IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 May 2022, accepted 18 July 2022, date of publication 21 July 2022, date of current version 27 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192858

RESEARCH ARTICLE

Multihop Question Answering by Using Sequential Path Expansion With Backtracking

IYAD ALAGHA
Faculty of Information Technology, Islamic University of Gaza, Gaza 00972, Palestine
e-mail: ialagha@iugaza.edu.ps

ABSTRACT Multi-hop question answering from knowledge graphs has gained a growing attention in the past few years due to its vast applications in AI. Existing approaches in this regard mostly rely on exhaustively traversing all paths in the graph until reaching a candidate answer entity, leading to high time complexity and reasoning errors when training the model. Recently, a few works have tried to reduce the search space by proposing sequential-decision techniques so that only the path that is likely to lead to the answer is traced. However, the sequential decision on the relations is likely to accumulate errors due to the potential incorrect matching with the question. Alternatively, this work proposes a method that leverages sequence matching and a backtracking algorithm to identify the correct path to the answer while minimizing the accumulated error. The process starts from the question entity and grows the path iteratively by reasoning over the outgoing relations. The aim is to find the relation with the highest similarity to the question and to transit through it to the next entity. Meanwhile, a termination/backtracking check is performed at each iteration to validate the path and act accordingly either by proceeding to the next entity, stopping the process, or backtracking to correct a wrongly expanded path. The proposed method does not require setting a maximum number of hops in advance and uses a compare-aggregate model with attention mechanism for effective sequence matching. Experimental results show that our method significantly outperforms existing methods on three benchmarking datasets.

INDEX TERMS Multi-hop question answering, knowledge graph, backtracking, deep learning.

I. INTRODUCTION

Open-domain question answering (QA) is the process of finding answers to questions represented in natural languages. This topic has recently sparked a significant progress due to the exploitation of large-scale graph structured Knowledge Bases (KBs) such as Freebase [1] and DBpedia [2]. A KB-QA system often takes a natural language question as input, and refers to an underlying KB to find a relevant answer. KB-QA systems have applications in several AI domains, such as virtual assistants and chatbots.

A typical KB can be viewed as a knowledge graph (KG) consisting of nodes and edges, where nodes denote domain entities, and edges denote the relations between entities (see Fig. 1). Facts in the KB are typically represented by triples showing relations between pairs of entities. Given a natural language question, a KB-QA system can answer the question by searching the graph for the triple(s) that best match with the input question. This process is generally based on the following assumptions: First, it is assumed that the question contains a word that maps to a specific graph entity, which we call the question entity. Second, it is assumed that the answer to the question exists in the KB as another entity. The question and answer entities should be linked through one or more paths. Answering a question can be seen as a graph traversal that starts from the question entity and explores the neighboring paths until an answer entity is reached.

KB-QA falls into two categories [3]: single-relation QA and multi-relation QA. Single-relation QA focuses on simple questions that can be answered by finding one fact triple in the KB. For example, the question (''Who directed 'Texi Driver'?'') can be answered from the triple (''Texti Dirver'',

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

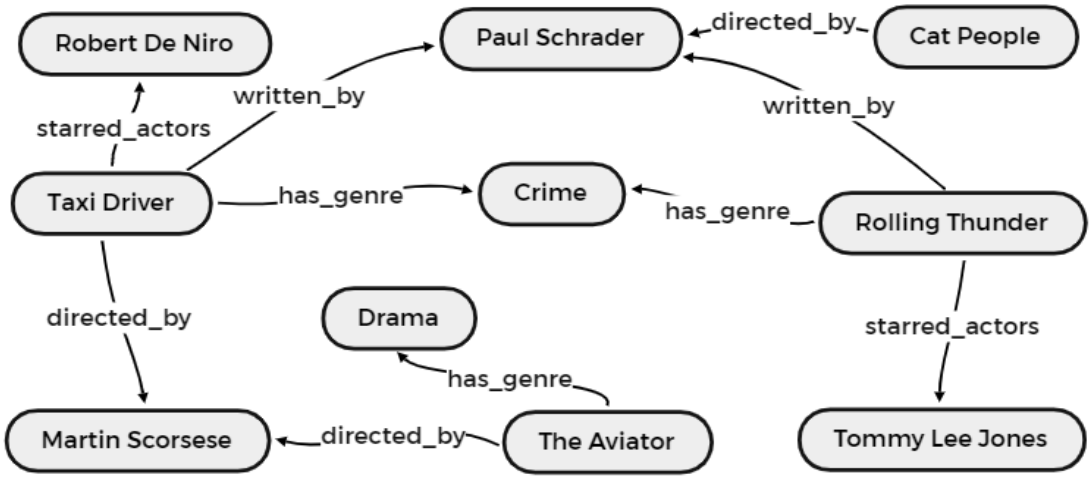

FIGURE 1. An excerpt of a knowledge graph.

''directed_by'',''Martin Scorsese'') from the graph in Fig. 1. Single-relation QA has been widely explored in several studies [4]–[7]. Multi-relation QA requires reasoning over multiple triples to answer the question. For example, the question (''Who starred in films for the screenwriter of Taxi Driver?'') can be answered through a path of 3 hops, i.e. (Texi Driver $\overset{\text{written\_by}}{\longrightarrow}$ Paul Schrader $\overset{\text{written\_by}}{\longleftarrow}$ Rolling Thunder $\overset{\text{starred\_actors}}{\longrightarrow}$ Tommy Lee Johns) whereas each arrow indicates a hop. In general, multi-relation questions are harder to answer because the answer entity lies away from the question entity, leading to a larger search space. Note that although the edges in Fig. 1 are directed, we assume that they can be traversed in both directions, and that inverse relations exit but are not shown for simplicity.

In recent years, a few works have proposed relation-chain-based methods for multi-relation QA over knowledge graphs [8]–[11]. These methods aim to extract a path from the question entity to the answer entity by iteratively taking steps forwards on the KB. They start from the question entity and traverse all outgoing paths over a predefined number of hops. Then, candidate answers are extracted and ranked based on their similarity to the question. The main problem with these methods is the large number of paths that need to be explored, and that number grows exponentially with the path length. This does not only increase the burden of searching the graph and training the model, but also makes it difficult for the model to rank the candidate answers because it has to learn how to distinguish the correct answer from too many wrong candidates [12].

To reduce the time complexity, some works have recently proposed sequential-decision based methods that attempt to reduce the search space by eliminating paths that most likely will not lead to an answer [8], [12], [13]. The selection of a particular path depends on the similarity of its constituents, i.e., relations and entities with the question [13]. The process starts from the question entity and explores its outbound relations to find that path that best matches the question. It then hops to the next entity through the selected relation. This process is repeated, and the path grows iteratively until a candidate answer is reached. Fig. 2 shows an example that applies this process to the graph in Fig. 1 to answer the question:''Who starred films for the screenwriter of Taxi Driver?''. Faded parts in Fig. 2 denote pruned paths.

Despite the potential of sequential decision-based methods for QA, we believe that the main problem lies in the possibility of pruning away paths that may lead to a correct answer. It can mistakenly make the decision to follow a wrong path only because its starting components match the question well, but the subsequent components do not. Take the following question as an example: ''The films that share directors with the film 'Taxi Driver' were in which genres?''. The only correct path in Fig. 1 that answers this question is: Taxi Driver $\overset{\text{directed\_by}}{\longrightarrow}$ Martin Scorses $\overset{\text{directed\_by}}{\longleftarrow}$ The Aviator $\overset{\text{has\_genre}}{\longrightarrow}$ Drama. However, a method may decide to take the path: Taxi Driver $\overset{\text{has\_genre}}{\longrightarrow}$ ...instead because the word ''genre'' is included in the question. This problem is likely to occur at the first hop when there are not enough features to make effective reasoning. As the decision to keep or skip a path is taken sequentially, the error is likely to accumulate as the number of hops increase. Experimental results from related works [8], [13], [14] have reported good performance with simple questions that contain one and two relations (1-hop and 2-hop questions), but the performance declines with complex questions that comprise three or more relations.

