Research Article

Edge-Aware Graph Neural Network for Multi-Hop Path Reasoning over Knowledge Base

Yanan Zhang, Li Jin, Xiaoyu Li, and Honqi Wang

1 Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China
2 Key Laboratory of Network Information System Technology (NIST), Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China
3 University of Chinese Academy of Sciences, Beijing 100049, China
4 School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100190, China

Correspondence should be addressed to Li Jin; jinlimails@gmail.com

Received 21 October 2021; Accepted 23 August 2022; Published 12 October 2022

Academic Editor: Muhammad Tariq

Copyright © 2022 Yanan Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Multi-hop path reasoning over knowledge base aims at finding answer entities for an input question by walking along a path of triples from graph structure data, which is a crucial branch in the knowledge base question answering (KBQA) research field. Previous studies rely on deep neural networks to simulate the way humans solve multi-hop questions, which do not consider the latent relation information contained in connected edges, and lack of measuring the correlation between specific relations and the input question. To address these challenges, we propose an edge-aware graph neural network for multi-hop path reasoning task. First, a query node is directly added to the candidate subgraph retrieved from the knowledge base, which constructs what we term a query graph. This graph construction strategy makes it possible to enhance the information flow between the question and the nodes for the subsequent message passing steps. Second, question-related information contained in the relations is added to the entity node representations during graph updating; meanwhile, the relation representations are updated. Finally, the attention mechanism is used to weight the contribution from neighbor nodes so that only the information of neighbor nodes related to the query can be injected into new node representations. Experimental results on MetaQA and PathQuestion-Large (PQL) benchmarks demonstrate that the proposed model achieves higher Hit@1 and F1 scores than the baseline methods by a large margin. Moreover, ablation studies show that both the graph construction and the graph update algorithm contribute to performance improvement.

1. Introduction

Knowledge base question answering (KBQA) is a task to figure out the entities as answers for an input question from a given knowledge base (KB) and has attracted many researchers to work on it [1–10]. It is a challenging academic task, especially when answering multi-hop questions. As shown in Figure 1, a 3-hop complex question example is given. The topic entity of the question, “What languages are the movies that share directors with Dick Tracy in?”, is Dick Tracy, and the multi-hop triplet path \[(((\text{Dick Tracy}, \text{directed by}, \text{Warren Beatty}), (\text{Warren Beatty}, \text{directed}, \text{Reds}), (\text{Reds}, \text{in language}, \text{Russian}))\] needs to be extracted from the KB to find the answer Russian. Since the intermediate entity in multi-hop path reasoning is not unique, there may be multiple correct answers to the input question. For example, if the green node Heaven Can Wait in Figure 1 has the relation in_language with the entity English, the question will have two answers, English and Russian. Both precision and recall metrics are important for this type of multi-hop questions.

Early studies on multi-hop questions mainly focus on methods based on semantic parsing [11]. The intuitive idea is to convert a question into the corresponding structured query (e.g., in SPARQL) to extract the resulting entity from a KB. Although these methods prove effectiveness, their
processing steps and conversion process are relatively complex and meanwhile may involve expert knowledge or heuristic rules. Considering that answering multi-hop questions requires searching for reasoning paths, starting from the entity mentioned in question, and consisting of both the relations at each hop and intermediate entities, recent studies [12–14] have focused on the power of graph neural networks (GNNs) to solve the above limitations. They often model the question directly to get a candidate entity graph, and then leverage graph neural network-based information propagation methods to update the node representations in the graph, which are used to choose the answer entities. However, the current GNN-based methods and multi-hop path reasoning over knowledge base tasks are not compatible enough.

For answering a multi-hop question, people usually start from the topic entity mentioned in the question and search for the corresponding relation path in turn until the answer entity to the question is reached. Because GNN retains a state that can represent information from its neighborhood with arbitrary depth, the GNN-based information propagation process can simulate this kind of human problem-solving idea. In one hop message passing, the information of neighboring nodes is passed to the current node. After multiple propagation, nodes outside of multiple hops will obtain the information of the reasoning path starting from the topic entity. If this path and the question are highly matched, the possibility that this node refers to the answer is very high. Based on this assumption, we believe that the node representation after message passing contains the pivotal information needed to answer the question. Therefore, improvements are made on the basis of GNN network to adapt it to the multi-hop path reasoning over knowledge base task. First, the information contained in the predicate is essential, which determines the degree of matching with the question. However, previous GNN-based models [14–16] mostly consider how to propagate information between entities, while predicate information is more used as the weight of entity information in the propagation, ignoring their semantic information. To solve the above issues, the predicate information on the edges of the graph needs to be used reasonably and the predicate representation needs to be updated during message passing. Second, the way that the question is updated is also essential, which relates to the interactive information with the candidate entity graph. The previous methods [2, 12–14] either fix the question representation or update it with the representation of all nodes in the graph. To solve the issues,
further increasing the information interaction between the question and the entity nodes, improvements are made on the way to construct the graph. The question text is added to the entity graph as a node. Because the question contains the mention of the topic entity, the question node is directly connected to the topic entity node in the graph. In this way, during the graph update process, not only the information contained in the question is fused into the entity node representation but also the question node representation is also updated. To confirm the observations, we conducted experiments with our proposed model on two datasets, MetaQA [17] and PathQuestion-Large (PQL) [18], and the experimental results reached the state-of-the-art level. In particular, the F1 score has been significantly improved (the F1 score of our model is 10.7% and 28.6% higher than baselines on the MetaQA 2-Hop and 3-Hop, respectively), which shows that the proposed method improves Hit@1 without sacrificing the recall.

