PerKGQA: Question Answering over Personalized Knowledge Graphs

Ritam Dutt*
Carnegie Mellon University
rdutt@andrew.cmu.edu

Kasturi Bhattacharjee
AWS AI Labs
kastb@amazon.com

Rashmi Gangadharaiah
AWS AI Labs
rgangad@amazon.com

Dan Roth
AWS AI Labs
University of Pennsylvania
drot@amazon.com

Carolyn Penstein Rosé
Language Technologies Institute
Carnegie Mellon University
cp3@andrew.cmu.edu

Abstract

Previous studies on question answering over knowledge graphs have typically operated over a single knowledge graph (KG). This KG is assumed to be known a priori and is leveraged similarly for all users' queries during inference. However, such an assumption does not translate to real-world settings, such as healthcare, where one needs to handle queries of new users over unseen KGs during inference. Furthermore, privacy concerns and high computational cost render it infeasible to query the single KG that has information of all users while answering a specific user's query. This motivates our setting of question answering over personalized knowledge graphs (PerKGQA) where each user has restricted access to their own KG. We observe that current state-of-the-art KGQA methods that require learning prior node representations fare poorly in this setting. We propose two complementary approaches, PathCBR and PathRGCN, to tackle PerKGQA. The former is a simple non-parametric technique that employs case-based reasoning, while the latter is a parametric approach using graph neural networks. Our proposed methods circumvent learning prior representations, can generalize to unseen KGs, and outperform strong baselines on an academic and an internal dataset by 6.5% and 10.5%.

1 Introduction

The task of Question Answering over Knowledge Graphs (KGQA), involves answering a natural language question by querying a predefined knowledge graph (KG), such as WikiData or Freebase. Progress in KGQA research has addressed several challenges, such as answering complex questions, multi-hop reasoning, (Lan and Jiang, 2020; Ren et al., 2021), conversational KGQA (Kacupaj et al., 2021), and multi-lingual KGQA (Zhou et al., 2021), and has also found applications in tax, insurance, and healthcare (Lüdemann et al., 2020; Huang et al., 2021; Park et al., 2020).

To date, most KGQA research has focused on generalizable or generic knowledge, which assumes there is a predefined global KG for all queries. This licenses the assumption that nodes used during inference were already defined in the KG during training. While this assumption holds true for cases which focus on generalizable knowledge, it renders past research unsuitable for scenarios where the KG is not known a priori. In this work, we propose approaches that circumvent the need to make such an assumption.

Furthermore, using a single global KG to handle queries of different users raises additional concerns, especially when a user's query requires situated knowledge such as personal information.

- **Scalability:** The massive size of the global KG make it computationally infeasible to apply sophisticated neural architectures over it.
- **Privacy:** Unfettered access to information of all individuals raises ethical or legal concerns.

In this paper, we formulate PerKGQA or question answering over personalized knowledge graphs for users. Here the user has access to their individual KG, a subset of the global KG, that contains only the information relevant to the user. We are restricted to the user's KG to answer their queries, both during training and inference. Such a setting addresses the aforementioned challenges of scalability, privacy, and generalizing over unseen KGs.

PerKGQA appears deceivingly simple in conception since we are restricted to a subset of the larger global KG. One can claim that our setting is similar to the KGQA subtask where subgraphs and questions are predefined and thus traditional KGQA methods are applicable. However, information retrieval based KGQA methods employs knowledge graph completion techniques like TransE (Bordes et al., 2013), to learn node rep-

* Work done during internship at Amazon AWS AI Labs.

resentations over the global KG and reuse them during inference. Alternately, other approaches leverage additional information in the form of semantic parses, logical forms, and query graphs to answer queries.

This sets PERKGQA apart because we lack access to any prior information, be it text, semantic parses, or prior representations of KG nodes. Our setting requires learning node representations from scratch for each KG to handle unknown entities during inference. Moreover other challenges prevalant in KGQA settings, namely multi-hop reasoning, or answering complex/constraint-based questions are also applicable to PERKGQA. To the best of our knowledge, we are the first to address the challenges of KGQA over unseen KGs in the absence of other information.

We propose two approaches, PATHCBR and PATHRGCN, that are well-suited to the realistic settings above. PATHCBR is a simple non-parametric case-based reasoning approach that encodes path information of past queries to answer a new query. PATHRGCN is a parametric approach that employs graph neural networks, path information, and the KG's structure to extract answers. These approaches circumvent the need for learning prior node representations and can be readily applied to unseen KGs.

Contributions of the paper:

- We formulate PERKGQA, a new setting for KGQA where we operate over unseen KGs in absence of any additional information. We observe that SOTA methods that need to learn underlying node embeddings fare poorly.
- To encourage research, we modify an existing academic dataset (Yih et al., 2016) and make it available to the research community (as Mod-WebQSP).
- Our proposed approaches, PATHCBR and PATHRGCN, outperform strong baselines on Mod-WebQSP and an internal dataset by 6.5% and 10.5% respectively.

2 Preliminaries

2.1 Task Formulation

A Knowledge Graph (KG) is represented as $\mathcal{K} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of entities in the KG, $\mathcal{R}$ represents the set of relations, and $\mathcal{E}$ is the set of triplets (e_1, r, e_2), where $e_1, e_2 \in \mathcal{V}$ and $r \in \mathcal{R}$. Thus $\mathcal{E} \subset (\mathcal{V} \times \mathcal{R} \times \mathcal{V})$. Given a natural language question q, the objective of KGQA is to retrieve answer entities from $\mathcal{V}$.

For PERKGQA, we treat each question as posed by a separate user, and each question is associated with its corresponding knowledge graph, $\mathcal{K}_q$. A given $\mathcal{K}_q$ has a subset of nodes, $\mathcal{V}_q$ and relations, $\mathcal{R}_q$. Two knowledge graphs, $\mathcal{K}_q$ and $\mathcal{K}_{q^*}$ associated with questions q and q^* can have a varying degree of overlap, even being distinctly different.

2.2 Running Example

We now demonstrate the applicability of PERKGQA for a cloud service provider (e.g. Microsoft Azure) in Figure 1. Here, users (blue and red) can create cloud resources (yellow) which are indexed by a unique system identifier. These resources have a corresponding user-specific tag (green), are located in a specific region (orange), and have predefined services deployed on them (purple). The entire system can be envisioned as a knowledge graph (called CloudKG) where nodes represent concepts (users and services) and edges define the relations between concepts. Due to confidentiality, user names are replaced with anonymous identifiers, while concept and relation names in CloudKG are modified. The underlying schema is unchanged.

Deploying a chatbot-based assistant that performs QA over CloudKG would facilitate use, especially by novice users. It would enable users to navigate the system and glean information by posing natural language questions. In Figure 1, when User 101 asks "Which resources have nlp-serv and demo_1 tags?", the system is expected to answer "res_1, res_3". We refer to Figure 1 as a running example in subsequent sections. As new users become a part of CloudKG, the QA system should accommodate their requests over the corresponding KG without any retraining. KGQA approaches that operate upon the entire CloudKG would be computationally infeasible due to the massive size of the user-base [1]. Moreover, the approach should be privacy-preserving wherein a given user's information is not revealed to another.