In this work, we propose a method for multi-hop QA that addresses the aforementioned limitations by focusing on two aspects: 1) reducing the time complexity by optimizing the search space and eliminating false candidate answers as possible: Instead of using an exhaustive search on the KG, our method attempts to predict a single path that leads to a candidate answer and expands iteratively. 2) resolving the problem of false judgement on relations at early stages. Our

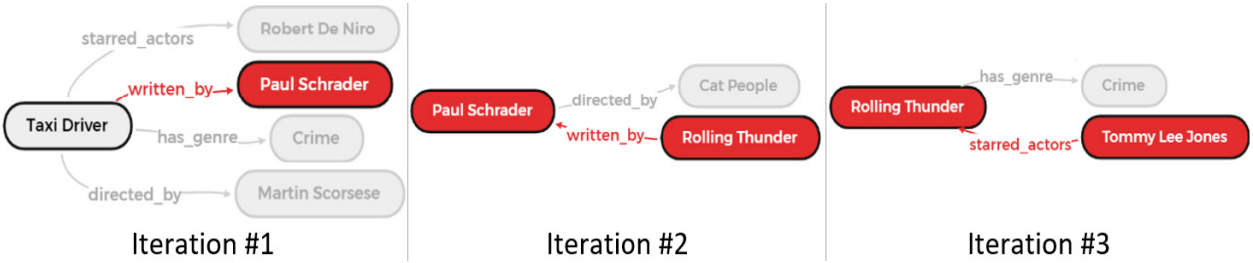

FIGURE 2. Sequential path expansion to answer the question: "who starred films for the screenwriter of Taxi Driver?" based on the KG in Fig 1.

main contribution is to propose an iterative path expansion approach based on backtracking in the KG to revert a false decision on relations. It tracks the changes in the quality of the path iteratively until its similarity to the question starts to decline. This indicates that the path may be falsely extended, and thus an action is taken to backtrack and try another route to seek an answer. In addition, our method proposes a termination check to determine when to stop iterations automatically.

The following section reviews related works and highlights the differences with existing works where possible.

II. RELATED WORKS

Methods for KB-QA are generally classified into two types: semantic parsing methods and embedding-based methods. Semantic parting methods aim to map natural language sentences to machine interpretable meaning representations such as executable queries [15], [16] or logical forms [17], [18]. The generated query can then be executed on the KB to retrieve the answer. Recently, many of these approaches have used neural networks to improve semantic parsing [19]–[23]. Semantic parsing methods mainly rely on hand-crafted rules or templates, and cannot effectively handle complex queries that do not map to the predefined logical forms. Embedding-based methods, under which this work falls, attempt to convert the entities/relations of the KB to low-dimensional vectors, so that the original structures and relations in the KG are preserved in these learned vectors [24], [25]. Subsequently, a given question can be answered by taking the embedding vector of the question as input to predict its corresponding entity/relation from the embedding space of the KB.

With the progress of deep learning, the use of neural networks to promote QA over knowledge graphs (QA-KG) has received tremendous attention. Several works [26]–[30] conducted the QA-KG task by first generating deep embedded representations of questions and KG constituents, that is, graph relations or entities. The KG constituents are then ranked on the basis on their semantic similarities with the embedding of the question. Finally, the top ranked constituent is chosen as a correct answer. In general, existing deep-learning based approaches for QA-KG differ in three aspects: 1) the deep learning model, 2) the method for extracting candidate answers from the graph, and 3) the method for ranking candidate answers. With respect to the deep learning model, a variety of models have been used to generate embedded representations of questions and candidate answers using convolutional neural networks [3], [31], LSTM networks [8], [13], [32], GRU networks [28], [33], [34], or transformers [35], [36]. Some studies [37], [38] showed that attention-based models such as BiLSTM with attention and transformers outperform other models and reduce training time.

When it comes to searching the KG for candidate answers, there are two lines of work: The first line comprises exhaustive search methods that systematically enumerate all paths to retrieve all candidate answers, and then check whether each candidate satisfies the question statement [3], [8], [39], [40]. However, processing through all relations is not practical for large-scale graphs, as explained earlier. One way to reduce the exhaustive searches is to limit the search space by setting the maximum number of hops in advance. However, this decision become unrealistic with complicated questions whose answer entities lie beyond the predefined number of hops. Das *et al.* [41] used case-based reasoning approach that retrieves similar k-nearest neighbor queries from the training set. Then, the subgraph neighborhoods containing entities present in those queries are extracted and searched for the answer entities. However, reasoning over multiple subgraphs increases the complexity of the search process and the model training. Niu *et al.* [42] proposed a model that incorporates path information with KG embeddings for multi-relational KBQA. The model uses a path representation mechanism to evaluate the ambipolar correlation between a path embedding and a multi-relational question embedding. However, pre-training the KG embeddings and then training the KB-QA model is computationally experience.

The other line of work includes methods that iteratively grow candidate paths and skip those paths that are unlikely to lead to correct answers. For example, Chen *et al.* [12] proposed an approach that extracts relations connected to the question entity and then predicts the most relevant relation to transit to the next entity. This process is repeated until a candidate answer is reached. They also proposed a termination mechanism to avoid setting the maximum number of hops. Similarly, Lan *et al.* [13] trained their model to iteratively grow the path that is semantically most similar to the question. They also proposed an approach to avoid

revisiting ranked paths when measuring the similarity to the question after each iteration. Shi *et al.* [43] proposed a generic approach that can handle QA from background knowledge represented in both graph and text forms. Their approach infers the answer by transferring entity scores along relation scores of multiple steps. He *et al.* [44] highlighted the problem of spurious relation paths, which are paths that lead to the correct answers but through incorrect relations. They used two different models to tackle this issue: one to find the correct answer, while the other tries to learn intermediate supervision signals to improve the reasoning of the first model.

Recently, reinforcement learning (RL) has shown a great potential to model reasoning systems, and thus several works formulated multi-hop KBQA as a RL task [45]–[48]. The idea is to set up a policy network to teach agents how to sequentially extend its inference path until it reaches a target entity. However, RL-based methods do not often improve performance because they lack effective supervision signals at intermediate steps and ignore semantic features of KG reasoning such as the semantic meaning of entities and relations [44]. Some works [14], [48] tried to overcome this limitation by exploiting embedding-based models to capture semantic features and use them to estimate the reward for target entities. Liao *et al.* [49] proposed a RL-based reasoner with two components: a stop signal that forces the model to stop after finding the answer and prevents it from hopping further, and a worth-trying signal that uses experiences from failed reasoning paths to amplify rewards.

A common problem of the aforementioned works has been clarified in the Introduction, which is the accumulated error in the process of progressive reasoning over relations. The method proposed in this work is also classified as an iterative path expansion method. It formulates the multi-relation QA as a sequential decision problem where candidate relations are ranked at each hop. It extends the former efforts by proposing a mechanism to iteratively validate the path and correct it if it has been wrongly extended by using a backtracking algorithm. In addition, our method proposes a termination check to determine when to stop iterations. This eliminates the need to set the number of hops to be known in advance, as in many works [3], [8], [29].