The main contributions of our paper can be summarized as follows: 1) We propose an edge-aware graph neural network to simultaneously update the representation of both the nodes and the predicate edges, better measuring the relevance of the question and relations for multi-hop path reasoning. 2) We construct a query graph, treat the question as a node, and connect it to the topic entity, which allows the question information to flow to candidate entities along the propagation path during updating the graph. 3) We conduct experiments on two widely used multi-hop KBQA datasets, MetaQA and PQL, to prove our theory and effectiveness of the proposed model.

2. Related Work

Multi-hop path reasoning over knowledge base aims at finding answer entities for an input question by walking along a path of triples from graph structure data, which is directly related to the existing KBQA research field. Besides, the method presented in our paper is similar to previous studies using GNN for question answering.

2.1. Knowledge Base Question Answering. Embedding based KBQA models are mainly divided into a single-hop task and multi-hop task. The single-hop KBQA models [19–23] predict the answer by judging the similarity between the question and relations in candidate triples. For example, Zhao et al. [20] proposed a joint scoring conventional neural network model that leverages subject-predicate dependency. Moreover, they used a novel well-order loss function to consider the different subject and predicate contributions to answer prediction. Zhou et al. [21] proposed a parameter-shared deep fused model that integrates subject detection and question-predicate matching in a unified framework. Wang et al. [23] used a multi-task learning framework to train a unified model, which shared a BERT encoder across all KBQA sub-tasks and define task-specific layers on top of the shared layer to reduce the cost.

Multi-hop KBQA tasks often adopt methods based on memory networks [24, 25], semantic parsing [26–28], or reinforcement learning [29–32]. For example, Xu et al. [25] improved traditional key-value memory networks to answer complex questions by designing a new query updating strategy to mask previously addressed memory information from the query representations, and they introduced a novel STOP strategy to read a flexible number of triples from memory slots. Maheshwari et al. [27] treated question answering as a problem of semantic query graph generation and re-ranking. They proposed a self-attention based slot matching model that exploits the inherent structure of the query graphs to learn how to rank core chain candidates. Hua et al. [31] took a meta-reinforcement learning approach to adapt the meta-learned programmer to new questions based on the most similar questions retrieved. To effectively create the support sets, they proposed an unsupervised retriever to find the questions that are structurally and semantically similar to the new questions from the training dataset. In addition, there are some other KBQA methods [33, 34]. For example, He et al. [34] proposed a novel teacher-student approach, in which the student network aimed to find the correct answer to the query, while the teacher network tried to learn intermediate supervision signals for improving the reasoning capacity of the student network. However, these methods lack of considering graph structure information contained in the KB. Recent studies [2, 13, 14] have introduced graph neural networks into multi-hop KBQA tasks, which is an efficient way to leverage graph structure information to represent complex relationships among entities. GNN-based methods will be introduced in detail in the following section.

2.2. Graph Neural Network Based Question Answering. Recent studies on multi-hop question answering attempt to build graphs based on entities and conduct reasoning over the constructed graph using graph neural networks [35–38], which are introduced to modify propagation limitation in long-distance relation. GNN-based question answering consists of many popular research directions, including reading comprehension, multiple-choice question answering, open domain question answering, and KBQA.

2.2.1. Multi-Hop Reading Comprehension. GNN-based multi-hop reading comprehension approaches aggregate scattered pieces of evidence across documents into a graph and then employ GNN-based message passing algorithms to perform multi-step reasoning. This work proposed by [39] is the first attempt to explore how GNN can work in the context of multi-document reading comprehension. They constructed an entity graph, where nodes are entity mentions from supporting documents and edges encode relations between different mentions (e.g., within- and cross-document coreference), and then adapted a graph convolutional network (GCN) to answer questions by updating node representations based on neighborhood features. The subsequent research work based on GNN can be roughly divided into two directions. One direction is to calculate the graph node representation [40–42]. For example, Cao et al. [40] applied bi-directional attention between an entity graph
and input query after GCN reasoning over the graph to formulate a query-aware node representation, which could derive the mutual information between the query and entities for final prediction. Tang et al. [41] proposed a Gated-RGCN to utilize the attention and question-aware gating mechanism to regulate the usefulness of information propagating across documents and add question information during reasoning. The other direction is the graph construction [43–46]. For example, Tu et al. [43] constructed a heterogeneous graph, which contained different types of nodes (not just entity nodes) representing different granularity levels of information. Fang et al. [44] constructed a hierarchical graph to connect clues from different sources.

2.2.2. Question Answering over Knowledge Base. The research direction, commonsense question answering [15, 16], also uses the external KB as one information source to answer questions. For example, Feng et al. [16] equipped pre-trained language models with a multi-hop graph relation network, which inherites the interpretability of path-based models and the scalability of GNNs. Yasunaga et al. [15] designed a joint graph and proposed the node relevance scoring function to estimate the importance of KB nodes related to the question context. However, commonsense question answering, also called multiple-choice question answering, only needs to choose one answer from a fixed number of candidate options, which is equivalent to a simplified version of the KBQA task where any entity in the KB may be the specified answer. Moreover, several work [12, 47, 48] studied question answering over the combination of a large-scale KB and entity-linked text task, called open domain question answering. For example, Sun et al. [12] proposed a novel heterogeneous update rule based on GCN to aggregate evidence across different information sources. And they proposed an integrated framework called PullNet in follow-up published work [48], which used an iterative process to construct a question-specific subgraph containing information relevant to the question. However, these models pay more attention to the problem of heterogeneous information fusion.