3 Datasets

We operate on two datasets: an internal dataset called CloudKGQA build on top of CloudKG, and an academic dataset called Mod-WebQSP which was designed to mimic our setting. An instance

[1] https://www.statista.com/statistics/321215/global-consumer-cloud-computing-users/

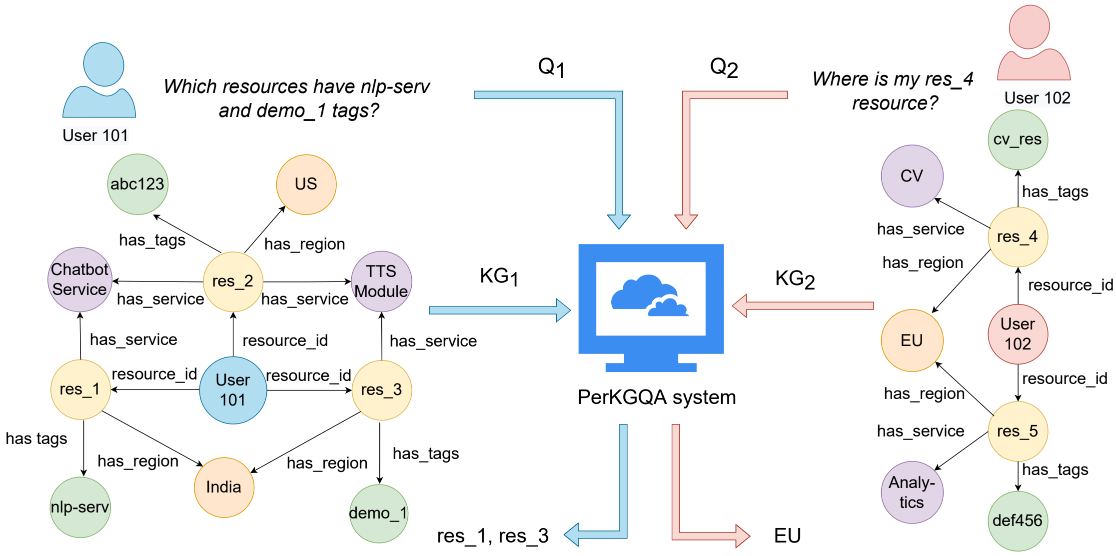


Figure 1: PERKGQA for a cloud service provider setting. The two users (in blue and red) create cloud resources (in yellow) in specific regions (in orange), and deploy services e.g. *Chatbot service*, or *Analytics* (in purple) on them. The users assign customized tags (in green) to the resources. Each user has their unique KG. The system should scale to support queries of new users over unseen KGs without any retraining or additional knowledge.

in either dataset follows the same task formulation in Section 2, namely, for each question q there exists a corresponding KG, K_q which contains all the necessary information. In addition, each question q is associated with one or more source entities; these correspond to nodes in the K_q linked through salient mentions of entities in q. E.g., the source entity for, "Who was responsible for Lincoln's assassination?", is the node corresponding to Abraham Lincoln.

3.1 CloudKGQA

The internal dataset, which we refer to as CloudKGQA, entails question-answering of a customer's queries on their respective cloud resources. We refer the readers to Figure 1 as we present examples that outline the key characteristics of CloudKGQA.

- **Multiple Answers:** A question can have one or more correct answers.
- **Varying Complexity:** A question can either be simple or complex.
  (i) **Simple:** The question can be answered by a single-hop relation, e.g. "Which resource has the tag nlp_serv?"
  (ii) **Complex :** The question involves logical operations like union or intersection e.g."Show me resources in US and India" or contains multiple constraints, e.g. "Which resource has the TTS and MongoDB service and is located in US?" has three constraints, TTS, MongoDB, and US.
- **Multi-Hop distance:** The distance between the source entities and the answers is variable (e.g., the hop-distance for "Show me tags for resources in US" is 2 in Figure 1).
- **Variable graph size:** The size of the KG varies in terms of the number of nodes, edges, and relations for each question.
- **Unseen nodes:** Nodes in the KG seen during inference might not be encountered while training.

3.2 Modified WebQSP (Mod-WebQSP)

We also operate on the publicly-available WebQSP dataset (Yih et al., 2016), built over the Freebase KG $\mathcal{F}$. We chose WebQSP since it shares the first three properties of CloudKGQA, namely multi-answer, multi-hop, simple and complex questions. To completely mimic our setting, we construct a KG, $\mathcal{F}_q$ for each question q, with the caveat that a significant fraction of nodes remain unseen during inference. We describe our process for creating individual KG in the Appendix A.1. Our modification achieves a low overlap of 4.0% between entities across training and test splits, implying that 96% of entities remain unseen during inference.

3.3 Differences between the datasets

We present the descriptive statistics of the two datasets in Table 1 corresponding to the mean number of nodes, edges, relations, answers, and hops for a KG. We also depict the degree of overlap between nodes in training and test splits. The number of instances in CloudKGQA and Mod-WebQSP

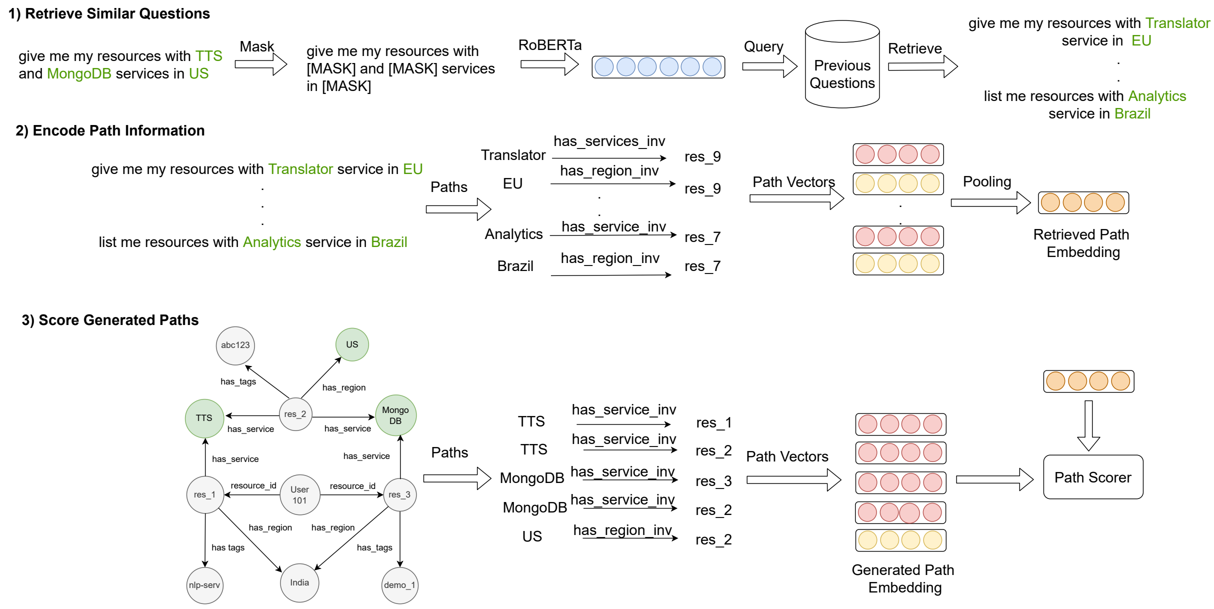


Figure 2: PATHCBR Overview: (1) Retrieve questions similar to a given query template from set of questions; (2) Encode path information as a path embedding; (3) Score generated paths using the retrieved path embedding.

are 800 and 4468 respectively. Moreover, for both datasets, we split the data into train, development, and test in the ratio of 8:1:1.