When candidate answers are selected, the next step is to rank them, and the top ranked result is chosen as the correct answer. In KB-QA, the ranking process is performed either at the final stage after selecting all candidate answers, or iteratively after each hop to select the next path to follow. In both cases, ranking is carried out by matching the question with candidate answers or paths. Several works on multi-relation KB-QA [8], [40] transform the question and the candidate answer to embedding vectors, and then the similarity between the two vectors is calculated by using one of the commonly-used similarity measures such as the dot product, cosine and Euclidean distance. However, it has been found that using a single vector to encode an entire sequence is not sufficient to capture all important information from the sequence [50]. Yu *et al.* [39] proposed a HR-BiLSTM model to match the question with candidate answers at the word level and token level. Xu *et al.* [51] used the slot filling method to transform the questions into structured information about the intents and used it to improve the accuracy of the ranking. Bordes *et al.* [26] took advantage of the subgraphs surrounding the candidate answers as features to include in the ranking process. Dong *et al.* [31] used multi-column convolutional neural networks to analyze additional aspects such as the answer type, path and context information, and used them to compute the score for the question-answer pair. Alternatively, this work uses the compare-aggregate model [52] with more effective comparison methods and the attention mechanism. The compare-aggregate model has been shown to generally outperform previous models and can better capture semantic similarity between textual sequences [52].

III. PROBLEM DEFINITION

Formally, a knowledge graph KG is represented as $G = (E, R)$, where E denotes the set of entities, and R denotes the set of relations between entities. Both E and R are associated with textual descriptions consisting of one or more words. KG is denoted as a set of triples, where a triple $(e, r, \grave{e}) \in G$ represents a fact in real life such as (The Wolf Man, has_genre, Horror).

A question entity $e_q \in E$ is a graph entity that corresponds to the main topic or keyword in a question q. It is assumed that the question entity is known in advanced and thus its detection is out of the scope of this work. The entities in the question can often be detected by using an entity linking tool such as S-mart [53].

An answer entity $e_a \in E$ is a graph entity that answers the question. e_q is supposed to be linked with e_a through a path consisting of one or more hops. For example, given the question and the graph shown in Fig. 2, the question entity is ''Taxi Driver''. The answer entity that we aim to find is ''Tommy Lee Johns'', which is 3 hops away from e_q. Note that multiple paths may originate from e_q, but the paths that lead to e_a are unknown in advance.

Give a question q in natural language, the goal is to search the KG for an answer entity e_a that is located away from e_q by one or more hops. For the multi-hop QA task, we aim to train a model by providing it with a set of (q, e_q, e_a) pairs. The model should learn how to efficiently find the answer path p_a that link e_q with e_a from a potential large number of candidate paths. In this work, p_a can be represented as the sequence $(e_q, r_1, e_1; r_2; e_2; \ldots; e_a)$, where [;] denotes the concatenation operation. It starts with the question entity e_q, ends with the answer entity e_a, and contains all the intermediate relations and entities in between. The combination of textual descriptions of the relations and entities along the answer path should correspond to what is expressed in the question.

Our method focuses on predicting the path that is likely to lead to the answer entity. Meanwhile, it aims to optimize this

task by 1) reducing the search space by eliminating irrelevant paths, 2) resolving the errors that occur due to false reasoning over relations, and 3) terminating the process automatically at the right time without having to set the maximum number of hops in advance.

IV. METHOD OVERVIEW

A KG-QA task can be seen as a graph traversal that starts from the question entity e_t and scores all outgoing routes by matching them to the question. Then the method transitions to the next entity through the route with the highest similarity score. The next iteration starts from the new entity, and repeats the same process to decide the best route to the move forward. Throughout iterations, we maintain the path that starts from the question entity e_q and extend it iteratively by concatenating the most related relation-entity pair to its end. That is, the path to the candidate answer is built incrementally by a sequence of candidate extension steps.

After each iteration, the method performs a check to decide whether to proceed to the next iteration, stop, or backtrack to choose an alternative extension route: The method proceeds as along as the path sequence becomes better aligned with the question as it is expanded, and should stop when the alignment maximizes and can no longer been improved. In the latter case, the method should terminate, and the answer is the last entity of the path, i.e. $e_a = tail(p)$, where p is the path to the answer entity. Otherwise, the method may find out that a transition in the path has been falsely made due to incorrect relation matching, and that it becomes no longer possible to reach an answer through the current path. In this case, the method should perform backtracking to consider another route for path expansion.

Our method follows an iterative approach whereas three steps are carried out in each iteration that are: 1) path extraction, 2) path ranking and pruning, and 3) termination/backtracking check. These steps are explained in what follows.

A. PATH EXTRACTION

To elaborate the path extraction step, we need first to define the following terms that are visually depicted in Fig. 3:

The candidate path $\boldsymbol{p}$: It is the set of entities and relations along the path from the question entity e_q to e_t, where e_t is the entity at iteration t. Let us represent p as the sequence $(e_q; r_1; e_1; r_2; e_2; \ldots; r_{t-1}; e_t)$, where $r_{t-1}, e_t \in G$. $p = (e_q)$ is the path at the beginning of the process before making any hop. p in our method grows incrementally, hop after hop, until eventually formulating the answer path p_a that has e_a in its tail.

The candidate extension relation set $\boldsymbol{R_t}$: it is the set of all outgoing relations of the current entity e_t. R_t is represented as the set $\{r_1, r_2, \ldots, r_k\}$, where k is the number of relations originating from e_t. At each iteration t, we aim to find the best relation $r_i \in R_t$ to extend p and move forward to next entity. The path after iteration t becomes $p = p \oplus (r_i; e)$, where $\oplus$ denotes the concatenation operator.

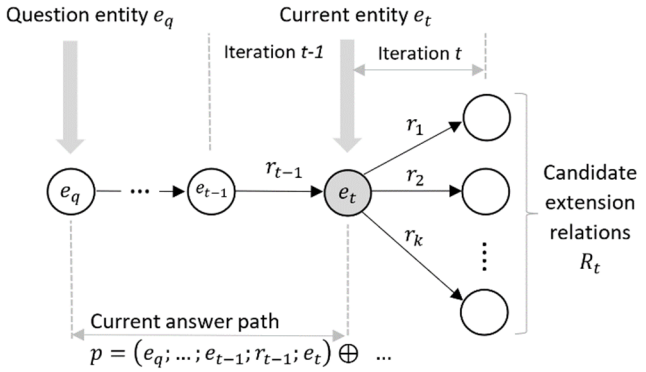

FIGURE 3. Symbols related to path extraction used in the proposed method.

A candidate path set $\boldsymbol{P_t}$: At iteration t, candidate relations in R_t should be scored and ranked based on their semantic similarity to the question, and top-ranked relation is used for path expansion while other relations are skipped. Sequence matching can be used to match candidate relations with the question. However, each relation often corresponds to a single word that is inappropriate to compare with the sequence of the whole question. Thus, each relation $r_i \in R_t$ along with its target entity are concatenated with a copy of p to create a set of candidate paths at iteration t, formally represented as the set $P_t = \{p_1, p_2, p_3, \ldots, p_k\}$, where $k = |R_t|$. That is, $p_i = p \oplus (r_i; e)$. Note that every $p_i \in P_t$ shares the same prefix that is p and only differs in the last relation-entity pair $(r_i; e_i)$. For example, assuming that "Taxi_Driver" in Fig. 1 is the question entity e_q, then the candidate path set in the first iteration is P1={("Taxi_Driver";"starred_actors";"Robert_ De _Niro"), ("Taxi_Driver";"written_by";"Paul_Schrader"), ("Taxi Driver";"directed_by";"Martin_Scorsese")}

B. PATH RANKING AND PRUNING

In each iteration, the set P_t is constructed, and each $p_i \in P_t$ is scored based on its semantic similarity to the question q. The top-scored p_i will become the candidate answer path p. The matching process is handled by the Question-Path Matching module (see Fig. 4) which consists of three layers: the encoder layer, the alignment layer, and the comparison layer.

