning efficient because each forward/backward pass scales with the number of relation types and does not depend on the number of nodes or edges in the KG.

**Knowledge seeking in $\tilde{G}_Q$.** The coalesced graph allows us to provide approximate answers to input questions in an efficient manner. Specifically, we seek $k \geq 1$ sequences of relations $R^*_i$, such that:

$$\mathcal{A}_Q \subseteq \tilde{A}_Q = \bigcup_{i=1}^{k} reach_G(V_Q, R^*_i).$$

We can achieve this by using a model $\phi$ that only considers relation sequences originating from $V_Q$. The model predicts the likelihood $\phi : \tilde{E}_Q \to [0, 1]$ of following a certain edge in a relation sequence from $V_Q$ to $\tilde{A}_Q$. Then, given $R = (r_1, \ldots, r_{|R|})$ and a node in the coalesced graph $V_Q$, we can compute the likelihood of $R$ by multiplying the likelihood of all edges traversed by $R$ in $\tilde{G}_Q$:

$$P(R \mid Q, \tilde{G}_Q, V_Q) \propto \prod_{i=1}^{|R|} \phi(reach_G(V_Q, R_{1:i-1}), r_i, reach_G(V_Q, R_{1:i})) \mid Q),$$
where \( R_{1\rightarrow i} = (r_1, \ldots, r_i) \) is the subsequence of \( R \) up to the \( i \)-th relation. We generate \( \hat{A}_Q \) by selecting the top \( k \) relation sequences \( R^*_k \) with maximum likelihood \( P(R^*_k | Q, \hat{G}_Q, V_Q) \). This can be done by an efficient search algorithm, such as beam search starting from \( V_Q \). Then, we compute \( \hat{A}_Q \) as the union of all target nodes of the selected relation sequences. More details about the knowledge-seeking algorithm are provided in Appendix [B].

### 2.2 Refining the solution on the original KG

In certain cases, like multi-hop question answering [50], the set of candidate answers \( \hat{A}_Q \) may already be a reasonable estimate of \( A_Q \). We will substantiate this claim experimentally in Section 4. In general, however, we recover \( A_Q \) by using an edge-level model \( \psi \) applied on a subgraph \( G(\hat{A}_Q) \) of \( G \). Specifically, we construct \( G(\hat{A}_Q) \) as the subgraph induced by the set of nodes \( V(\hat{A}_Q) \), which includes all nodes visited when following the top-\( k \) relation sequences along with their neighbors (see Figure [1] for an example). Any existing method for KBQA can be used to instantiate \( \psi \) by running it on \( G(\hat{A}_Q) \) rather than \( G \). We opted to use a Graph Convolutional Network (GCN) conditioned on the input question with the same architecture as in [41]. The edge-level model is constrained to predict an answer among the candidates generated by the relation-level model.

### 2.3 Analysis of scalability and expressive power

This section provides a scalability analysis of our approach and shows that the relation-level model scales linearly with the number of relation types in the graph. Then, we analyse the expressive power of SQALER and we show the class of supported logical queries.

**Computational complexity.** As mentioned, we do not evaluate the likelihood \( \phi \) for all edges in \( G_Q \), but we generate the most likely relation sequences using a knowledge-seeking procedure based on the beam search algorithm. At any given time step, only the \( \beta \) most likely relation sequences are retained and further explored at the next iteration. Hence, the time complexity required by our algorithm is \( O(\tau_{\text{max}} \cdot \beta \cdot d_{\text{max}}(G_Q)) \), where \( \tau_{\text{max}} \) is the maximum allowed number of decoding time steps and \( d_{\text{max}}(G_Q) \) is the maximum outdegree of \( G_Q \). Note that \( d_{\text{max}}(G_Q) \) is bounded by the number of relations in the graph, whereas \( \tau_{\text{max}} \) and \( \beta \) are constant parameters of the algorithm and are usually small. This gives a time complexity of:

\[
O(\tau_{\text{max}} \cdot \beta \cdot |R|) = O(|R|).
\]

Hence, the knowledge-seeking algorithm scales linearly with the number of relations in the KG. The space complexity is also \( O(\tau_{\text{max}} \cdot \beta \cdot |R|) \). A more detailed analysis is provided in Appendix [B].