We observe that CloudKGQA is comparatively smaller in size, had significantly fewer relations, but had longer reasoning chains. Moreover CloudKGQA had more complex questions, in terms of logical operations and multiple-constraints. Specifically, CloudKGQA had one or more source entities for each question, q, whereas Mod-WebQSP had only one source entity. The KGs in CloudKGQA had a similar underlying schema; different KGs had the same set of relations but different entities. However the questions in the test data, had distinct question templates from those during training as seen in Figure 2. The Mod-WebQSP dataset, on the other-hand, had KGs with different relations, but questions during test were similar to those asked during training. We chose these two datasets because they capture two different scenarios.

Dataset	CloudKGQA	Mod-WebQSP
Nodes	23.39	518.21
Edges	35.59	1334.10
Relations	8.00	36.20
Answers	1.99	4.94
Hops	1.75	1.36
Overlap	3.21%	4.01%

Table 1: A brief overview of the statistics of the two datasets, CloudKGQA and Mod-WebQSP. We present the mean number of nodes, edges, relations, answers, and hops, and the overlap between nodes during test and train.

4 Methodology

We propose two approaches to handle PERKGQA. The former is a non-parametric case-based reasoning approach that does not require training, and the latter is a deep neural architecture that employs graph convolutions and encodes structural and path information for reasoning.

4.1 PATHCBR

PATHCBR is a non-parametric approach and comprises the following steps as shown in Figure 2.

(i) **Query Retrieval:** For a query, q, we first retrieve similar questions from the available training set. We consider questions to be similar if they share similar answer types with the query rather than the entities (Das et al., 2020). We perform Named Entity Recognition (NER) to identify text-spans that correspond to source entities in the K_q (Sun et al., 2019; Wang et al., 2020b). We substitute the extracted text-span with a special [MASK] token, yielding the masked query template. We hypothesize that masking entities can help us learn the association of the entity with the template and help generalize to unseen entities. We employ a

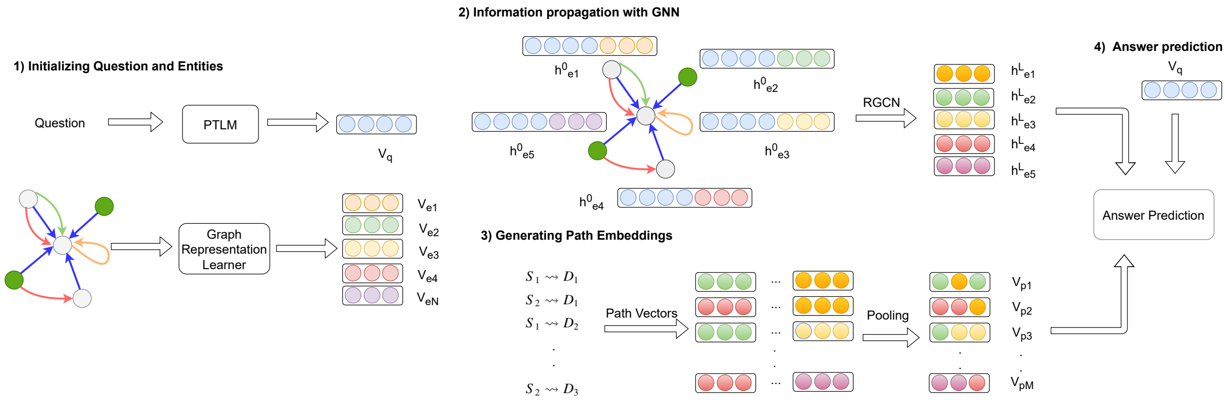


Figure 3: PATHRGCN Overview: (1) Initialize the question using a pretrained language model (PTLM) and the nodes in the corresponding KG; (2) Perform information propagation using RGCN to update node embeddings; (3) Encode path information from the source entities (shown in green) to all possible target nodes by pooling over the constituent node embeddings; (4) Perform answer prediction at both the path and node level.

pretrained language model to create a contextualized embedding of this masked query and retrieve questions whose embeddings have high cosine similarity with the corresponding query embedding. Thus for a query q and its corresponding KG $\mathcal{K}_q$, we retrieve the top n questions, i.e. $q_1, ...q_n$ and their respective KGs, $\mathcal{K}_{q_1}, ..., \mathcal{K}_{q_n}$.

(ii) **Encoding path information:** We now construct the answer paths for the retrieved KGs $\mathcal{K}_{q_i}$. An answer path p comprises a sequence of relations, starting from a source entity to the answer entity in the KG. There can be multiple answer paths between the source and the answer, but for simplicity we consider only the shortest paths, similar to Srivastava et al. (2021). We represent an answer path, either explicitly as a sequence of relations or by pooling over its constituent relation embeddings. Once we have embeddings for the paths, $p_1, p_2, ..p_m$, we pool them to obtain the retrieved path embedding, v_P^q for q.

(iii) **Scoring generated paths:** For the given query q, we generate all possible paths of a certain length, arising from the source entities in $\mathcal{K}_q$. The length of the path is determined by the maximum length of the answer path encountered during retrieval. These generated paths are encoded similarly to the retrieved paths, and a similarity score is computed between a generated path embedding and the retrieved path embedding. A higher similarity implies that the generated path is more likely to lead to an answer. On the other hand, if we store the path information as a sequence of relations (obtained during retrieval), then nodes we reach by traversing the retrieved sequence are answers for q.

4.2 PATHRGCN

We now propose our parametric PATHRGCN model that can encode and fine-tune path embeddings for KGQA. Given a question q, the corresponding knowledge graph $\mathcal{K}_q$ and the source entities, $s_1, s_2, ..s_k$, PATHRGCN (Figure 3), encompass the following steps during training:

(i) **Initialization:** We encode q using a pretrained language model (PTLM), to obtain the corresponding representation, v_q. We use unsupervised graph-representation learning techniques (Grover and Leskovec, 2016; Perozzi et al., 2017), that leverage the structural and neighbourhood information of nodes in the sub-graph, to obtain the corresponding node embeddings: $v_{e_1}, v_{e_2}, ...v_{e_N}$ for the N nodes $e_1, e_2, ..e_N$ in $\mathcal{K}_q$. Unlike Wang et al. (2020a,b), we do not use pretrained word embeddings since certain user-provided names can be arbitrary.

(ii) **Information propagation using GNN:** We employ graph neural networks (GNN) to perform information propagation over $\mathcal{K}_q$. We modify $\mathcal{K}_q$ by adding the inverse-relations between nodes and self-loops. Similar to Wang et al. (2020b,a), we concatenate the node representation, say e_i with v_q to capture the question information. Additionally, we also concatenate a binary value of b_i of 1 or 0, corresponding to whether e_i is a source entity. The resultant representation, $h^0_{e_i} = [v_q, v_{e_i}, b_i]$, is then passed as input to the GNN convolution layer, and the node representations are updated. We perform such updates L times, where L denotes the number of GNN layers, resulting in the final representation of $h^L_{e_i}$. We use non-linear activation and add dropout for regularization between updates. We use the RGCN model (Schlichtkrull et al., 2018) to account for different types of relationships between nodes.

(iii) **Path embedding generation:** We construct all possible paths $p_1, p_2, ...p_m$ upto a fixed distance from the source entities, and generate their corresponding path embeddings. The embeddings for path p_j, v_{p_j} is obtained by pooling over the updated representations of the nodes that constitutes the path, p_j. We hypothesize that learning the path structure can provide intermediate supervision (Srivastava et al., 2021) and can help prune-out nodes that are unlikely to be reached from the source.