The encoder layer represents q and each $p \in P_t$ as d-dimensional matrices. First, each word in the question and the candidate path is converted to embedding vectors by using GloVe. Then, a BiLSTM model with average pooling is used to create more composite embeddings for both q and p. Let $\grave{q} = (\grave{q}_1, \grave{q}_2, \ldots, \grave{q}_i, \ldots, \grave{q}_n)$ be the output of the BiLSTM structure representing the question's embedding, where $\grave{q}_i$ is the vector embedding of the i-th word in q. Let $\grave{p} = (\grave{w}_1, \grave{w}_2, \ldots, \grave{w}_j, \ldots, \grave{w}_m)$ be the output of the BiLSTM structure for p, where $\grave{w}_j$ is the vector embedding of the j-th word in p.

The alignment layer takes $\grave{q}$ and $\grave{p}$ from the encoder as input, and uses attentive BiLSTM structure to compute an aligned representation of the path sequence as output.

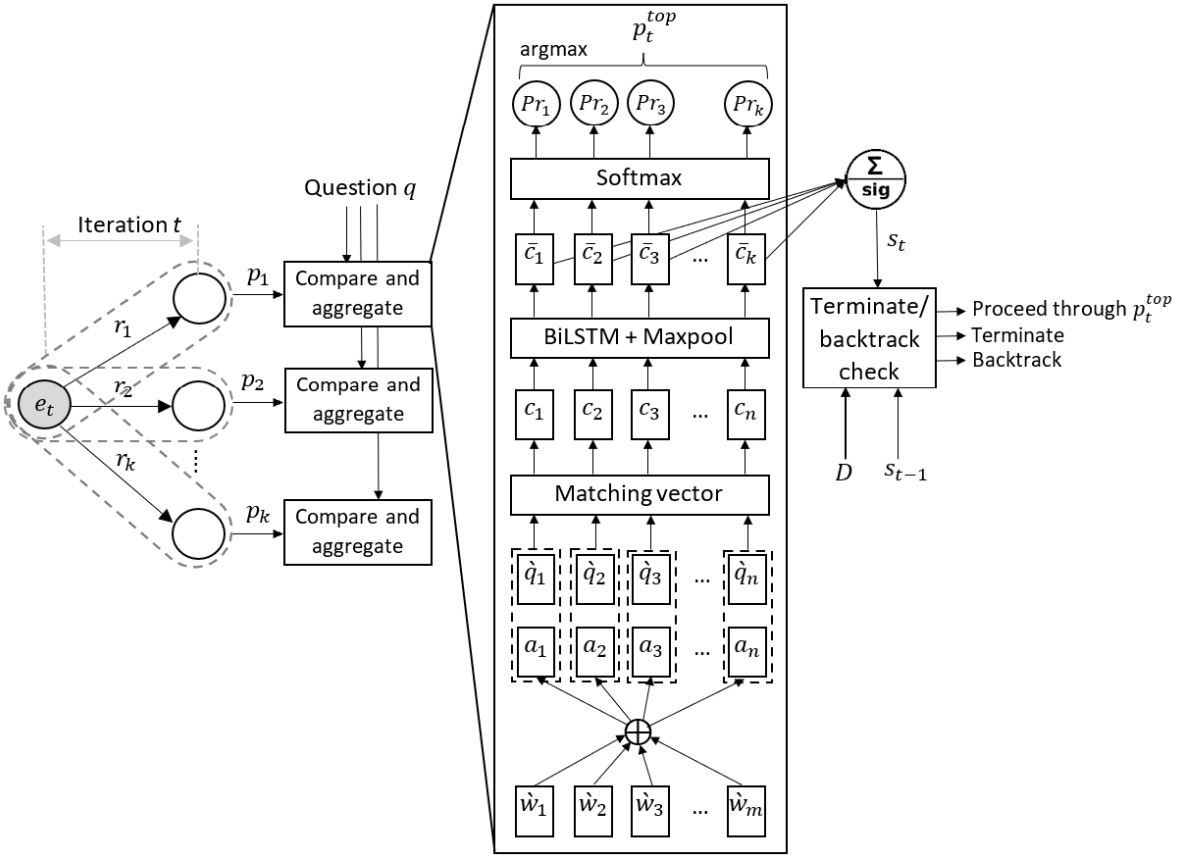

FIGURE 4. The sequential path expansion with backtracking method.

A question-to-answer attention mechanism is used to signify which words in the path sequence have the closest similarity to one of the query words and are hence critical for answering the question. The attention weights on the path words are calculated as follows:

$$u_{\grave{q}_i} = \tanh(W\grave{q}_i + b) \tag{1}$$

$$u_{\grave{w}_j} = \tanh(W\grave{w}_j + b) \tag{2}$$

$$\alpha_{i,j} = \frac{\exp(u_{\grave{q}_i}^T u_{\grave{w}_j})}{\sum_{j=1}^{n} \exp(u_{\grave{q}_i}^T u_{\grave{w}_j})} \tag{3}$$

$$a_i = \sum_{j=1}^{n} \alpha_{i,j} \cdot \grave{w}_j \tag{4}$$

where each $\grave{q}_i \in \grave{q}$ and $\grave{w}_j \in \grave{p}$ are fed into a layer with tanh as its activation function to get $u_{\grave{q}_i}$ and $u_{\grave{w}_j}$ respectively. W and b are parameters to be learned. $\alpha_{i,j}$ is the attention weight that demonstrates how well the i-th word in the question matches the j-th word in the candidate path. a_i is the attention-weighted sum that represents the part of the path that best matches the j-th word in the question. The expectation is that the words on the candidate path that are more important with regard to the input question should have larger weights.

The goal of the comparison layer is to match each $\grave{q}_i$, which represents the i-th word in the question with a_i, which represents a weighted version of the candidate path. Let $F\left(\grave{q}_i, a_i\right)$ be the comparison function that transforms $\grave{q}_i$ and a_i into a matching vector c_i that represents the comparison result. $F\left(\grave{q}_i, a_i\right)$ uses the comparison functions proposed by [54].

$$c_i = F\left(\grave{q}_i, a\right) = ReLU(W\begin{bmatrix}(\grave{q}_i - a_i) \odot (\grave{q}_i - a) \\ \grave{q}_i \odot a_i\end{bmatrix} + b) \tag{5}$$

where $\odot$ represents element-wise multiplication of two vectors. This function uses both vector subtraction and multiplication for sequence matching. Existing studies [52] show that this function performs better than common comparison functions such as the cosine and Euclidean distance. The function F (.) is a standard neural network layer that consists of a linear transformation followed *ReLU* as an activation function, where W and b are parameters to be learned.

After we apply the comparison function to each pair of $\grave{q}_i$ and a_i to obtain the sequence: $(c_1, c_2, \ldots, c_{|q|})$. Finally, we aggregate this sequence of vectors by using BiLSTM, followed by 1-max pooling to obtain a single vector:

$$\bar{c} = \max\_pool(BiLSTM\ (c_1, c_2, \ldots, c_n)) \tag{6}$$

where $\bar{c_t}$ encodes the complete matching vector between each path $p \in P_t$ and the question q at iteration t.