GNN-based KBQA approaches [2, 13, 14, 49] are most closely related to our method. The work proposed by [49] is the first attempt to apply GNNs to KBQA. They proposed to use the GNN to encode the graph structure of the semantic parse. Wang et al. [14] introduced a novel model based on GNNs to capture long-distance node information. Han et al. [2] proposed a directed hypergraph convolutional network to handle multi-hop KBQA task, which leverages hyperedges to connect more than two nodes more than pairwise connection. Moreover, they designed a dynamic relation strategy for dynamically updating relation states and paying attention to different relations at different hops during the reasoning process. Different from these models, our paper focuses on the message flow and the interaction between the query with the candidate graph. We not only construct a query graph to allow a two-way interaction of question information and candidate entity information but also design an edge-aware message passing algorithm for fusing relation information into the entity representation to facilitate the final matching of questions and candidates.

3. Task definition

Let \( \mathcal{K} = (\mathcal{V}, \mathcal{E}, \mathcal{R}) \) denote a knowledge base, where \( \mathcal{V} \) is the set of entities, \( \mathcal{R} \) is the set of relations, and \( \mathcal{E} \) is the set of triples in the KB. A triple \( t \in \mathcal{E} \) is denoted as \( t = (s, r, o) \), where \( s, o \in \mathcal{V} \) are entities and \( r \in \mathcal{R} \) is the relation between head entity \( s \) and tail entity \( o \). Given a multi-hop path question \( Q = (w_1, w_2, \ldots, w_n) \), where \( w_j \) denotes the \( j \)-th word and \( n \) is the length of the question word sequence. The question contains only one topic entity \( s_1 \in \mathcal{V} \), which can be annotated by some existing entity linkers, and its answer can be found by walking down a triplet path \( \{t_1, t_2, \ldots, t_L\} \subseteq \mathcal{E} \), where \( t_l \) denotes the \( l \)-hop triple answering the question \( Q \). The first hop triple \( t_1 \) starts from the topic entity of \( Q \), and the last hop triple \( t_L \) ends with the answer entity. Note that \( o_{l-1} \) in \((l-1)\)-hop triple and \( s_l \) in \( l \)-hop triple are the same entity. The task is to find a triple path from \( \mathcal{E} \) and extract its end entity as the reasoning answer.

4. Method

As shown in Figure 2, to reason over a given query context using specific knowledge from a KB, the edge-aware GNN model consists of four main components. First, retrieve the subgraph \( G_c \) corresponding to the given question from the KB, and construct the query graph \( G_q \) by connecting the query node to the topic entities in \( G_c \). Second, encode the query graph \( G_q \) by using GNN-based information propagation for multiple rounds. Finally, predict the answer based on the relevance scores of both the final query node and candidate entity node representations. The details of each phase are described in the following sections.

4.1. Graph Construction. Given a question \( Q \), the topic entity \( s_t \) can be identified by any entity linker. Then, a subgraph \( G_c \) can be retrieved from the KB by querying \( \mathcal{N} \)-hop entities around the topic entity \( s_t \), which contains answer entities. All entity nodes in the subgraph are candidate entities \( \{C_q\} \). Traditional GNN-based methods directly perform message propagation on the retrieved subgraph \( G_c \). To strengthen information flowing between the question and entity nodes in the following information propagation, a new query node \( q \) representing the question context is introduced into the above subgraph, where \( q \) is connected to the topic entity \( s_t \) using a new predicate type \( r_q \). Note that the predicate type represents the relationship between the query context and the relevant entities in \( G_c \). Such a newly obtained graph containing the query and candidate nodes is termed as query graph \( G_q \).

4.2. Graph Encoder. The graph encoder layer initializes all graph nodes representing entities, and graph edges representing predicates to vector representation. The node
embedding for $q$ is initialized by using a long short-term memory network (LSTM) to encode the query context $\mathbf{e}_q = \text{LSTM}(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$, where $\mathbf{e}_q \in \mathbb{R}^d$ is the last state of LSTM output and $d$ is the hidden state size. Specifically, a LSTM has several cell layers to make memories, and each cell layer involves the forget, input, and output gates. Let $f_j, i_j,$ and $o_j$ denote the $j$th cell layer outputs of forget, input, and output gates. The following formula elaborates the technical details of the $j$th cell layer:

$$
\begin{align*}
    f_j &= \sigma(W_f w_j + V_f s_{j-1} + b_f), \\
    i_j &= \sigma(W_i w_j + V_i s_{j-1} + b_i), \\
    o_j &= \sigma(W_o w_j + V_o s_{j-1} + b_o), \\
    c_j &= f_j \otimes c_{j-1} + i_j \otimes \tanh(W_c w_j + V_c h_{j-1} + b_c), \\
    s_j &= o_j \otimes \tanh(c_j),
\end{align*}
$$

(1)

where $c_j$ is the cell state for long-term memory, $s_j$ is the intermediate state for short-term memory, $W$ and $V$ with different subscripts are the weight matrices, and $b$ with different subscripts is bias vectors. In addition, $\sigma()$ is a Sigmoid function, $\tanh()$ is a Tanh function, and $\otimes$ denotes the element-wise multiplication. The value of intermediate state of the last cell layer, $s_n$, is the query node embedding $\mathbf{e}_q$.