(iv) **Answer prediction:** We perform answer prediction both at the node and path level. We concatenate the updated representation for node, e_i, $h^L_{e_i}$, with the question-embedding v_q, and pass it through a linear layer with sigmoid activation to predict whether the node is an answer or not. We carry out the same procedure for the path representation, v_{p_j} to predict whether the path, p_j, ends in an answer or not. We use binary cross-entropy loss for answer prediction at both the node (Node Loss) and path level (Path Loss) to minimize these losses during training.

Inference: During inference, given a question q^* and its corresponding sub-graph $\mathcal{K}_{q^*}$, the learnt PATHRGCN models outputs (i) probability that the node $e_1, e_2, ...e_N$ is an answer and (ii) probability that the paths $p_1, p_2, ..p_m$ leads to an answer. Thus for a given entity, e_i, we compute the maximum probability amongst all paths that end in e_i. We compute the mean of this probability alongside the probability of e_i being an answer.

5 Experiments

5.1 Baselines

We choose GNN-based retrieval models as our baselines since they have achieved high performance across different KGQA datasets without additional information (query-answer paths or semantic parses). We experiment with three relevant KGQA retrieval techniques, namely, **Embed-KGQA** (Saxena et al., 2020), **Rel-GCN** (Wang et al., 2020a), and **GlobalGraph** (Wang et al., 2020b). We do not use baselines that require additional textual information to generate the heterogeneous graph, such as GraftNet (Sun et al., 2018) or PullNet (Sun et al., 2019) since this information is not available to us. We present a detailed description of the baselines in the Appendix B.1.

5.2 Experimental Details

PATHCBR: We experiment with the impact of masking entities on QA performance. For CloudKGQA, we identify entities by performing simple string-matching of text spans in a given question to their corresponding nodes in the KG, whereas for Mod-WebQSP, we use the publicly-available SpaCy NER[2]. We also experiment with SpaCy's POS-Tagger to mask proper nouns in the text. The masked query is encoded using the [CLS] token of RoBERTa-BASE (Liu et al., 2019). We experiment with different ways to encode relations, either as a one-hot vector or using RoBERTa-BASE to encode the text. We perform max-pooling over the constituent relation embedding to obtain the resultant path-embedding. Likewise, max-pooling over the resultant path-embeddings yields the retrieved path-embedding. We also experimented with mean-pooling, but max-pooling fared consistently better. The generated paths are similarly encoded during inference. We compute cosine-similarity between a generated and retrieved path embedding. For a given query, we retrieve the top 5 questions in descending order of their similarity.

PATHRGCN: For PATHRGCN, we use RoBERTa-BASE to encode the question text, and Walklet (Perozzi et al., 2017) to generate the unsupervised node-representations for the KG corresponding to the question. We use Walklet instead of Node2Vec since it exhibits the highest performance over several node classification tasks (Rozemberczki and Sarkar, 2020). Moreover, it does not require any additional features to generate the embeddings and is computationally fast; Walklet was ≈ 20 times faster than Node2Vec.

Baselines: We defer the reader to Appendix B.2 for the exact hyper-parameter settings and experimental details of the baselines.

5.3 Evaluation Metrics

We evaluate the performance of the baselines and our proposed approaches across two metrics commonly used in KGQA, namely, Hits@1 and Accuracy. For a given question, Hits@1 has a value of 100 if the highest-scoring candidate is a correct answer else it is 0. Accuracy denotes the fraction of answers predicted correctly amongst the top K candidates (as a percentage). We also measure Hits@K for a question, for which the value is 100

[2]`https://spacy.io/usage/spacy-101#annotations-ner`

	CloudKGQA			Mod-WebQSP		
Method	Hits@1	Hits@K	Accuracy	Hits@1	Hits@K	Accuracy
EmbedKGQA	31.6 ± 3.3	31.6 ± 3.3	31.6 ± 3.3	29.1 ± 1.9	32.6 ± 2.2	25.1 ± 1.8
Rel-GCN + TransE	44.9 ± 8.7	52.5 ± 6.1	41.4 ± 6.3	49.4 ± 2.3	59.6 ± 1.2	48.5 ± 1.8
GlobalGraph + TransE	46.6 ± 3.6	56.1 ± 1.9	43.6 ± 2.5	48.4 ± 0.6	59.1 ± 0.7	48.3 ± 0.9
PATHCBR (Ours)	**95.4 ± 0.3**	**96.7 ± 0.3**	**95.8 ± 0.5**	49.3 ± 0.1	56.0 ± 0.1	48.0 ± 0.1
PATHRGCN + Walklet (Ours)	<u>90.4 ± 2.1</u>	<u>91.3 ± 1.5</u>	<u>90.7 ± 1.5</u>	**68.6 ± 0.2**	**75.2 ± 0.4**	**68.5 ± 0.3**
PATHRGCN + Walklet - NL	90.3 ± 7.1	91.1 ± 6.9	90.6 ± 6.8	<u>65.7 ± 1.0</u>	<u>73.0 ± 1.1</u>	<u>65.8 ± 1.0</u>

Table 2: Performance of the baselines and our proposed approaches on CloudKGQA, and Mod-WebQSP. K denotes the number of correct answers. *NL* refers to Node Loss. We report the mean and standard deviation across 5 runs. The best performance is highlighted in bold and the second best is underlined.

	No Masking			Masking Entities			Masking Proper Nouns		
CloudKGQA	Hits@1	Hits@K	Acc	Hits@1	Hits@K	Acc	Hits@1	Hits@K	Acc
Path Sequence	67.9	67.9	67.9	67.9	67.9	67.9	66.4	66.4	66.4
One-Hot Vector	88.8	89.4	88.8	<u>95.4</u>	<u>96.7</u>	<u>95.8</u>	82.4	84.9	83.6
Text Embedding	83.6	86.1	84.8	**95.7**	**96.9**	**96.0**	78.4	80.9	79.5
Mod-WebQSP	Hits@1	Hits@K	Acc	Hits@1	Hits@K	Acc	Hits@1	Hits@K	Acc
Path Sequence	33.0	37.9	32.8	41.6	46.5	41.1	<u>47.4</u>	<u>52.2</u>	<u>46.2</u>
One-Hot Vector	32.5	41.1	32.3	44.6	52.1	43.7	**49.3**	**56.0**	**48.0**
Text Embedding	13.7	21.1	16.1	22.4	28.7	23.5	25.2	32.1	26.7

Table 3: Mean performance of PATHCBR across different settings for entity masking and encoding path information, as a sequence of relations (Path Sequence), as a One-Hot Vector, or as a Text Embedding using a PTLM. The best performance is highlighted in bold and the second best is underlined.

if the answer is present amongst the top K candidates, else it is 0. For both Accuracy and Hits@K, K is the number of correct answers. We carry out experiment for five random seeds and report the mean and standard deviation. We perform statistical significance using the paired bootstrapped test of Berg-Kirkpatrick et al. (2012), as stated by Dror et al. (2018).

6 Results

In this section we pose the following research questions (RQs) and attempt to answer the same. We present instances of preprocessed questions that the models sees during training.

RQ1. How well do our proposed approaches fare on PERKGQA as compared to other KGQA baselines?