**Expressive power.** Given a natural language question \( Q \), we can represent the inferential chain needed to obtain \( A_Q \) from \( V_Q \) as a logical query \( Q \) on \( G \). As an example, the question in Figure 2, “Who starred in films directed by George Lucas?”, can be represented by the logical query:

\[ Q[V_2] = V_2, \forall V : \text{Directed}(\text{George_Lucas}, V) \land \text{Starred}(V, V_2). \]

We denote with \( V_t \) the target variable of the query and we say that \( v \in V \) satisfies \( Q \) if \( Q[v] = \text{True} \). A query \( Q \) is an existential positive first-order (EPFO) query if it involves the existential quantification (\( \exists \)), conjunction (\( \land \)), and disjunction (\( \lor \)) of literals corresponding to relations in the KG. Each literal is of the form \( r(V, V') \), where \( V \) is either a node in \( V_Q \) or an existentially quantified bound variable, and \( V' \) is either an existentially quantified bound variable or the target variable. A literal \( r(V, V') \) is satisfied if \( V \models r(V, V') \), for \( r \in R \). Any EPFO query can be represented in disjunctive normal form (DNF) [12], namely as a disjunction of conjunctions. Note that, we do not consider queries with universal quantification (\( \forall \)), as we assume that in real-world KGs no entity connects to all the others. Then, the following proposition holds for any knowledge graph and EPFO query.

**Proposition 1.** Let \( G = (V, R, E) \) be a knowledge graph and \( V_Q \subseteq V \) denote a set of entities in \( G \). Let \( Q \) be a valid existential positive first-order query on \( G \) and let \( n_V \) be the number of disjunction operators in the disjunctive normal form of \( Q \). Then, there exist \( k \leq n_V + 1 \) sequences of relations \( R^*_k \in R^* \) such that:

\[
A_Q \subseteq \bigcup_{i=1}^{k} \text{reach}_G(V_Q, R^*_i),
\]
where $A_Q = \{v \in V \mid Q[v] = \text{True}\}$ is the denotation set of $Q$, namely the entities satisfying $Q$.

This shows that sampling $n_\psi + 1$ sequences of relations allows generating a set of candidate answers $\hat{A}_Q$ that does not miss any of the real answers $A_Q$. Then, assuming that the edge-level model $\psi$ can recover $A_Q$ from $\hat{A}_Q$, our approach can be used to answer any EPFO query on $G$. More details about the expressive power of SQALER and the proof of Proposition 1 are provided in Appendix C.

## 3 Architecture of the relation-level model

For the relation-level model $\phi$, we propose an auto-encoder, where the the decoder is constrained to follow sequences of relations in the coalesced representation $G_Q$. We train the network with weak supervision, assuming that a sequence of relations is correct if it reaches a set of candidate answers $\hat{A}_Q$ that is the smallest reachable superset of the $A_Q$. We found it useful to pretrain the model in order to infuse knowledge from the KG. In this case, we train the model to predict a path in the KG, given the representations of the source and target nodes. More details about training strategies are given in Appendix D.

The architecture of the model (see Figure 2) includes three main components: a question encoder, a relation encoder and a graph-guided decoder. We explain each one below.

**Question encoder.** The encoder receives as input a natural language question, which comprises a sequence of tokens $Q = (q_1, q_2, \ldots, q_{|Q|})$. The question is encoded using a pre-trained BERT model and processed with the same positional encoding technique used in [44]. The resulting embeddings are then fed into $n_t = 3$ transformer encoder layers [44]. This results in a matrix $Q \in \mathbb{R}^{|Q|+1 \times d_{\text{model}}}$, where the first row vector is an overall representation of the whole query $Q$ (derived from the embedding of the [CLS] token introduced by BERT) and each remaining row represents the final $d_{\text{model}}$-dimensional encoding of a token in the input question.

**Relation encoder.** The relation encoder produces a representation $r \in \mathbb{R}^{d_{\text{model}}}$ for each relation type $r \in R$. We decided to encode relations based on their surface form, with the same pre-trained BERT model used in the question encoder. In this case, only the embedding of the [CLS] token is used in order to get the final representation $r$ of each relation type $r \in R$. At inference time, or in
case the BERT model is not fine-tuned, the embeddings of the relations can be precomputed as a preprocessing step to improve the efficiency of the approach.