Other nodes and edges on $G_q$ are initialized by using pre-trained word vectors or random initialized vectors. Let $e_m$ represent the entity vector for entity node $m$ in $G_o$, and $x_r$ represent the predicate vector for predicate edge $r$ in $G_q$. The nodes and edges in the graph are stored in the entity matrix $E = \{e_1, \ldots, e_n\}$ and fact matrix $R = \{x_1, \ldots, x_n\}$, respectively, where $E \in \mathbb{R}^{n_e \times d}, R \in \mathbb{R}^{n_f \times d}$, $n_e$ is the number of entity nodes in the graph $G_o$, $n_f$ is the number of triples in the graph $G_q$, and $d$ is the embedding size that is equal to the hidden state size of the LSTM.

4.3. Multi-Hop Graph Update. The basic recipe for graph-propagation based models is to update node representations via iterative message passing between neighbors on the graph. This phase is called message passing (namely,
information propagation) that runs for \( L \) time steps (namely, \( L \) hops). The general formulas of node update are defined as follows:

\[
h_m^{(l)} = \chi \left( h_m^{(l-1)} \sum_{r \in N_r(m)} \psi \left( h_k^{(l-1)}, x_r \right) \right),
\]

where \( l \in \{1, \ldots, L\} \) is the hop number, \( N_r(m) \) denotes all the entity neighbors of the current node \( m \) along the incoming edges of relation \( r \). \( \chi \) is an updating function, and \( \psi \) represents a message function. In (2), the updating function and message function can be any reasonable model or algorithm, which can be designed according to different targets. These two function designed in our model will be described in detail in the following section.

During the graph updating process, our edge-aware GNN model has conducted three strategies to enhance the path reasoning performance. First, every time node representations are updated, only if the information of neighbor nodes related to the query is calculated. Second, in order to get the node representations matching the input query, the edge information needs to be incorporated once the node is updated. Third, since relation information and query information are equally important, in addition to entity node representations, both query node and edge representations also need to be updated once an update operation is triggered. In conclusion, every time the graph is updated, there are three aspects that need to be updated: the relation edges \( R \), the entity nodes \( E \), and the query node \( e_q \) in the graph \( G_c \).

The detailed process of \( l \)th message passing is described in Figure 3. In addition, the entire message passing process is described in algorithm 1.

4.3.1. Entity Nodes Update. To shorten the semantic gap between entities and the natural language question, we concatenate each node representation \( e_m \), node \( m \in G_c \), with the question node embedding \( e_q \), which is defined as \( h_q = [e_m; e_q] \). Every time an entity node representation is updated, some new information needs to be added on the basis of the original entity embedding. This information is aggregated from the entity neighbors related to the query of the current node. In addition, our model borrows the core idea of graph attention networks [35], learns the relative weights between two connected nodes through the attention mechanism, which makes the information added from different neighbors have different weights. The difference is that in order to better adapt to the multi-hop path reasoning task, using the similarity between the relation and the question to calculate weight, instead of using the information of two adjacent nodes. Thus, in the \( l \)th hop graph updating stage, the representation \( h_m^{(l)} \in \mathbb{R}^d \) of each node \( m \in G_c \) can be updated by

\[
h_m^{(l)} = \text{FFN} \left( h_m^{(l-1)}, \sum_{r \in N_r(m)} \alpha_{rk} \phi_r \left( h_k^{(l-1)} \right) \right) + h_m^{(l-1)},
\]

where \( \text{FFN}() \) represents a single-layer feed-forward network, \( \phi() \) denotes the relation \( r \) specific message transformation function, and \( \alpha_{rm} \) is an attention weight that contains messages from node \( k \) to \( m \) connected with relation \( r \).

Specifically, the attention weight \( \alpha_{rm} \) is the relevance probability of the query and predicate embeddings,

\[
\alpha_{rk} = \text{softmax}(\langle x_r^{(l-1)} \rangle^T h_q^{(l-1)}),
\]
where $h_q$ is the query node representation (described in equation (8) (Eq6)) and softmax() is the softmax normalization over all outgoing edges from node $k$. From both (3) and (4), it is observed that the current node’s updated information comes more from these nodes connecting to the edges that are more relevant to that query.

As $G_q$ is a multi-relational graph, the message passed from a source node to the target node should capture their relationship. Thus, the message transformation function $\phi(\cdot)$ calculates the information transferred from neighbor node $k$ to $m$, which contains information of the edge between two nodes by introducing the relation embedding $x_r$,

$$\phi_k(h^{[l-1]}_k) = M^{[l-1]}_k \text{FFN}(x^{[l-1]}_r, h^{[l-1]}_k),$$  \hspace{1cm} (5)

where $M^{[l-1]}_k$ is a directed propagation matrix inspired from [12]. Combining (3) and (5), we can see that the edge information is fused into entity representation. Specifically, the directed propagation matrix uses the relevance of the query and predicates to control information flow direction,

$$M^{(0)}_m = \begin{cases} 1, & \text{if } m = q, \\ 0, & \text{otherwise}, \end{cases}$$  \hspace{1cm} (6)

$$M^{(l)}_m = (1 - \lambda)M^{[l-1]}_m + \lambda \sum_{k \in N_r(m)} \sum_{l} \alpha_{mk}^{ml} M^{[l-1]}_k,$$  \hspace{1cm} (7)

where (6) means that propagation starts from the query node. It can be observed from both (6) and (7) that $M$ can be regard as a weight factor that controls information flow along the edge related to the query.