We observe that both PATHCBR and PATHRGCN, yield the highest performance on CloudKGQA, outperforming the existing baselines by over 100% for Hits@1 and Accuracy in Table 2. We attribute the poor performance of prior KGQA techniques to their inability to (i) learn global node embeddings over the large base KG or (ii) update the embeddings during training.

For Mod-WebQSP, PATHRGCN achieves the highest performance outperforming preexisting baselines significantly (p-value ≤ 0.001). However, PATHCBR achieves performance comparable to the baselines, and can answer questions corresponding to templates encountered during training, for instance, *"who plays ken barlow in coronation"*. We attribute the low performance of PATHCBR to:

(i) The underlying global KG for Mod-WebQSP is more complex and dense. There are 572 possible relations as opposed to 8 for CloudKGQA . Moreover, there can be multiple relations between two entities, (e.g. *'location.country.capital'* and *'location.contained_by'* are both valid relations between Tokyo and Japan), a characteristic absent in CloudKGQA. The possible paths increases exponentially with hops, and additional supervision afforded by GNNs helps answer these questions with long-range dependencies (Wang et al., 2020b).

(ii) Not all possible relations encountered during inference were available during training. E.g., the most relevant question retrieved for *"what was wayne gretzky 's first team"* was *"what team does plaxico burress play for"*, because the relation corresponding to *"first team"* was absent during train-

	CloudKGQA			Mod-WebQSP		
Method	Hits@1	Hits@K	Accuracy	Hits@1	Hits@K	Accuracy
Rel-GCN + TransE	44.9 ± 8.7	52.5 ± 6.1	41.4 ± 6.3	49.4 ± 2.3	59.6 ± 1.2	48.5 ± 1.8
GlobalGraph + TransE	46.6 ± 3.6	56.1 ± 1.9	43.6 ± 2.5	48.4 ± 0.6	59.1 ± 0.7	48.3 ± 0.9
PATHRGCN + TransE	51.4 ± 4.8	68.4 ± 2.6	57.0 ± 4.4	53.1 ± 0.9	62.6 ± 0.7	52.0 ± 0.8
Rel-GCN + Walklet	79.1 ± 3.9	79.8 ± 4.2	79.3 ± 4.0	63.0 ± 1.1	71.3 ± 0.8	63.0 ± 1.2
GlobalGraph + Walklet	86.3 ± 3.8	87.2 ± 4.0	86.5 ± 3.9	64.4 ± 0.9	72.6 ± 0.9	64.6 ± 0.8
PATHRGCN + Walklet	**90.4 ± 2.1**	**91.3 ± 1.5**	**90.7 ± 1.5**	**68.6 ± 0.2**	**75.2 ± 0.4**	**68.5 ± 0.3**

Table 4: Performance of the baselines and PATHRGCN when initialized with different node embeddings. We report the mean and standard deviation across 5 runs. The best performance is highlighted in bold.

ing. At times the pretrained language model was unable to infer the semantic meaning of the query. E.g, the most relevant question for *"what town was martin luther king assassinated in"* was *"what town was abe lincoln born in"*, despite the occurrence of questions like *"where was huey newton killed"*. Thus if the templates are widely different, it is not sufficient to encode the question using a PTLM; rather we need to fine-tune the questions to learn meaningful representation.

We further inspect the capabilities of our techniques to address the individual characteristics of PERKGQA, namely multiple answers, variable hop distance, multiple constraints, and variable KG size. Our approaches outperform baselines consistently and significantly on all such fronts.

A thorough analysis of our proposed approaches on the different properties of these two datasets reveals their complementary strengths. We note PATHRGCN has a better performance on larger KG size, more answers, longer hops, and additional constraints, and vice-versa for PATHCBR. We defer the reader to Appendix C for these results.

RQ2. What is the impact of entity masking and encoding different path-information strategies on PATHCBR's performance?

We observe that masking entities using NER, or proper nouns using a POS Tagger improves performance in Table 3. The only exception is for CloudKGQA where due to arbitrary naming conventions (e.g. "abc123"), entities were not detected as proper nouns creating inconsistent templates. We observe that encoding relations as a one-hot vector yields better performance than a text embedding, especially when the relation-names exhibit high lexical overlap as in Mod-WebQSP. Moreover, representing the path information as a sequence of relations cannot deal with unseen templates as in CloudKGQA. We highlight instances that substantiate our claim in Appendix C.

RQ3. What role does graph structure and path-information play on PERKGQA?

We investigate the benefits of unsupervised graph representation learning techniques to initialize node embeddings. In particular, we compare the efficacy of Walklet and TransE embeddings, when applied to Rel-GCN, GlobalGraph, and PATHRGCN. We see significant improvements for all models when TransE embeddings are substituted with Walklet in Table 4.

Since we operate for individual KGs, TransE does not have sufficient information to generate meaningful node representations. Walklet leverages the neighbourhood information and thus can capture the structural representation for each KG. PATHRGCN significantly outperforms the baselines on both fronts, when all three models are initialized with Walklet or when all three models are initialized with TransE embeddings.

We also investigate the importance of incorporating node loss (NL in Table 2) for additional supervision. This aids Mod-WebQSP, where the presence of multiple relations between entities gives rise to several possible paths between source and answer, most of which are spurious. Since multiple paths do not exist for CloudKGQA, removing the node loss does not deteriorate performance.

7 Related Work

The KGQA task has evolved from a simple-classification setting (Mohammed et al., 2018) to an information retrieval paradigm (Wang et al., 2020b; Saxena et al., 2020; Yasunaga et al., 2021; Sun et al., 2019; Xiong et al., 2019) that can tackle multi-hop relations or complex questions. Other approaches include semantic parsing (Lan and Jiang, 2020; Ding et al., 2019; Maheshwari et al., 2019; Zhu et al., 2020; Ren et al., 2021) and reinforcement learning (Das et al., 2018; Lin et al., 2018; Saha et al., 2019; Ansari et al., 2019). We investigate graph-based information retrieval methods in this work since they achieve SOTA performance *without any additional information like logical forms or semantic parses.* This sets us apart from recent work on KGQA generalizability (Gu et al., 2021; Chen et al., 2021) which require such logical forms during training; information often unavailable for real-world data settings. Our work also differs from Sidiropoulos et al. (2020) which is more focused on entity-linking and relation prediction for unseen domains, and leverages existing web-resources, which is not applicable for us.

Most KGQA approaches that operate in an information retrieval setting over predefined (or base) knowledge graphs follow a similar procedure to make the problem computationally feasible. (Sun et al., 2018, 2019; Wang et al., 2020b,a). They first construct a smaller sub-graph for each question from the base graph, using the Personalized PageRank algorithm (Haveliwala, 2003). then re-use the base graph's node representation to initialize the nodes in the sub-graph. Thus during inference, they already have prior representation of the nodes. However, in our setting, we encounter new KG during inference, and thus we need to learn the representations of those unseen nodes from scratch.

Our PATHCBR approach is closely related to Das et al. (2020), which performs relation linking such as (Delhi, capital_of, _?_). They first retrieve entities similar to the query entity and the corresponding reasoning paths that lead to an answer for those retrieved entities. They then apply reasoning paths to the query entity. However PATHCBR differs in two ways; (i) We operate upon complex or compositional questions and retrieve similar *templates* rather than entities, (ii) We do not use a rule-based framework to generate reasoning paths. Rather, we encode the retrieved path information as an embedding and use it to score paths generated during inference, which ensures generalization. In a similar vein, Das et al. (2021) uses a neuro-symbolic case-based reasoning approach for answering complex, multi-hop questions over a KG. However, their approach cannot be applied to our setting since it requires logical forms (SPARQL queries) as an additional input. We circumvent this requirement by designing PATHRGCN that leverages GNNs, KGs' structure, and path information between source and answers.