**Graph-guided decoder.** The decoder’s job is to predict a sequence of relations leading from \( V_0 \) to \( V_n \) in \( \hat{G}_Q \). At any time step \( t \), it receives as input a sequence of relations \( R_t = (s_{self}, r_1, \ldots, r_{t-1}) \) and predicts the next relation \( r_t \) (\( s_{self} \) is used as a special token to denote the start of decoding). Note that the input sequence uniquely determines a node \( V_i \) in the graph \( \hat{G}_Q \), namely the node reachable from \( V_0 \) by following \( R_t \). The decoder thus selects \( r_t \) by choosing amongst the outgoing edges \( \hat{E}_t \) of \( V_i \). We use the same number of layers \( n_l \) both for the question encoder and the decoder. Let \( X^l_t = [x^l_0, \ldots, x^l_{t-1}]^T \in \mathbb{R}^{E \times \text{model}} \) denote the hidden state of the \( l \)-th layer of the decoder preceding time step \( t \). Note that \( X^0_t \) is the representation of the sequence \( R_t \), obtained by using the relation encoder described above and the same positional encoding technique used in the question encoder. For each decoder layer, we perform self-attention over the target sequence \( X \) by computing:

\[
x^l_t = \text{Attention}(x^l_t, X^l_t, X^l_t),
\]

where \( \text{Attention} \) is a function that performs multi-head scaled dot-product attention [44] with skip connections and layer normalization [4]. The above step allows each relation in the decoded sequence to attend to all the others predicted up to time step \( t \). We then let the result attend to the question as:

\[
x^{Q,l}_t = \text{Attention}(x^{l}_{Q,t}, Q, Q).
\]

This is done in order to update the current state of the decoder based on the input question. Next, let \( R_t \in \mathbb{R}^{E \times \text{model}} \) denote the encoding of the relations labeling all edges in \( \hat{E}_t \). We constrain the decoded sequence to follow the structure of the graph by attending only to valid relations as follows:

\[
x^{R,l}_t = \text{Attention}(x^{Q,l}_t, R_t, R_t).
\]

We get the hidden state of the next layer \( x^{l+1}_t \) by processing the result with a feed forward network. The model outputs a categorical distribution \( \phi(e \mid Q) \in [0, 1] \) over the edges \( e \in \hat{E}_t \), by applying a softmax function as follows:

\[
\phi(V_i \xrightarrow{r} V_j \mid Q) = \frac{\exp(r^T x^{n_i}_t)}{\sum_{V_i \xrightarrow{r'} V_j \in \hat{E}_t} \exp(r'^T x^{n_i}_t)},
\]

where \( x^{n_i}_t \) is the output of the final layer of the decoder, whereas \( r \) and \( r' \) denote the representations of relations \( r \) and \( r' \) respectively.

### 4 Experiments

This section presents an evaluation of our approach with respect to both reasoning performance and scalability. We first show that **SQALER** reaches state-of-the-art results on popular KBQA benchmarks and can generalize compositionally out of the training distribution. Then, we demonstrate the scalability of our approach on KGs with millions of nodes. We refer the reader to Appendix [E] for more details about the experiments.

#### 4.1 Experimental setup

**Datasets.** We evaluate the reasoning performance of our approach on **MetaQA** [50] and **WebQuestionsSP** [49]. **MetaQA** includes multi-hop questions over the WikiMovies KB [45] and we consider both 2-hop (MetaQA 2) and 3-hop (MetaQA 3) queries. **WebQuestionsSP** (WebQSP) comprises more complex questions answerable over a subset of Freebase [21], a large KG with millions of entities. We further assess the compositional generalization ability of **SQALER** on the **Compositional Freebase Questions** (CFQ) dataset [28]. Each question in CFQ is obtained by composing primitive elements (atoms). Whereas the training and test distribution of atoms are similar, the test set contains different compounds, namely new ways of composing these atoms. CFQ comprises three dataset splits (MCD1, MCD2, and MCD3), with maximal compound divergence (MCD) between the training and test distributions. We refer the reader to Appendix [E] for an extensive description of the datasets.
**Evaluation protocol.** In our experiments on *MetaQA* and *WebQuestionsSP*, we assess the performance of three variants of our approach: (a) a version that only makes use of the relation-level model without the refinement step (**SQALER – Unrefined**), (b) a model that utilizes a key-value memory network to identify the correct answers from the candidates (**SQALER – KV-MemNN**), and (c) a model that uses a GNN architecture for the refinement step (**SQALER – GNN**), as explained in Section 2.2. Following previous work [41, 42, 43, 39], we evaluate the models based on the *Hits@1* metric. On the *CFQ* dataset, we evaluate the accuracy of the refined model with the GNN based on whether it predicts exactly the same answers given by the corresponding SPARQL query.