Assume that there are k sibling relations originating from the entity e_t as shown in Fig. 3. The set P_t will consist of k candidate paths that differ only in the last relation-entity pair. Let the sequence $(\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_k)$ be the matching vectors of paths in P_t, each calculated using Equation 6. The probability for us to choose $p_i \in P_t$ to be the answer path is computed using a SoftMax function:

$$Pr(p_i) = \frac{\exp(\bar{c}_i)}{\sum_{i=1}^{k} \exp(barc_i)} \tag{7}$$

Finally, Let p_t^{top} be the path with highest probability at iteration t. p_t^{top} is retrieved using the following:

$$p_t^{top} = \arg max_{1 \le i \le k} Pr(p_i) \tag{8}$$

C. TERMINATION AND BACKTRACKING

After selecting the top-scored path at iteration t, we need to take one of three actions: either to proceed, terminate, or backtrack. Intuitively, the proceed action means to transit from e_t to the next entity through the top-scored path, and repeat the process explained in the previous sections. The terminate action means that the method should stop either because it has reached a candidate answer when the current path p matches the question q well. Alternatively, the method may find out that p cannot be completed to an answer possibly due to incorrect matching in the preceding steps. This means that none of the outgoing relations of the current entity e_t can give rise to the matching score of p. In this case, p should be corrected thorough backtracking, which entails dismissing e_t, and stepping back to e_{t-1} to test the next top-ranked relation of e_{t-1}. The termination/backtracking actions are explained in what follows:

The method proceeds as long as the similarity between the path p and the question q increases iteratively. This indicates that p is becoming more similar to the question as it is expanded. When the similarity score maximizes and then starts to drop at iteration t, the method should stop or backtrack.

To determine when the similarity between p and q maximizes, the top matching score vector $\bar{c_t}$ of p_t^{top} from Equation 6 should be converted to a single value to ease our termination checking. $\bar{c_t}$ can be converted to a single value representing the final matching score by using the following equation:

$$s_t = \sigma(W^T \bar{c_t} + b) \tag{9}$$

where σ is a sigmoid function. W and b are parameters to be learned. s_t is the matching score of the path p at iteration t, where $s_t \in [0, 1]$. As the method progresses, it maintains s_t and s_{t-1}, which are the last two matching scores of the expanded path at the current and previous iterations respectively. When s_t becomes lower than s_{t-1}, this indicates one of two things: either the method has found the best matching path and cannot find a better one, or the path is incorrectly expanded from the previous iteration. The former condition demands a stop action, while the latter demands backtracking to try an alternative extension route.

To decide whether to stop or backtrack, we need to estimate the minimum distance from the question entity e_t and the answer entity e_a. This distance can be roughly estimated by counting the number of relations and entities in the question as proposed by [55]. A part of speed tagging is performed over the question text by using a POS tagger from HanLP.[1] Each possible entity or relation will be labeled as NN or NNP. Then the distance D between e_t and e_a is estimated as the following:

$$D = \begin{cases} 2 - hop & count\ (NN + NNP) \ge 2 \\ 3 - hop & count\ (NN + NNP) \ge 4 \\ 2 - hop/3 - hop & count\ (NN + NNP) = 3 \end{cases} \tag{10}$$

Let $\delta\,(s_t, s_{t-1}, D)$ be the check function that takes the matching scores of the path at iterations $t, t-1$ respectively and D to decide whether to proceed, terminate or backtrack. $\delta\,(s_t, s_{t-1}, D)$ is defined as the following:

$$\delta\,(s_t, s_{t-1}, D =) \times \begin{cases} proceed & s_t > s_{t-1} \\ terminate & (s_t < s_{t-1} \wedge t \ge D) \\ & \quad \vee (\forall r_i \in R_{t-1} : r_i\ is\ visited) \\ backtrack & s_t < s_{t-1} \wedge t < D \end{cases} \tag{11}$$

The estimated length D is used as the following: D is ignored as long as $s_t > s_{t-1}$. If s_t starts to decrease after exceeding D, the method will terminate concluding that the path respective to s_{t-1} is the best-matching path. If s_t drops before exceeding D hops, this means that the answer still lies ahead but the current path may no longer lead to it. The current path may have been expanded due to incorrect reasoning. Thus, the method will dismiss the current entity e_t and step back to the previous entity e_{t-1} to try an alternative sibling path. If all outgoing relations of e_{t-1} are visited and no result can result in an improved path score, then the method will terminate with a conclusion that the path respective to e_{t-1} is the best-matching path as it can never obtain a path with a better similarity score.

To illustrate in example how backtracking can improve the performance, we refer to the question discussed in the introduction, which is: ''the films that share directors with the film 'Taxi Driver' were in which genres?''. The distance D for this question is estimated as 3-hop based on Equation 10. Assume that the model decides in the first iteration to take the wrong path ''Taxi Driver $\xrightarrow{\text{has\_genre}}$ ...'' instead of the correct path ''Taxi Driver $\xrightarrow{\text{directed\_by}}$ ...'', and moves to the entity titled ''Crime''. From the second iteration, the method will soon find out, based on the matching results, that no relation can improve the score of the path as its context becomes less like the question. In this case, the condition to backtrack is

[1] https://hanlp.hankcs.com/

satisfied (refer to Equation 11), and the method should dismiss the "Crime" entity and step back to the "Taxi Driver" entity to test an alternative unvisited path. Algorithm 1 shows our path expansion method along with the termination/backtracking mechanism.

D. LOSS FUNCTION

We compute the cross-entropy loss as the following:

$$loss\,(z, y) = -\frac{1}{D}\sum_{t=1}^{D}[y_t * \log\,(s_t) + (1-y_t)\log(1-s_t)] \tag{12}$$

where D is the estimated length of the path from Equation 10. The gradient approach is adopted to minimize the cross-entropy between the predicted path score s_t from Equation 9 and the target y_t.

V. EVALUATION

A. DATASETS

To evaluate the proposed method for multi-hop KG-QA, we conducted experiments on three benchmarking datasets that are:

- **MetaQA**[40][2]: MetaQA (MoviE Text Audio QA) contains more that 400k questions for both single and multi-hop reasoning in the domain of movies. It has 1-hop, 2-hop and 3-hop questions with answers. Along with the QA data, MetaQA provides a knowledge graph consisting of 43k entities, 135k triples, and nine relations. We used the textual version of the dataset that is called "vanilla".
- **PathQuestion**[8][3]: is built by [8], and consists of two datasets: PathQuestion (PQ) and PathQuestion-Large (PQL). All questions require 2- or 3-hop relation path to reach answer entities, and are answerable through subsets of Freebase knowledge graph. The questions in PQL are more difficult to answer than those in PQ because PQL utilizes a larger knowledge base and provides fewer training instances.
- **WorldCup2014**[56][4]: It contains about 8000 questions with answers related to the 2014 World Cup. Questions are a mixture of 1-hop and 2-hop questions.

Statistics for the three datasets are shown in Table 1. Note that the three datasets differ in scale, domain, and complexity, a thing that helps us to better validate the effectiveness of our method.

B. EXPERIMENTAL SETTINGS

All word embedding vectors were obtained by using GloVe [57], with a dimension of 300. The ADAM optimizer [58] was used for parameter optimization, including gradient clipping by norm at a threshold of 5 and a learning rate of 0.001. All hyper-parameters were tuned on the development data. The batch size was set at 48. Hidden dimensions in the model were set to 200, and the dropout ratio was set to 0.2. We partitioned the entire dataset into the train/valid/test subsets as shown in Table 1. Note that the three datasets have topic entities annotated, thus no entity linking was required.