4.3.2. Query Node Update. The initial representation of the query node is denoted as $h_q^{(0)} = [e_q; c_q]$, similar to the other node initial representation in the graph $G_q$. In l-layers, considering that the query node is directly connected to the topic entity node, the query representation also adds messages from the topic entity after using (3) to update with other entity nodes simultaneously,

$$h_q^{(l)} = \text{FFN}(h^{(l)}_q, h_r^{(l)}) + h_q^{(l-1)},$$  \hspace{1cm} (8)

where $h_q$ is the topic entity representation.

4.3.3. Predicate Edges Update. To obtain question-aware relation representation, during the l-layer graph updating process, the predicate vector connecting node $m$ and $k$ is updated by

$$x_r^{(l)} = \text{FFN}(x_r^{[l-1]}, \alpha_{mk}^{ml} h^{(l)}_k + x_r^{[l-1]}),$$  \hspace{1cm} (9)

where $\alpha_{mk}^{ml}$, the similarity between the relation $r$ and question $Q$, has been calculated by using (4). Residual connections are used when updating each node and edge because it can stitch together features at different levels to increase feature diversity and speed up training.

4.4. Answer Prediction and Training. After $L$-hop information propagation, we have final query representation $h_q^{(L)}$ and entity representation $h_m^{(L)}$ for entity $m$. The probability of this entity being the answer is calculated by the relevance score of the query and the entity representations,

$$p_m = \sigma((h_q^{(L)})^T h_m^{(L)}).$$  \hspace{1cm} (10)

Locating the answers among the candidate entities in the query graph can be regarded as a node classification task, judging whether an entity node is the answer entity or not. Thus, the training process uses binary cross-entropy loss over above probabilities, which is defined as

$$L(\theta) = - \sum_{m=1}^{n_c} [y_m \log(p_m) + (1 - y_m) \log(1 - p_m)],$$  \hspace{1cm} (11)

where $\theta$ represents the model parameters and $y_m$ is the golden probability distribution over the entity.

At the testing stage, the entity with the highest score on the query graph is selected as the answer to calculate the Hit@1 metric. In addition, if the difference between the scores of other entities and the highest score does not exceed the threshold $0.1$, these entities are also used as the answer selected by the model to calculate the F1 score.

4.5. Computation Complexity. We analyze the time and space complexity of our method and compare with prior works, GRAFT-Net [12], MHGRN [16], and QA-GNN [15] in Table 1. As we handle edges of different relation types using different edge embeddings instead of designing independent graph networks for each relation as in MHGRN, the time complexity of our method is constant with respect to the number of relations and linear with respect to the number of nodes. In addition, our model achieves the same space complexity as other models.

5. Experiments

5.1. Datasets. We used two benchmark datasets to evaluate our proposed edge-aware GNN model: MetaQA and PQL. The statistics of these datasets are described in Table 2.

MetaQA is a large-scale multi-answer dataset for KBQA in the movie domain. It contains three versions of questions, namely, Vanilla, NTM, and Audio, and each version consists of 1-hop questions, 2-hop questions, and 3-hop questions. The data form is a question-answer pair, namely, each question is followed by a list of answer entities. The dataset also provides a background KB, which contains 40128 entities and 9 relations. To make a fair comparison with previous work, we use the Vanilla version and query the given KB to predict answer for three sets of different hops.

PQL is a single-answer multi-hop KBQA dataset, which is a more challenging version in PathQuestion. The dataset consists of 2-Hop (PQL-2H) questions and 3-Hop (PQL-3H) questions, which contains 1594 and 1031 data samples, respectively. The data form is a question labeled with the golden reasoning path starting from the topic entity to the answer entity. It also provides corresponding background KB, which contains 5035 entities and 364 relations. The
original dataset does not have a standard training set, test set, and dev set, we divide them at a ratio of 8:1:1 to make fair comparisons.

5.2. Implementation Details

5.2.1. Experimental Settings. We run the experiments on a V100 GPU with 16G memory. The batch size is set to 32. All the embeddings are initialized randomly. The hidden dimension of the LSTM is 300. The hidden dimension of all GNN layers is set to 300. The layer number is 4 for all GNNs in 2-hop settings and 5 in 3-hop settings. The dropout rate is set to 0.2. The Adam optimizer [50] is used with the learning rate of 0.001.

5.2.2. Data Pre-Processing. First, entity linking is performed to get the topic entity of a question. For entity linking, we use simple surface level matching. Then, query forming to get the topic entity of a question. For entity linking, we use simple surface level matching. Then, query

original dataset does not have a standard training set, test set, and dev set, we divide them at a ratio of 8:1:1 to make fair comparisons.

5.2. Implementation Details

5.2.1. Experimental Settings. We run the experiments on a V100 GPU with 16G memory. The batch size is set to 32. All the embeddings are initialized randomly. The hidden dimension of the LSTM is 300. The hidden dimension of all GNN layers is set to 300. The layer number is 4 for all GNNs in 2-hop settings and 5 in 3-hop settings. The dropout rate is set to 0.2. The Adam optimizer [50] is used with the learning rate of 0.001.