**4.2 Main results**

**KBQA Performance.** Table 1 summarizes the results of our experiments on the two benchmark datasets. For the two multi-hop *MetaQA* datasets, we achieve state-of-the-art performance by only using the relation-level model of **SQALER**. As shown in Table 1, **SQALER** outperforms all the baselines on *MetaQA 3*, demonstrating the ability of our approach to perform multi-hop reasoning over a KG. For the more complex questions in the *WebQuestionsSP* dataset, the unrefined **SQALER** model achieves better performance than all but one (**EmQL**) of the baselines. To achieve such performance, however, EmQL creates a custom set of logical operations tailored towards the specifics of the target KG and the kind of questions in the dataset, while our approach is agnostic with respect to such details. Combining the relation and edge-level models improves the performance on *WebQSP*. In particular, **SQALER – GNN** outperforms all considered baselines on the three datasets.

<table>
<thead>
<tr>
<th></th>
<th>MetaQA 2</th>
<th>MetaQA 3</th>
<th>WebQSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>KV-MemNN [35]</td>
<td>82.7</td>
<td>48.9</td>
<td>46.7</td>
</tr>
<tr>
<td>GRAFT-Net [41]</td>
<td>94.8</td>
<td>77.7</td>
<td>70.3</td>
</tr>
<tr>
<td>ReifKB + mask [10]</td>
<td>95.4</td>
<td>79.7</td>
<td>52.7</td>
</tr>
<tr>
<td>PullNet [42]</td>
<td>99.9</td>
<td>91.4</td>
<td>69.7</td>
</tr>
<tr>
<td>EmbedKGQA [39]</td>
<td>98.8</td>
<td>94.8</td>
<td>66.6</td>
</tr>
<tr>
<td>EmQL [43]</td>
<td>98.6</td>
<td>99.1</td>
<td>75.5</td>
</tr>
<tr>
<td><strong>SQALER – Unrefined</strong></td>
<td>99.9</td>
<td>99.9</td>
<td>70.6</td>
</tr>
<tr>
<td><strong>SQALER – KV-MemNN</strong></td>
<td>99.9</td>
<td>99.9</td>
<td>72.1</td>
</tr>
<tr>
<td><strong>SQALER – GNN</strong></td>
<td>99.9</td>
<td>99.9</td>
<td>76.1</td>
</tr>
</tbody>
</table>

**Compositional generalization.** In order to evaluate the compositional generalization ability of **SQALER**, we performed additional experiments on the *CFQ* dataset. Table 2 shows the accuracy on the three MCD splits and the mean accuracy (MCD-mean) in comparison to the other methods in the leaderboard. Note that the other approaches address a semantic parsing task and require additional supervision, as they are trained to predict the target query. On the other hand, we aim to predict directly the set of answers to the input question. The experiment shows that **SQALER** is able to achieve compositional generalization with an accuracy comparable to the state-of-the-art model on *CFQ* for semantic parsing.

**Subgraph extraction.** We analyzed the candidate solutions produced by the relation-level model in order to evaluate the suitability of our approach to building small question subgraphs that are likely to contain the answers to a natural language question. For this purpose, we computed the precision and recall of the set of candidate answers with varying number of relation sequences sampled by the relation-level model. Figure 3 shows the top relation sequences predicted by the relation-level model on two questions from the test set of *WebQuestionsSP*. The precision and recall curves are shown in Figure 4. As expected, on *MetaQA* the recall is high for all values of k, because selecting the most likely sequence of relations is sufficient to solve the multi-hop question answering task. On *WebQuestionsSP*, only 3 sequences of relations are sufficient to obtain a recall of 0.91, and we can improve it to 0.95 by generating still small subgraphs consisting of only 10 sequences of relations.
What town was Martin Luther King assassinated in?