Algorithm 1 Multi-Hop KG-QA by Using Sequential Path Expansion With Backtracking

Initialize:
$t \leftarrow 1$ ▷ Iteration number
$p \leftarrow (e_q)$ ▷ p is the accumulated path
$e_t \leftarrow e_q$ ▷ e_t is the current entity at the t-th iteration, initialized with e_q
$s_t \leftarrow 0$ ▷ s_t is the matching score of the path at the t-th iteration
$s_{t-1} \leftarrow 0$ ▷ s_{t-1} is the matching score of the path at the previous iteration
Function: Expand_path(G, q, e_t, D)
Input: A knowledge graph G, question q, question entity e_t, the estimated distance D from e_q to e_a
Output: Answer entity e_a
Begin
$R_t \leftarrow \{r_1, r_2, \ldots, r_i, \ldots, r_k\}$ ▷ R_t is the set of outgoing relations of e_t
$P_t \leftarrow \emptyset$
for each $r_i \in R_t \wedge r_i$ *is unvisited* do
  $p_i \leftarrow p \oplus (r_i; \grave{e}_i)$ ▷ $\grave{e}_i$ is the destination entity of r_i
  $P_t \leftarrow P_t \cup \{p_i$ ▷ P_t is the candidate path set at t -th iteration
end for
$s_t \leftarrow$ *score elements in* P_t *and obtain the top score from Equation* 9
$p_t^{top} \leftarrow$ *Gettop – scored path from Equation* 8
if $s_t > s_{t-1}$ **then** ▷ Condition to proceed to the next entity
  $p \leftarrow p_t^{top}$
  $e_t \leftarrow tail(p)$
  $s_{t-1} \leftarrow s_t$
  $t \leftarrow t + 1$
  Expand_path (G, q, e_t, D)
else if $(s_t < s_{t-1}) \wedge (t < D) \wedge (e_t \neq e_q)$ **then** ▷ Condition to backtrack
  Retrieve r_{t-1} and e_{t-1} from G ▷ Not detailed for simplicity
  Retrieve s_{t-2} if exists from the score history, or set as 0 if does not exist
  $p \leftarrow p \ominus (r_{t-1}; e_t)$ ▷⊖ denotes de-concatenate operation
  $e_t \leftarrow e_{t-1}$
  $s_{t-1} \leftarrow s_{t-2}$
  $t \leftarrow t - 1$
  Expand_path (G, q, e_t, D)
else if $(s_t < s_{t-1} \wedge t \geq D) \vee (\forall r_i \in R_{t-1} : r_i isvisited)$ **then**
  $e_a \leftarrow tail(p_{t-1})$ ▷ Condition to terminate
  return e_a
end if
End

[2]https://github.com/yuyuz/MetaQA
[3]https://github.com/zmtkeke/IRN/tree/master/PathQuestion
[4]https://github.com/zmtkeke/IRN

TABLE 1. Statistics of benchmark datasets.

Dataset	#Entity	#Relation	#Triple	#Question	Hops (%)	Train	Validation	Test
MetaQA	43k	18	135k	407481	1 (29%)	96106	9992	9947
					2 (36%)	118948	14872	14872
					3 (34%)	114196	14274	14274
PQ	2215	14	4049	7106	2 (27%)	1526	191	191
					3 (73%)	4158	520	520
PQL	5035	364	9758	3653	2 (44%)	1272	159	160
					3 (56%)	1649	206	207
WC2014	1127	12	3977	7954	1 (80%)	5186	100	1196
					2 (20%)	1178	100	194

Following previous works [8], [14], [59], we measure the performance of models by the Hits@1 score, where the predicted answer is considered correct if it exactly matches the gold one. When a question has multiple answers, the predicted answer is considered correct if it matches any of the answers.

C. BASELINE

We compared our method with the following methods from the literature:

- **RL-MHR**: This is the RL-based method that uses stop and worth-trying signals to teach the agent when to stop, and is proposed by [41].
- **TransferNet**: This is the iterative method proposed by [45], which infers the next relation to hop along by transferring entity scores along of multiple steps.
- **UHop-HR**: this is the Unrestricted-Hop Relation Extraction Framework proposed by [12]. We implemented UHop on top of HR-BiLSTEM model and called it UHop-HR for short.
- **ISM**: This is the iterative matching model proposed by [13].
- IRN: This is the Interpretable Reasoning Network proposed by [8].
- **HR-BiLSTM**: This method uses hierarchical matching between questions and KB relations, and is proposed by [39]
- **Ours**: This is our sequential path expansion method that uses a compare-aggregate model for sequence matching and a termination/backtracking check to decide when to stop or consider another branch for path expansion.

Note that the IRN and HR-BiLSTM methods are based on an exhaustive search by traversing all paths in the KG over a predefined number of hops. They require setting the maximum number of hops in order to stop. Therefore, we implemented them with T=3 hops. In contract, RL-MHR. TransferNet, UHop-HR and ISM are similar to our method in that they are sequential decision-based methods based on hop-by-hop relation extraction. They also do not require setting the number of hops as they implement mechanisms for termination.

TABLE 2. Test results (% Hits@1) on vanilla of metaqa dataset.

`	Vanilla-1-hop	Vanilla-2-hop	Vanilla-3-hop
RL-MHR	92.7	81.3	82.5
TransferNet	92.4	83.2	85.1
UHop-HR	**96.2**	86.4	81.3
ISM	95.2	89.6	87.4
IRN	58.3	61.0	57.3
HR-BiLSTM	78.5	66.2	61.3
Ours	95.4	**94.1**	**93.4**

D. RESULTS AND DISCUSSION

Table 2, 3 and 4 summarize the comparison results between the baseline methods on the three datasets. The columns in each table show the Hits@1 scores for questions with different lengths of hops separately and when they are mixed together. Results show that our method achieves accurate results, and outperforms all other methods in most cases. In general, the methods that are based on exhaustive search, i.e. IRN, HR-BiLSTM, give less accurate results as compared to the sequential decision-based methods including RL-HMR, TransferNet, UHop-HR and ISM. This result is due to the fact that exhaustive search methods attempt to enumerate all paths, resulting in too many wrong paths mixed inside the training data. Thus, it is difficult for the model to distinguish the correct paths from the many wrong paths. What supports this conclusion is the poor performance of exhaustive search-based methods, which is much observable with MetaQA. In particular, MetaQA dataset has the largest KB as compared to other datasets, a thing that leads to a larger search space and more competing wrong answers.

When comparing our method with other sequential decision-based methods, we notice that all methods result in high accuracy values for 1-hop questions on all datasets without significant differences ($p = 0.08$). However, the difference becomes more noticeable when the complexity of the question increases. Our method clearly outperforms other methods with 3-hop questions. This can be particularly observed in the performance results on 3-hop questions in

TABLE 3. Test results (% Hits@1) on pq and pql of pathquestion dataset.

	PQ-2-hop	PQ-3-hop	PQ-M-hop	PQL-2-hop	PQL-3-hop	PQL-M-hop
RL-MHR	94.1	87.2	80.8	82.2	77.8	78.4
TransferNet	93.2	91.3	91.8	84.1	82.7	83.0
UHop-HR	97.6	91.3	84.3	82.6	80.1	80.2
ISM	**99.1**	95.7	88.2	84.9	81.7	81.3
IRN	78.3	74.3	75.9	66.2	59.1	63.0
HR-BiLSTM	76.8	74.1	74.9	71.9	61.6	66.2
Ours	97.4	**98.7**	**96.4**	**92.3**	**89.7**	**91.4**

TABLE 4. Test results (% Hits@1) on worldcup2014 dataset.

	WC-1-hop	WC-2-hop	WC-M-hop
RL-MHR	94.8	93.6	92.1
TransferNet	97.9	96.5	96.8
UHop-HR	**98.1**	96.6	95.1
ISM	96.3	98.0	**97.3**
IRN	71.2	66.2	64.7
HR-BiLSTM	86.5	83.2	72.3
Ours	97.4	**98.6**	96.0

MetaQA and QuestionPath, with a significant difference at $p < 0.05$. The difference is insignificant ($p = 0.1$) in the WorldCup2014 since it is the simplest dataset in terms of the complexity of questions, and does not contain 3-hop questions.