8 Conclusion and Future Work

We propose PERKGQA, a realistic setting for performing question answering over knowledge graphs; for each user's question, we have their corresponding KG, but no additional information. Such a setting addresses the challenges of unseen nodes during inference, prevents access to information of other users, while being computationally feasible. However, state-of-the-art KGQA techniques that require learning node-representations a priori fare poorly. We propose two approaches, a simple non-parametric case-based-reasoning model and a supervised neural architecture, both of which harness path information for QA. Our approaches improve upon the baselines by 6.5% on an academic dataset and 10.5% on an internal dataset.

Having demonstrated, the applicability of PERKGQA in the cloud service provider domain, we aim to explore other scenarios involving personalized or sensitive information, like healthcare. Prior work in medical NLP has focused predominately on generic or ontological knowledge such as UMLS. A personalized KG, constructed over a patient's health records, will encode information specific for the individual and not the general population, e.g. whether the patient is allergic to certain medications. We plan to collaborate with medical professionals and create such personalized KG in the healthcare domain to assist patients.

Furthermore, we seek to address certain limitations of our current approach, namely their inability to tackle spurious paths. We plan to rectify it either by explicitly providing the correct path information or incorporating some learning paradigm to detect them (He et al., 2021). Moreover for PATHCBR, the retrieval phase is a bottleneck since one needs to compare a given query with all possible training questions, and requires better indexing schemes like FAISS (Johnson et al., 2019). Likewise the inference time for both approaches increases as the number and the length of the paths increases. However PATHRGCN is able to adapt to longer paths, since the node embeddings provide some degree of additional supervision. Nevertheless, we plan to explore techniques beyond embedding-based approaches, namely semantic parsing or query-graph generation to alleviate the path-based constraint, and adapt them to our PERKGQA. In absence of gold logical forms, we plan to learn semantic parses through a weakly-supervised or distantly-supervised setting similar to Cheng et al. (2019).

References

Ghulam Ahmed Ansari, Amrita Saha, Vishwajeet Kumar, Mohan Bhambhani, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. Neural program induction for kbqa without gold programs or query annotations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4890–4896. International Joint Conferences on Artificial Intelligence Organization.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. 2020. Libkge-a knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 165–174.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. Retrack: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 325–336.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Rajarshi Das, Ameya Godbole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2020. Non-parametric reasoning in knowledge bases. In *Automated Knowledge Base Construction*.

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.

Jiwei Ding, Wei Hu, Qixin Xu, and Yuzhong Qu. 2019. Leveraging frequent query substructures to generate formal queries for complex question answering. 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)*, pages 2614–2622, Hong Kong, China. Association for Computational Linguistics.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Taher H Haveliwala. 2003. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering*, 15(4):784–796.

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 *WSDM*.

Xiaofeng Huang, Jixin Zhang, Zisang Xu, Lu Ou, and Jianbin Tong. 2021. A knowledge graph based question answering method for medical domain. *PeerJ Computer Science*, 7:e667.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. 2021. Conversational question answering over knowledge graphs with transformer and graph attention networks. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 850–862, Online. Association for Computational Linguistics.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th*

Annual Meeting of the Association for Computational Linguistics, pages 969–974, Online. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253, Brussels, Belgium. Association for Computational Linguistics.

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*.

Niklas Lüdemann, Ageda Shiba, Nikolaos Thymianis, Nicolas Heist, Christopher Ludwig, and Heiko Paulheim. 2020. A knowledge graph for assessing agressive tax planning strategies. In *International Semantic Web Conference*, pages 395–410. Springer.

Gaurav Maheshwari, Priyansh Trivedi, Denis Lukovnikov, Nilesh Chakraborty, Asja Fischer, and Jens Lehmann. 2019. Learning to rank query graphs for complex question answering over knowledge graphs. In *International semantic web conference*, pages 487–504. Springer.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296, New Orleans, Louisiana. Association for Computational Linguistics.

Junwoo Park, Youngwoo Cho, Haneol Lee, Jaegul Choo, and Edward Choi. 2020. Knowledge graph-based question answering with electronic health records. *arXiv preprint arXiv:2010.09394*.

Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. 2017. Don't walk, skip! online learning of multi-scale network embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 258–265.

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*, pages 8959–8970. PMLR.

Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1325–1334.

Amrita Saha, Ghulam Ahmed Ansari, Abhishek Laddha, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. Complex program induction for querying knowledge bases in the absence of gold programs. *Transactions of the Association for Computational Linguistics*, 7:185–200.

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*, pages 4498–4507, Online. 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*, pages 593–607. Springer.

Georgios Sidiropoulos, Nikos Voskarides, and Evangelos Kanoulas. 2020. Knowledge graph simple question answering for unseen domains. In *Automated Knowledge Base Construction*.

Saurabh Srivastava, Mayur Patidar, Sudip Chowdhury, Puneet Agarwal, Indrajit Bhattacharya, and Gautam Shroff. 2021. Complex question answering on knowledge graphs using machine translation and multi-task learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3428–3439, Online. Association for Computational Linguistics.

Haitian Sun, Tania Bedrax-Weiss, and William 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)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William 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*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Xu Wang, Shuai Zhao, Bo Cheng, Jiale Han, Yingting Li, Hao Yang, and Guoshun Nan. 2020a. Hgman: multi-hop and multi-answer question answering based on heterogeneous knowledge graph (student abstract). In *Proceedings of the AAAI Conference* *on Artificial Intelligence*, volume 34, pages 13953–13954.

Xu Wang, Shuai Zhao, Jiale Han, Bo Cheng, Hao Yang, Jianchang Ao, and Zhenzi Li. 2020b. Modelling long-distance node relations for KBQA with global dynamic graph. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2572–2582, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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*, pages 4258–4264, Florence, Italy. Association for Computational Linguistics.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Yucheng Zhou, Xiubo Geng, Tao Shen, Wenqiang Zhang, and Daxin Jiang. 2021. Improving zero-shot cross-lingual transfer for multilingual question answering over knowledge graph. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5822–5834, Online. Association for Computational Linguistics.

Shuguang Zhu, X. Cheng, and Sen Su. 2020. Knowledge-based question answering by tree-to-sequence learning. *Neurocomputing*, 372:64–72.

Supplementary Material

A Dataset

A.1 Constructing Modified WebQSP (Mod-WebQSP)

We also mimic our setting on the publicly-available WebQSP dataset (Yih et al., 2016), which operates on the Freebase KG, $\mathcal{F}$, . We use the pruned version of the dataset provided by Saxena et al. (2020). To completely mimic our setting, we construct a graph $\mathcal{F}_q$ associated with each question q with the caveat that a significant fraction of nodes we encounter during inference are not observed during training.

Each question is associated with a source entity as noted in the dataset of Saxena et al. (2020). The question's corresponding KG, comprises all nodes, a distance of k-hop from the source entity, where k is the shortest distance between the source and the answer. We limit ourselves to k=2 similar to Saxena et al. (2020). Furthermore, to constrain the size of $\mathcal{F}_q$, we randomly sample 1000 paths at a k-hop distance from the source entity; these are inclusive of all paths that lead from the source to an answer.