5.2.2. Data Pre-Processing. First, entity linking is performed to get the topic entity of a question. For entity linking, we use simple surface level matching. Then, query

the background KB to obtain entities and predicates within n hops for a n-hop question and obtain a question-related subgraph. For the PQL dataset, all entities and predicates within n hops for a n-hop question are kept to construct the subgraph. For the MetaQA dataset, the maximum number of fact triples retrieved for a question is very large. To fit into GPU memory for gradient-based learning, the size of the retrieved subgraph is limited. We randomly remove some non-answer entities and predicates from the obtained n-hop subgraph. The statistics of the final

Table 3: The statistics of the subgraph input to the model on PQL.

<table>
<thead>
<tr>
<th></th>
<th>PQL 2-hop</th>
<th>PQL 3-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>925</td>
<td>103</td>
</tr>
<tr>
<td>Test</td>
<td>103</td>
<td>103</td>
</tr>
<tr>
<td>Entities</td>
<td>5035</td>
<td>364</td>
</tr>
<tr>
<td>Relations</td>
<td>27.06</td>
<td>17.97</td>
</tr>
<tr>
<td>Max facts</td>
<td>416</td>
<td>204</td>
</tr>
<tr>
<td>Max entities</td>
<td>184</td>
<td>92</td>
</tr>
<tr>
<td>Avg. Entities</td>
<td>19.77</td>
<td>20.49</td>
</tr>
</tbody>
</table>

Algorithm 1: Multi-hop edge-aware message passing algorithm.

\[ \text{Input: } \text{Query graph } G_q = (V_q, E_q, R_q); \text{ hop number } L; \text{ initialized query embedding } \mathbf{e}_q; \text{ initialized entity embedding } \mathbf{e}_m \forall m \in V_q; \text{ initialized predicate embedding } \{\mathbf{x}_r \forall r \in R_q\}; \text{ neighborhood function } \mathcal{N}(\cdot); \text{ feed- forward network } FFN(\cdot) \]

\[ \text{Output: } \text{Vector representations } \mathbf{z}_m \text{ for all } m \in V_q \]

1. \( x_m^{(0)} \leftarrow x_m, \forall r \in R_q \)
2. \( h_m^{(0)} \leftarrow [\mathbf{e}_m; \mathbf{x}_m], \forall m \in V_q \)
3. for \( m \in V_q \) do
4. \( \text{If } m = q \text{ then} \)
5. \( M_m^{(0)} \leftarrow 1 \)
6. else
7. \( M_m^{(0)} \leftarrow 0 \)
8. end if
9. end for
10. for \( l \leftarrow 1 \text{ to } L \) do
11. for \( m \in V_q \) do
12. for \( k \in \mathcal{N}(m) \) do
13. \( a_{mk} = \text{softmax}( (x_{r}^{(l-1)})^T h_{k}^{(l-1)}), r; k \leftarrow m \)
14. \( \phi_{r/m}(h_{k}^{(l-1)}) = M_{k}^{(l-1)} FFN(x_{r}^{(l-1)}, h_{k}^{(l-1)}) \)
15. end for
16. \( h_{m}^{(l)} = FFN(h_{m}^{(l-1)} \sum_{k \in \mathcal{N}(m)} a_{mk} \phi_{r/m}(h_{k}^{(l-1)})) + h_{m}^{(l-1)} \)
17. \( M_{m}^{(l)} = (1 - \lambda) M_{m}^{(l-1)} + \lambda \sum_{r \in \mathcal{N}(m)} a_{rk} M_{k}^{(l-1)} \)
18. end for
19. for \( r \in R_q \) do
20. \( x_{r}^{(l)} = FFN(x_{r}^{(l-1)} a_{rk} h_{k}^{(l-1)}) + x_{r}^{(l-1)} \)
21. end for
22. \( h_{r}^{(l)} = FFN(h_{q}^{(l)}, h_{i}^{(l)} + h_{q}^{(l-1)} \backslash \text{entity} \rangle \)\( s_{1} \)
23. end for
24. \( \mathbf{z}_m \leftarrow h_{m}^{(l)}, \forall m \in V_q \)

Table 1: The computation complexity of different L-hop question answering models on a graph \( G = (V, E, R) \).

<table>
<thead>
<tr>
<th>Model</th>
<th>Time complexity</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-hop GRAFT-Net</td>
<td>( O(</td>
<td>V</td>
</tr>
<tr>
<td>L-hop MHGRN</td>
<td>( O(</td>
<td>V</td>
</tr>
<tr>
<td>L-hop QA-GNN</td>
<td>( O(</td>
<td>V</td>
</tr>
<tr>
<td>L-hop ours</td>
<td>( O(</td>
<td>V</td>
</tr>
</tbody>
</table>

Table 2: The statistics of the datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
<th>Entities</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Hop</td>
<td>96106</td>
<td>9992</td>
<td>9947</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MetaQA</td>
<td>118980</td>
<td>14872</td>
<td>14872</td>
<td>40128</td>
<td>9</td>
</tr>
<tr>
<td>3-Hop</td>
<td>114156</td>
<td>14274</td>
<td>14274</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQL 2-hop</td>
<td>1274</td>
<td>160</td>
<td>160</td>
<td>5035</td>
<td>364</td>
</tr>
<tr>
<td>PQL 3-hop</td>
<td>925</td>
<td>103</td>
<td>103</td>
<td>852</td>
<td>125</td>
</tr>
</tbody>
</table>

Table 3: The statistics of the subgraph input to the model on PQL.