In 3-hop questions, there is more possibility to prune away correct relations during the iterative reasoning process. This makes it difficult to reach a correct answer entity with incorrect intermediate relations. This claim is supported by looking at the experimental results of related works [8], [13], [14], all of which reported that the error generated in sequential decisions accumulates as the complexity of the question increases. The superiority of our method over other methods in the case of 3-hop questions can be attributed to the low accumulated error as a result of the termination/ backtracking check carried out at each hop. Once the similarity between the path and the question begins to decline, our method can backtrack to consider an alternative branch to move forward.

To have a deeper insight on this difference, we analyzed the error cases in RL-HMR, TransferNet, UHop-HR and ISM, and found that the majority of errors (62%) occurred due to incorrect relation matching at the first hop. This is due to the lack of enough features for effective reasoning at the beginning of the path expansion process. Furthermore, we inspected sample 3-hop questions that were incorrectly answered by the former methods and found that many were of 3-hop questions whose question entities are connected to the answer entities via multiple paths. Other methods answered them incorrectly because

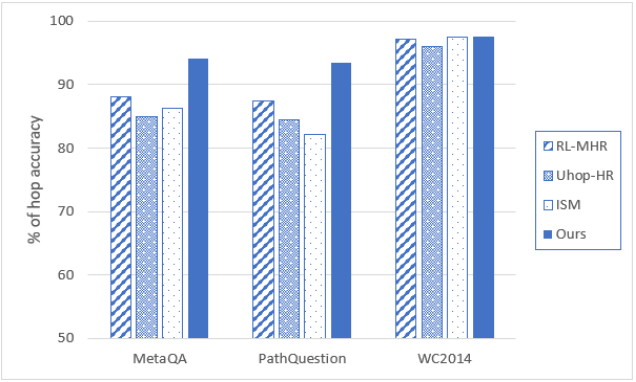

FIGURE 5. Comparison of hop number accuracy by RL-MHR, UHop-HR, ISM and our method across the three datasets.

they took paths that did not lead to an answer entity. For example, the question: "Who starred films for the screenwriter of 'Taxi Driver'?" requires taking the relation path: (Texi Driver $\overset{\text{written\_by}}{\longrightarrow}$ Paul Schrader $\overset{\text{written\_by}}{\longleftarrow}$ Rolling Thunder $\overset{\text{starred\_actors}}{\longrightarrow}$ Tommy Lee Johns) to be answered correctly (refer to Fig. 1). However, other methods took the path (Taxi Driver $\overset{\text{starred\_actors}}{\longrightarrow}$ Robert De Niro) instead due to incorrect sequence matching between the path and the question. In fact, our method behaved similarly with this question in the first iteration, but this mistake was spotted and corrected by the backtracking check.

ISM is almost the second best performing method across all datasets, followed by UHop-HR, followed by TransferNet, and finally RL-HMR. In fact, all the former methods except RL-HMR employ attention layers for effective matching between the relation path and the question embeddings. This indicates that the methods that use the attention mechanism can result in more accurate results than the non-attentive methods.

In addition, we compared the accuracy of our termination check mechanism with similar stop mechanisms used in RL-MHR, UHop-HR, ISM. The termination accuracy is estimated on the basis of the ability to detect the hop number accurately. Fig. 5 shows the accuracy results on hop numbers across the three datasets. In general, all methods perform well and can detect the hop number accurately on the WorldCup2014 because it is the simplest dataset without 3-hop questions. Our method surpasses others on MetaQA and QuestionPath datasets, in particular, with a significant difference at $p < 0.05$.

Inspection of erroneous results has again attributed this difference to the false decision on paths, especially with 3-hop questions that have multiple answer paths. The decision to take a shorter wrong path for several questions caused the other models to obtain an early stop signal. Such scenarios occurred a lot in PathQuestion and MetaQA datasets.

In fact, the stop mechanisms implemented in UHop-HR, ISM rely on the change in the similarity score of the question with the relation path, and trigger the stop signal when it is not possible to reach a better similarity any more.

However, this does not apply in the first hop when there is no prior similarity score to compare with. RL-MHR resulted in the second highest termination accuracy because it uses an RL-reasoner to train the agent when to stop based on the search history. However, RL-based methods heavily rely on the terminal reward to bias the search. Thus, the agent can still be misled and hop to a false relation due to sparse or delayed rewards [44].

VI. ABLATION STUDIES

We performed model ablation studies to reveal the contribution of each component in our method. We compared between different modified versions of our method, which are summarized in Table 5.

- **V-NONE**: all core features are dismissed from this version of our method as the following: The path ranking and pruning mechanism is dismissed so that the method should search all paths to find candidate answers. The compare-aggregate model with attention is disabled. Instead, the question and path sequences are projected into single vectors, and compared by using the cosine measure. This version also runs no termination/backtracking check.
- **V-Rank**: This version is similar to V-NONE but with path ranking and pruning feature enabled.
- **V-ComAgg**: This version is similar to V-NONE, but uses the compare-aggregate model with attention mechanism and multiply-subtract comparison functions.
- **V-Rank-ComAgg**: This version performs path ranking and pruning, and uses the compare-aggregate model with attention. Only the termination/path validation check is removed.
- **V-Rank-Stop-Backtrack**: This version performs path ranking and pruning and performs termination/backtracking check. Only the compare-aggregate model with attention is removed and replaced with cosine similarity between vector representations as in V-NONE.
- **V-ours**: Our method with all features enabled.

Note that the termination/backtracking check cannot be tested independently of the path ranking and pruning because the ranking scores are required to trigger the stop/backtrack events.

Table 6 shows the performance results in terms of Hits@1 score of the five versions on the three datasets. In general, the V-NONE model, which solely relies on exhaustive search of candidate paths, performs worse than all over versions. V-ComAgg performs better than the V-NONE model, indicating the advantage of using the compare-aggregate model. The encoding of question-path sequences independently as embedding vectors in V_NONE is likely not sufficient to capture all the important information, as in the attentive model.

When path ranking and pruning is enabled in V-Rank-ComAgg, the accuracy increases significantly compared to the accuracy of V-NONE and V-ComAgg (the difference is significant at $p < 0.05$). This indicates that path pruning is effective not only in reducing the search complexity, but also in improving the accuracy by eliminating many of the false-positive paths from the training process. V-Rank-Stop-Backtrack, which has the termination/backtracking check enabled, performs better than the previous versions, and the difference is obvious in MetaQA and QuestionPath, which include 3-hop questions. Our overall method is better than V-Rank-Stop-Backtrack with a significant difference, showing that the combination of all features is the most effective.

TABLE 5. Variants of our method used in the ablation study.

Variant name	Path ranking and pruning	Compare-aggregate model with attention	Termination/backtracking check
V-NONE			
V-ComAgg		✓	
V-Rank-ComAgg	✓	✓	
V-Rank-Stop-Backtrack	✓		✓
V-ours	✓	✓	✓

TABLE 6. Ablation experiment results in terms of the Hits@1 score of the five variants over the three datasets.

Variant name	MetaQA	PathQuestion	WC2014
V-NONE	61.8	48.7	81.2
V-ComAgg	76.2	61.7	88.4
V-Rank-ComAgg	92.4	89.4	92.8
V-Rank-Stop-Backtrack	90.8	91.2	93.3
V-ours	94.2	93.5	97.4

VII. LIMITATION ANALYSIS

In general, the datasets used in this study and existing works consist of questions that are generated from textual templates. Most of the patterns of these templates can be easily captured by using a deep learning model with sufficient training data. Inspection of the errors generated by our method showed that about 78% of the errors came from incorrect termination without detecting the correct answer path, especially in the PQL2 dataset. Although the introduced backtracking mechanism could reduce this error, it sometimes does not help when the model predicts incorrect relations due to insufficient training samples. We conclude that having no sufficient samples of the predicted length for training still impairs performance. In addition, complex questions that involve constrained relations or numerical operations cannot be easily answered by traversal-based methods. When questions become complicated from both semantic and syntactic aspects, models will need additional capabilities of natural language understanding and generalization.