We observe a small fraction of questions ($\approx 5\%$), which have $\geq$ 100 answers; these correspond to simple 1-hop questions like *"What did Roald Dahl write?"*, or *"Who are famous people from Spain?"*. We remove such questions to constrain the size of the KG. Since, our objective was to retrieve all possible answers in the KG for a given question, there are no missing answers in the KG corresponding to the question in the Mod-WebQSP.

To achieve low overlap between nodes we encountered during training and inference, we modify them by assigning new identifiers. For example, a node, "m.0gzh" corresponding to "Abraham Lincoln", was modified to "**KG_i**_m.0gzh" and "**KG_j**_m.0gzh" for questions q_i and q_j in their respective KG $\mathcal{F}_{q_i}$ and $\mathcal{F}_{q_j}$. Although these nodes have the same underlying entity name in the original KG, $\mathcal{F}$, their node representations are different in these two questions. We rank all relations in $\mathcal{F}$, based on the decreasing order of frequency, and chose the top 39 relations that occur in 95% of all triplets in $\mathcal{F}$. We modify only those nodes which are associated with these 39 relations.

We added the graph-identifiers to the most frequent relations to ensure a small degree of overlap between the training and the test sets, similar to the CloudKGQA dataset, where certain entities were universal like names of regions (India, USA). This would facilitate prior KGQA techniques, like EmbedKGQA, that perform KGC on the individual KGs to share embeddings and perform better. However our proposed approaches, PATHCBR, and PATHRGCN remains agnostic to the degree of overlap. They do not keep track of any prior entities. Specifically PATHCBR masks these entities in the question, whereas PATHRGCN learns these entity representations from scratch for each KG.

B Experiments

In this section we present the baselines in detail and our experimental settings.

B.1 Baselines

EmbedKGQA: The EmbedKGQA model (Saxena et al., 2020) performs Knowledge Graph Completion (KGC) on an existing knowledge graph, to learn node representations. They use ComplEx (Trouillon et al., 2016) to generate node embeddings, to account for the anti-symmetric nature of the relations between nodes. Furthermore, they use RoBERTa (Liu et al., 2019) as the Pre-Trained Language Model (PTLM) to encode the question. They learn an objective function to select answers based on the similarity between question and node embeddings and further perform pruning based on the relation type to prevent over-generation of candidates. EmbedKGQA can perform arbitrary multi-hop reasoning, is not restricted to a specific neighbourhood, and can effectively handle problems of incomplete links/edges. To ensure EmbedKGQA can be applied in our setting, we carried out KGC on the KG associated with the question instead of the entire Freebase KG. This ensures that the entity representations are distinct for each individual KG.

Rel-GCN: The Rel-GCN approach of Wang et al. (2020a) first constructs a smaller sub-graph $\mathcal{K}_q$ for a given question, using PPR (Haveliwala, 2003) from the large base knowledge-graph, $\mathcal{K}$. They encode the question q using PTLM as v_q, and use TransE (Bordes et al., 2013) on $\mathcal{K}$ to obtain the node representations v_{e_i} for node e_i in $\mathcal{K}$. They concatenate the node embedding with the question-embedding e_q, and then perform RGCN on $\mathcal{K}_q$ to obtain their updated representa-

https://developers.google.com/freebase/data tions. These updated representations are used to score whether a given node is an answer or not. For PERKGQAsetting we perform TransE not on the original graph, $\mathcal{K}$, but on each sub-graph $\mathcal{K}_q$.

GlobalGraph: The GlobalGraph technique of Wang et al. (2020b) is similar in conception to Rel-GCN, having the same steps, (i) sub-graph construction, (ii) encoding representations of question and nodes, (iii) running RGCN to update the node representations. Moreover, to capture long-dependencies between nodes, the model leverages the set of incoming and outgoing relations to assign a global type for each node. They also identify nodes that are correlated with the question and construct a dynamic graph connecting such similar nodes. GCN over this dynamic graph yields updated representations for such nodes. Once again, for PERKGQA, we perform TransE on the individual KG associated with the question $\mathcal{K}_q$.

B.2 Experimental Details

We describe the hyper-parameters we employ for the parametric models, namely PATHRGCNand the baselines.

PATHRGCN: For PATHRGCN, we use RoBERTa-BASE to encode the question text, and Walklet (Perozzi et al., 2017) during initialization to generate the unsupervised node-representations for each KG. The embedding sizes for the question, nodes, and GNN layers was set to 768, 128, and 200, respectively. We fix L, the number of GNN layers to 1. For Path-RGCN, the length of an answer-path is chosen based on the maximum distance between a source entity and an answer entity encountered during training. This corresponds to a distance of 3 for CloudKGQA and a distance of 2 for Mod-WebQSP. We used Adam optimizer with a low learning rate of 2e-5, a decay of 5e-4, and patience of 30, and trained for 100 epochs. Each model took around 3 hours to complete on a p3.8x large EC2 instance.

Baselines: For Rel-GCN (Wang et al., 2020a), and GlobalGraph (Wang et al., 2020b), we use RoBERTa-BASE (Liu et al., 2019) to encode the question, and TransE embeddings to initialize the nodes (Bordes et al., 2013). We use the publicly available PyTorch-Geometric library (Fey and Lenssen, 2019) to implement RGCN (Schlichtkrull et al., 2018) for these two baselines. The embedding dimensions for our question, node, and GNN layers are 768, 128, and 200 respectively. The number of GNN layers, was set to 2 and 1 for Rel-GCN and GlobalGraph respectively, as specified in their papers. For EmbedKGQA, we use the publicly available code of Saxena et al. (2020) along with the default hyper-parameters for training. We use the publicly-available, LibKGE (Broscheit et al., 2020) library to generate Complex embeddings for each KG.

C Analysis

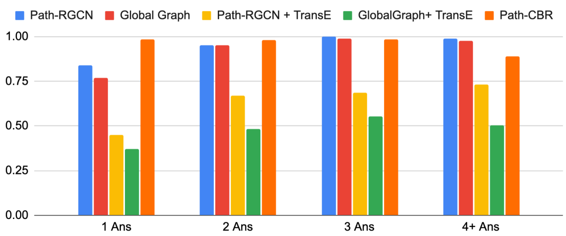
RQ1. What is the impact of entity masking and encoding different path-information strategies on PATHCBR's performance?

We investigate the impact of different strategies for masking entities and encoding path information on the performance of the PERKGQAtask for the two datasets and report them in Table 3.

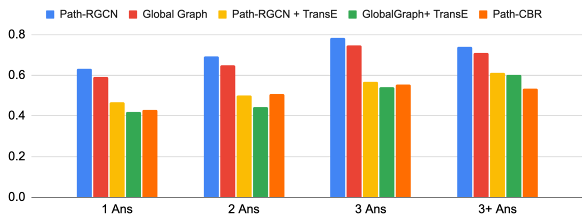
(i) **Entity-masking:** For Mod-WebQSP, entity masking using either a publicly-available NER or a POS Tagger, shows a huge boost in performance as seen in Table 3. Masking entities facilitates retrieving relevant questions which share similar answer types rather than similar entity names in the query. For example, for "What county is *greeley colorado* in ?", the most relevant question retrieved after masking is "What county is *novato california* in?", as opposed to "What college is in *greeley colorado*?". We observe a similar trend for CloudKGQA when we mask entities linked to nodes in the KG. However, the performance drops substantially when we use a POS-Tagger. Since the naming convention for nodes is arbitrary, like *"abc123"*, they are not detected as proper nouns; this creates inconsistent templates, and irrelevant questions appear higher in the ranked list.