![Diagram](image)

**Figure 3:** Attention weights given by the relation-level model to the edges of the coalesced graph for two questions in *WebQuestionsSP*. Thicker and darker edges represent higher attention weights.

### 4.3 Efficiency and scalability

We analyze the efficiency of our approach on synthetic KBs (as in [9, 10]) and then compare the scalability of different preprocessing methods on the KGs of *MetaQA* and *WebQuestionsSP*. First, we perform experiments on KBs where the relational coalescing has no effect: the outdegree of each node is equal to the number of relation types and all edges originating from a node have different relation labels. We perform two experiments on such KBs. In the first one (Figure 5a), the number of relation types is fixed to $|R| = 10$ and the number of entities varies from $|V| = 10^2$ to $|V| = 10^6$. In the second task (Figure 5b), the number of entities is fixed to $|V| = 5000$ and the number of relations varies from $|R| = 1$ to $|R| = 10^3$. The single answer node is always two-hops away from the entities mentioned in the question. We compare *SQALER* (unrefined) against a GNN-based approach (*GRAFT-Net* [41]) and a key-value memory network (*KV-MemNN* [35]). The approaches are evaluated based on the queries per second at inference time with a mini-batch size of 1. The results show that increasing the number of entities has negligible impact on the performance of *SQALER*, whereas *GRAFT-Net* and the key-value memory network are limited to graphs with less than 10k nodes. This shows that, in large KGs like Freebase, the baselines would not be able to handle even a 2-hop neighborhood of the entities mentioned in the question (we refer the reader to Appendix E.5 for more details). Finally, from the results in Figure 5b we see that the throughput of our approach decreases with the number of relation types. However, in practice, we can leverage the GPU to score the edges of the graph in parallel. This is why we observe only a minor drop in performance when the number of relation types grows from $|R| = 1$ to $|R| = 100$.

In order to assess the scalability of the proposed relational coalescing operation, we further compare commonly used preprocessing methods on the KG of *WebQuestionsSP*. We evaluate the time required to extract complete 2-hop neighborhoods of the entities mentioned in the question and the time to perform Personalized Page Rank (PPR) on such graphs. The results are shown in Figure 5c. Note that, at inference time, we can perform the coalescing only on the portion of the graph explored by the model, which makes *SQALER* much more efficient. At training time, the preprocessing is comparable to the 2-hop neighborhood extraction. Finally, Figure 5d shows the performance of the models with the respective preprocessing step at inference time on synthetic KBs with growing number of edges.
4.4 Incomplete knowledge graphs

In order to evaluate the capability of our approach to cope with missing information in the knowledge graph, we performed two additional experiments. In the first experiment, we evaluated our approach (the \textsc{sqaler} – \textsc{gnn} variant) on \textit{WebQuestionsSP} using incomplete knowledge graphs with only 50% of the original edges (50% KG). Then, following previous work \cite{41, 42}, we tried to mitigate the missing information using additional sources of external knowledge. In particular, for each question, we used the same text documents extracted from Wikipedia as done by Sun et al. \cite{41} (50% KG + Text). In this experiment, the relation-level model is unaware of the additional source of knowledge, but the information from the text documents is infused into the edge-level GNN with the same strategy used in \textsc{graft-net} \cite{41} (note that this makes the edge-level GNN-based model essentially equivalent to the full version of \textsc{graft-net}, with both KG and text support). We compare our approach against \textsc{graft-net} and \textsc{pullnet}, namely the two baselines designed for open-domain question answering with incomplete KGs and text documents.

The results of the experiments are reported in Table \ref{tab:table3}. We observe that, despite not being designed for incomplete KGs, \textsc{sqaler} outperforms the baselines on both experimental settings. This is not surprising, as \textsc{graft-net} relies on a simple heuristic process to construct question subgraphs and \textsc{pullnet} is constrained to follow the structure of the incomplete graph, because its iterative retrieval process can only expand nodes that are reachable from the set of anchor entities. This means that, in principle, any node retrieved by \textsc{pullnet}’s iterative process can also be reached by \textsc{sqaler}’s relation-level model. Similarly to the baselines, we note only a minor gain in performance when using the text documents as an additional source of information.