<table>
<thead>
<tr>
<th></th>
<th>PQL 2-hop</th>
<th>PQL 3-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>925</td>
<td>103</td>
</tr>
<tr>
<td>Test</td>
<td>103</td>
<td>103</td>
</tr>
<tr>
<td>Entities</td>
<td>5035</td>
<td>364</td>
</tr>
<tr>
<td>Relations</td>
<td>27.06</td>
<td>17.97</td>
</tr>
<tr>
<td>Max facts</td>
<td>416</td>
<td>204</td>
</tr>
<tr>
<td>Max entities</td>
<td>184</td>
<td>92</td>
</tr>
<tr>
<td>Avg. Entities</td>
<td>19.77</td>
<td>20.49</td>
</tr>
</tbody>
</table>
subgraph input to the model are shown in Tables 3 and 4, respectively.

5.3. Baselines. We compare our model with the following baselines:

KV-MemNN: It was proposed by the authors of [51], which is an end-to-end memory network reasoning on the key-value structured memory storing KB facts. This model learns to use keys storing the subject and predicate to address relevant memories with respect to the question, whose corresponding values storing the object are subsequently returned.

IRN: It was proposed by the authors of [18], which is an interpretable hop-by-hop reasoning network. In each hop of reasoning, the model dynamically predicts a relation according to the corresponding part of the input question, and updates the state of both the reasoning process and the question representation utilizing the predicted relation.

VRN: It was proposed by the authors of [17], which is an end-to-end variational reasoning network to recognize the topic entity of the input question and learn multi-hop reasoning simultaneously in a unified probabilistic framework. In terms of logic reasoning task, the model uses a propagation-like deep learning architecture over the KB.

GRAFT-Net: It was proposed by the authors of [12], which is a graph convolution based neural network performing question answering over the combination of a KB and entity-linked text. The model uses a novel update rule to operate over heterogeneous graphs and a directed propagation method to constrain the reasoning starting from the topic entity.

SGReader: It was proposed by the authors of [47], which combines the unstructured text and KB triples to predict answers, where the SGReader employs graph attention networks to accumulate information for each entity in the question-related subgraph and the KAReader utilizes a gating mechanism to selectively incorporate the learned entity information into encoding the question and texts.

2HR-DR: It was proposed by the authors of [2], which is a directed hypergraph convolutional network-based model. The model learns the relation representation by connected entity features, allocates the weight dynamically for different relations, and then updates the entity representation based on dynamic relation weights.

GlobalGraph: It was proposed by the authors of [14], which is a GNN-based model capturing long-distance node relations by modeling the relation features of each node and further judging the feature similarity.

For KV-MemNN, GraftNet, and SGReader, the experiment results on MetaQA and PQL datasets are provided by [14].

5.4. Main Results and Analysis. Following the work proposed by Wang et al. [14], we employ the Hits@1 and F1 score to measure the performance of the models for the MetaQA dataset, and adopt Hits@1 for evaluating the PQL dataset.

Table 5 demonstrates the performance of the baseline methods and our model on the MetaQA dataset.

| Table 5: Experimental results on the MetaQA dataset. |
|-------------|-------------|-------------|-------------|-------------|
| Model       | MetaQA 1-hop Hits@1 | F1 | MetaQA 2-hop Hits@1 | F1 | MetaQA 3-hop Hits@1 | F1 |
| KV-MemNN    | 0.958 | — | 0.760 | — | 0.489 | — |
| VRN         | 0.975 | — | 0.899 | — | 0.625 | — |
| SGReader    | 0.967 | 0.960 | 0.807 | 0.798 | 0.610 | 0.580 |
| GRAFT-Net   | 0.974 | 0.910 | 0.948 | 0.727 | 0.778 | 0.561 |
| 2HR-DR      | 0.988 | 0.973 | 0.937 | 0.814 | — | — |
| GlobalGraph | 0.990 | 0.976 | 0.955 | 0.830 | 0.814 | 0.624 |
| Ours        | 0.972 | 0.985 | 0.968 | 0.937 | 0.963 | 0.910 |

The best results are indicated in bold values.

Table 6: Experimental results on the PQL dataset.

| Table 6: Experimental results on the PQL dataset. |
|-------------|-------------|-------------|
| Model       | PQL 2-hop Hits@1 | PQL 3-hop Hits@1 |
| KV-MemNN    | 0.622 | 0.674 |
| IRN         | 0.725 | 0.710 |
| SGReader    | 0.719 | 0.893 |
| GRAFT-Net   | 0.707 | 0.913 |
| 2HR-DR      | 0.755 | 0.921 |
| GlobalGraph | 0.760 | 0.941 |
| Ours        | 0.856 (± 0.050) | 0.931 (± 0.019) |

The best results are indicated in bold values.