(ii) **Encoding path information:** We observe that encoding relations as one-hot vectors fare just as well, if not better than encoding the relation-text using a PTLM. This is especially true for Mod-WebQSP where relation-names have high lexical overlap and thus exhibit high similarity. For example, for *"where is jamarcus russell from"*, the correct relation is **"people.person.place_of_birth"**, but the relation predicted, was **"people.person.date_of_birth"**. Encoding relations as one-hot-vectors circumvents this issue. Encoding the path-information, as a sequence of relations works well for Mod-WebQSP but not for our CloudKGQA, since the questions encountered during inference have different templates.

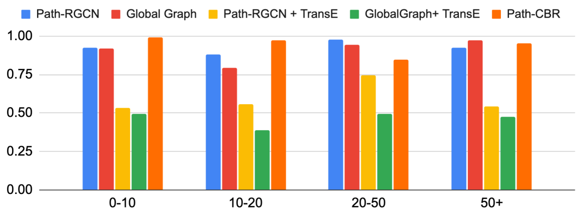
https://github.com/malllabiisc/EmbedKGQA



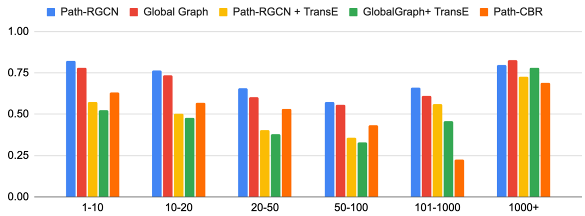
(a) CloudKGQA: Accuracy vs # Answers



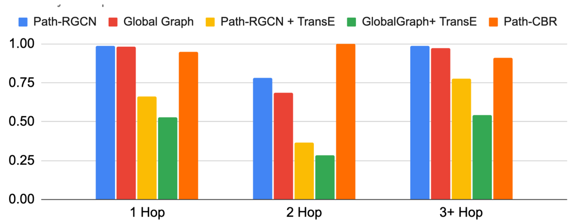
(b) Mod-WebQSP: Accuracy vs # Answers



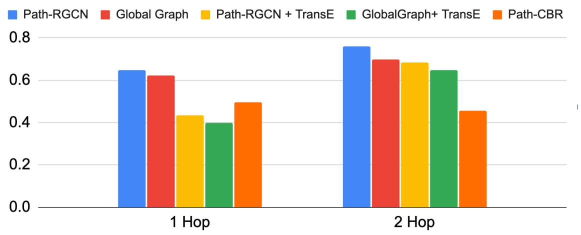
(c) CloudKGQA: Accuracy vs Subgraph size



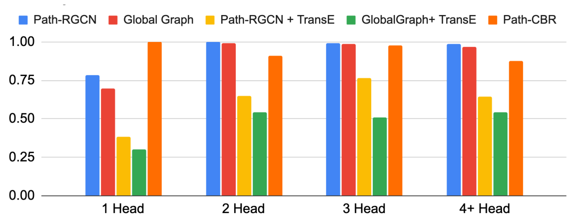
(d) Mod-WebQSP: Accuracy vs # Subgraph Size



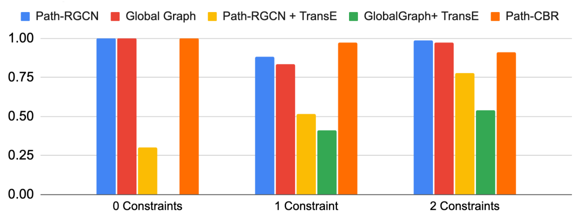
(e) CloudKGQA: Accuracy vs # Hops



(f) Mod-WebQSP : Accuracy vs # Hops



(g) CloudKGQA: Accuracy vs # source entities



(h) CloudKGQA: Accuracy vs # Constraints

Figure 4: Performance of the different techniques on the CloudKGQA dataset and the Mod-WebQSP dataset across different properties of the dataset. For CloudKGQA, we investigate the difference in performance based on the number of hops, head-nodes, logical constraints, and KG size. For Mod-WebQSP, we observe the difference in performance based on the number of hops and the size of the subgraph.

RQ2. How does our proposed approaches fare against the baselines for different KGQA properties?

We investigate the performance of the different methods (accuracy) on the PERKGQA task for different properties of the dataset. The methods we investigated were (i) PATHRGCN (ii) PATHCBR (iii) GlobalGraph initialized with Walklet (iv) PATHRGCN initialized with TransE, and (v) GlobalGraph initialized with TransE, the best baseline without any modifications. We investigate the following dataset properties.

(i) Variable number of answers: We observe the performance for variable number of answers, for CloudKGQA in Figure 4a and for Mod-WebQSP in Figure 4b.

(ii) Variable size of the graph: We note the effect of for varying graph size on different methods for CloudKGQA in Figure 4c and for Mod-WebQSP in Figure 4d.

(iii) Variable Hop Distance: We investigate the performance for varying number of hops for the CloudKGQA in Figure 4e and for Mod-WebQSP in Figure 4f.

(iv) Complex Questions: We observe specifically for CloudKGQA how the accuracy across methods varies for complex questions, based on the varying number of head-nodes in Figure 4g and the number of logical constraints in Figure 4h. This information was available to us for our internal dataset but not for Mod-WebQSP.

For CloudKGQA, we observe that our non-parametric PATHCBR approach achieves the highest performance when the number of answers is few (≤ 3), the subgraph is comparatively smaller (# edges ≤ 50), the number of hops is few (≤ 2), and when there are fewer constraints, (number of logical constraints ≤ 2, and number of source entities ≤ 3). PATHRGCN boasts a comparative higher performance for the converse scenarios, i.e., greater answers, larger size of the KG, more hops, and additional constraints. This observation highlights the trade-off between model complexity and the complexity of the question itself. The only exception lies for the 2-hop cases wherein PATHCBR achieves a score of 1.0 because the questions seen during training had a similar template, and answers were found within two hops. Nevertheless, across all sub-cases, we see that our proposed architectures, PATHRGCN or PATHCBR, boasts the highest performance, while the GlobalGraph + TransE, the best performing baseline, achieve the lowest performance. In fact, the baseline fares are consistently poorer than the PATHRGCN + TransE, which shows that incorporating the path information was beneficial across all stages.

For Mod-WebQSP, we see that our PATHRGCN model consistently boasts the highest accuracy across all sub-cases. The trend is similar to CloudKGQA, where the PATHRGCN model can handle larger KG size and more considerable hop distance. The only difference is the higher performance of PATHCBR when there are more answers, which is justifiable since the mean number of answers for Mod-WebQSP is five instead of two.

D Ethical Risks

In this paper, we propose PERKGQA, a realistic setting for performing question answering over knowledge graphs; for each user's question, we have their corresponding KG, but no additional information and we have to perform QA using that limited information. We acknowledge that our setting could applicable in scenarios involving personalized or sensitive information, like healthcare or insurance provider. Our settings is designed specifically to deter access to another user's information when a given user poses a query. However it is possible that the model might provide different answers to different users despite the same query, because the underlying KG is different for them. We acknowledge that our proposed idea is still in its infancy and requires more research to deem it suitable for real-world applications.