Table 3: Hits@1 on \textit{WebQuestionsSP} with incomplete KGs (50% of the edges) and additional text

<table>
<thead>
<tr>
<th></th>
<th>50% KG</th>
<th>50% KG + Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textsc{graft-net} \cite{41}</td>
<td>48.2</td>
<td>49.9</td>
</tr>
<tr>
<td>\textsc{pullnet} \cite{42}</td>
<td>50.3</td>
<td>51.9</td>
</tr>
<tr>
<td>\textsc{sqaler} – \textsc{gnn}</td>
<td>53.5</td>
<td>55.2</td>
</tr>
</tbody>
</table>
5 Related work

Several lines of research in the past few years have focused on introducing deep learning approaches aimed at reasoning over structured knowledge. In particular, this paper is closely related to methods for learning to traverse KGs [14, 13, 24] and recent works on answering conjunctive queries using deep learning approaches [25, 16]. In this context, several KB and query embedding methods have been proposed [45]. Many KB embedding approaches support the same operation performed by our relation-level model, namely relation projection [10, 43, 25, 38]. Some KB embedding methods also explicitly learn to follow chains of relations and traverse KGs [24, 30, 13]. Notably, Query2Box [38] is a query embedding method that represents sets using box embeddings and the more recent beta embeddings [57] extend the framework to support a complete set of first-order logic operators. The main difference with our model is that these methods operate on vector space, whereas our approach is constrained on the graph structure and learns to traverse the KG while keeping the ability to scale to large graphs. Also, our method answers questions in natural language, while the above methods are primarily designed for query answering. Recently, Sun et al. [43] introduced EmQL, a query embedding method which has also been integrated in a question answering model.

Other lines of research on KBQA have focused on unsupervised semantic parsing [3, 2, 1] or on the introduction of supervised models, like graph neural networks (GNNs) designed for reasoning over knowledge graphs [41, 42, 48]. These approaches pose the KBQA problem as a node classification task. For this reason, they have been applied successfully only on small query-dependent graphs. Cohen et al. [10] addressed the problem of creating a representation of a symbolic KB that enables building neural KB inference modules that are scalable enough to perform non-trivial inferences with large graphs. Another recent work [39] has explored using KG embeddings for question answering and handle incompleteness in the KG.

In our work, we combine relation projection with an edge-level GNN to address the KBQA problem. The same idea of combining GNNs with relational following was introduced in Gretel [11], which learns to complete natural paths in a graph given a path prefix. Also, our idea of accelerating GNNs by operating on a reduced graph representation has strong connections with graph coarsening and sparsification [32, 33, 6].

Methods based on reinforcement learning (RL) have also been proposed to perform multi-hop reasoning over knowledge graphs. Xiong et al. [46] proposed DeepPath, which relies on a policy-based agent that learns to reason over multi-hop paths by sampling relations at each step. Also, Das et al. [14] introduced MINERVA, a RL agent that learns how to navigate the graph conditioned only on an input entity and on a query. These approaches are designed for simple query answering and KB completion rather than KBQA. A main difference with our work is that SQALER samples multiple paths and employs an edge-level model to reach higher expressivity.

6 Conclusion

This paper introduced SQALER, a scalable approach to reasoning and question answering over KGs. Our method is expressive and can reach state-of-the-art performance on widely used and challenging datasets. Further, SQALER scales with the number of (distinct) relation types in the graph and can effectively handle large-scale knowledge graphs with millions of entities. Our empirical evaluation also showed that our approach can generalize compositionally and that it can be used to generate question-dependent subgraphs that strike a good trade-off between precision and recall.

Overall, our work proposes an improvement to existing KBQA technology which carries impact to several practical applications. Nevertheless, we remind that the deployment of such models needs to be done cautiously. KBQA replaces a mature technology (traditional KBs and query languages) with less understood methods. The underlying KB may be incomplete, contain misinformation or biases that could negatively affect the decisions of the learned model. We hope that our work will spur further research in this area and contribute to the development of reliable KBQA systems.

Acknowledgments and Disclosure of Funding

Andreas Loukas would like to thank the Swiss National Science Foundation for supporting him in the context of the project “Deep Learning for Graph Structured Data”, grant number PZ00P2 179981.
References