Table 7: Ablation experiments of our model on the PQL dataset.

| Table 7: Ablation experiments of our model on the PQL dataset. |
|-------------|-------------|-------------|
| Model       | PQL 2-hop Hits@1 | PQL 3-hop Hits@1 |
| Ours        | 0.856 | 0.931 |
| w/o predicate edges update | 0.813 | 0.850 |
| w/o query node update | 0.825 | 0.890 |
| w/o interaction between query and predicates | 0.800 | 0.879 |
| w/o query node | 0.838 | 0.906 |

For KV-MemNN, GraftNet, and SGReader, the experiment results on MetaQA and PQL datasets are provided by [14].
outperforms all baselines on the MetaQA 2-Hop and MetaQA 3-Hop dataset, improving Hits@1/F1 by 1.3%/10.7% and 14.9%/28.6%, respectively. In addition, for MetaQA 1-Hop, we obtain competitive Hits@1 and improve F1 from 97.6% to 98.5%. This reason for the relatively low Hits@1 on MetaQA 1-Hop is that models like GlobalGraph use the PageRank algorithm to pre-prune some candidate entities, but our model does not use any prior knowledge in data pre-processing. In general, our model makes a great improvement on the F1 score, which means our model achieves a higher recall while ensuring precision. This is because relation information is added to the message passing process so that the model gives similar scores to the entity nodes that arrive on the same relation path, thereby identifying a series of answer entities and improving the recall rate. Besides, our model performs better on multi-hop questions than 1-hop questions because of the reasoning advantage of graph propagation in dealing with multi-hop questions.

As shown in Table 6, our method achieves the best Hits@1 compared with the baseline models on the PQL 2-Hop dataset, which remains a great improvement, 9.6% higher than the second best model. It also obtains a good result on PQL-3H, 1.0% higher than the third best one and 1.0% lower than the best one. Note that the original PQL dataset does not provide a standardized training, test, and dev set; therefore, the way that the dataset is divided greatly affects the experimental results. Because the data in this dataset has many duplicates, if the test set contains data that is similar to the train set, the experimental result will be very high. Thus, we have adopted five division methods to avoid similar data in the test set and the train set as much as possible. The experiments are repeated 5 times, and the average value and fluctuation range in the table were obtained. Combined with the fluctuation range, our experimental results are still quite competitive.
5.5. Performance Analysis

5.5.1. Ablation Study on Model Components. We conduct ablation experiments to evaluate the performance of different components in our model, and the experimental results illustrate the effectiveness of these components. Note that w/o predicate edges update does not consider updating the predicate information represented by the edges in the query graph, which only performs nodes update. w/o query node update does not consider updating query node representation. w/o interaction between query and predicates removes similarity between query and predicates as the propagation weight and performs information propagation based on the neighbors of nodes. The w/o query node does not consider constructing the query graph but uses the candidate entity graph. As shown in Table 7, we can find that our overall model achieves the best performance. Without these components, the performance of the model has declined, which demonstrates the effectiveness of the design of graph construction and update in our model. Specially, by comparing w/o interaction between query and predicates and w/o predicate edges update, the results illustrate the significance of relation information, which could guide the model to choose the best matching path with the question hop by hop. Table 8 shows some case studies to analyze our model’s behavior. Using our full model can better answer a major category of questions, that is, questions that contain loops, which means there are repeated triples in the reasoning path, such as the first and third examples. In addition, through these examples, it can be observed that the reasoning path related to the question cannot be obtained without predicate updating or interaction components, confirming the importance of relation information.

5.5.2. Impact of Number of Hops (L). We investigate the impact of hyperparameter L for the edge-aware GNN with experiments on MetaQA 1-Hop (Figure 4). The increase of L continues to bring benefits until L = 3. Performance begins to drop when L > 3, which might be attributed to the noise caused by longer relation paths in the knowledge base. However, deep hops will not greatly affect the experimental effectiveness. We believe that this is because the addition of the query node enables the question information to be better integrated into other entity nodes and relation edges, thereby reducing the introduction of irrelevant information in the process of deep message passing.

6. Conclusion and Future Work

Multi-hop path reasoning over knowledge base aims to find the answer entities in graphical data that contains rich relation information among entities. In this paper, we proposed an edge-aware GNN model to deal with this kind of graph. Our model first adopts a special graph construction way to enable smooth information interaction between the question and the candidates. Then, it updates the vector representation of each element in the graph by introducing question-related relation information. Finally, the edge-aware GNN model predicts the answers by calculating the correlation between the question and node entities. Experiments on the MetaQA and PQL benchmarks demonstrate that the proposed model achieves better Hit@1 and F1 scores than the state-of-the-art models by a large margin. Furthermore, both the constructed query graph and the graph update algorithm contribute to the performance improvement.

This work opens several interesting directions for future research. First, the proposed GNN-based model can be well applied to single or multi-hop questions. We can further explore the application of GNN in reasoning questions with constraints such as aggregation and comparison in the following work. In addition, this paper focuses on answering questions that contain only one topic entity. If there are multiple topic entities in the question or the topic entity is not clear, we will treat this fuzzy phenomenon as future work.

Data Availability

The data used to support the findings of this study are available at https://github.com/zmtkeke/IRN and https://goo.gl/f3AmcY.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (no. Y835120378).

References


[34] G. He, Y. Lan, J. Jiang, W. X. Zhao, and Ji-R. Wen, “Improving multi-hop knowledge base question answering by learning intermediate supervision signals,” in Proceedings of the WSDM ’21, the Fourteenth ACM International Conference on Web Search and Data Mining, Israel, March 2021.