VIII. CONCLUSION AND FUTURE WORK

In this work, we proposed a deep learning-based method for multi-hop question answering over knowledge graphs. Our method attempts to reduce the search space by progressively tracing the path that is more similar to the question. To resolve the problem of accumulated error due to incorrect reasoning over relations, we proposed a novel termination/backtracking mechanism that validates the answer path as it expanded to decide whether to stop, backtrack or proceed. The proposed method also uses a compare-aggregate model for more effective semantic matching between the question and the candidate paths. Our method also employs the attention mechanism to decide which part of a given path should be focused on in the current iteration. Comparison with other baseline methods from the literature over three benchmark datasets has shown that the backtracking mechanism can effectively reduce the errors. Our method remarkably outperforms similar methods especially with 3-hop questions which are more error-prone due to the longer distance of hops needed to reach the answer entity.

In our future work, our aim is to investigate how to improve the model to handle complex questions with constraints. We will attempt to reduce errors in relation matching at the first hop by using dependency parsing and linguistic analysis of the question.

REFERENCES

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2008, pp. 1247–1250.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," in *The Semantic Web*. Berlin, Germany: Springer, 2007, pp. 722–735.

[3] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," 2016, *arXiv:1606.03391*.

[4] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," 2015, *arXiv:1506.02075*.

[5] D. Savenkov and E. Agichtein, "EviNets: Neural networks for combining evidence signals for factoid question answering," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 299–304.

[6] C. Ran, W. Shen, J. Wang, and X. Zhu, "Domain-specific knowledge base enrichment using Wikipedia tables," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 349–358.

[7] W.-T. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2014, pp. 643–648.

[8] M. Zhou, M. Huang, and X. Zhu, "An interpretable reasoning network for multi-relation question answering," 2018, *arXiv:1801.04726*.

[9] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang, "Cognitive graph for multi-hop reading comprehension at scale," 2019, *arXiv:1905.05460*.

[10] S. Kundu, T. Khot, A. Sabharwal, and P. Clark, "Exploiting explicit paths for multi-hop reading comprehension," 2018, *arXiv:1811.01127*.

[11] Y. Cao, M. Fang, and D. Tao, "BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering," 2019, *arXiv:1904.04969*.

[12] Z.-Y. Chen, C.-H. Chang, Y.-P. Chen, J. Nayak, and L.-W. Ku, "UHop: An unrestricted-hop relation extraction framework for knowledge-based question answering," 2019, *arXiv:1904.01246*.

[13] Y. Lan, S. Wang, and J. Jiang, "Multi-hop knowledge base question answering with an iterative sequence matching model," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2019, pp. 359–368.

[14] Y. Qiu, Y. Wang, X. Jin, and K. Zhang, "Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 474–482.

[15] M. A. Borroto, F. Ricca, and B. Cuteri, "A system for translating natural language questions into SPARQL queries with neural networks: Preliminary results," in *Proc. 29th Italian Symp. Adv. Database Syst.*, 2021, pp. 1–12.

[16] N. Bhutani, X. Zheng, and H. V. Jagadish, "Learning to answer complex questions over knowledge bases with query composition," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 739–748.

[17] K. Roberts and B. G. Patra, "A semantic parsing method for mapping clinical questions to logical forms," in *Proc. AMIA Annu. Symp.*, 2017, pp. 1478–1487.

[18] S. Reddy, O. Täckström, M. Collins, T. Kwiatkowski, D. Das, M. Steedman, and M. Lapata, "Transforming dependency structures to logical forms for semantic parsing," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 127–140, Dec. 2016.

[19] P. Jain and M. Lapata, "Memory-based semantic parsing," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 1197–1212, Nov. 2021.

[20] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," 2017, *arXiv:1709.00103*.

[21] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," 2018, *arXiv:1809.08887*.

[22] P. Kapanipathi, I. Abdelaziz, S. Ravishankar, S. Roukos, A. Gray, R. Astudillo, M. Chang, C. Cornelio, S. Dana, A. Fokoue, and D. Garg, "Leveraging abstract meaning representation for knowledge base question answering," 2020, *arXiv:2012.01707*.

[23] S. Chen, Q. Liu, Z. Yu, C.-Y. Lin, J.-G. Lou, and F. Jiang, "ReTraCk: A flexible and efficient framework for knowledge base question answering," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process., Syst. Demonstrations*, 2021, pp. 325–336.

[24] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 105–113.

[25] A. Saxena, A. Tripathi, and P. Talukdar, "Improving multi-hop question answering over knowledge graphs using knowledge base embeddings," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 4498–4507.

[26] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," 2014, *arXiv:1406.3676*.

[27] R. Das, M. Zaheer, S. Reddy, and A. McCallum, "Question answering on knowledge bases and text using universal schema and memory networks," 2017, *arXiv:1704.08384*.

[28] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer, "Neural network-based question answering over knowledge graphs on word and character level," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 1211–1220.

[29] M.-C. Yang, D.-G. Lee, S.-Y. Park, and H.-C. Rim, "Knowledge-based question answering using the semantic embedding space," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9086–9104, Dec. 2015.

[30] K. Qin, Y. Wang, C. Li, K. Gunaratna, H. Jin, V. Pavlu, and J. A. Aslam, "A complex KBQA system using multiple reasoning paths," 2020, *arXiv:2005.10970*.

[31] L. Dong, F. Wei, M. Zhou, and K. Xu, "Question answering over freebase with multi-column convolutional neural networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, vol. 1, 2015, pp. 260–269.

[32] L. Chen, G. Zeng, Q. Zhang, X. Chen, and D. Wu, "Question answering over knowledgebase with attention-based LSTM networks and knowledge embeddings," in *Proc. IEEE 16th Int. Conf. Cognit. Informat. Cognit. Comput. (ICCI*CC)*, Jul. 2017, pp. 243–246.

[33] Z. Dai, L. Li, and W. Xu, "CFO: Conditional focused neural question answering with large-scale knowledge bases," 2016, *arXiv:1606.01994*.

[34] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," 2018, *arXiv:1809.09600*.

[35] Y. Feng, X. Chen, B. Y. Lin, P. Wang, J. Yan, and X. Ren, "Scalable multi-hop relational reasoning for knowledge-aware question answering," 2020, *arXiv:2005.00646*.

[36] R. Das, A. Godbole, D. Kavarthapu, Z. Gong, A. Singhal, M. Yu, X. Guo, T. Gao, H. Zamani, M. Zaheer, and A. McCallum, "Multi-step entity-centric information retrieval for multi-hop question answering," in *Proc. 2nd Workshop Mach. Reading Question Answering*, 2019, pp. 113–118.
[37] M. Tan, C. dos Santos, B. Xiang, and B. Zhou, "LSTM-based deep learning models for non-factoid answer selection," 2015, *arXiv:1511.04108*.
[38] K. Nishida, K. Nishida, M. Nagata, A. Otsuka, I. Saito, H. Asano, and J. Tomita, "Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction," 2019, *arXiv:1905.08511*.
[39] M. Yu, W. Yin, K. S. Hasan, C. dos Santos, B. Xiang, and B. Zhou, "Improved neural relation detection for knowledge base question answering," 2017, *arXiv:1704.06194*.
[40] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 6069–6076.
[41] R. Das, A. Godbole, A. Naik, E. Tower, R. Jia, M. Zaheer, H. Hajishirzi, and A. McCallum, "Knowledge base question answering by case-based reasoning over subgraphs," 2022, *arXiv:2202.10610*.
[42] G. Niu, Y. Li, C. Tang, Z. Hu, S. Yang, P. Li, C. Wang, H. Wang, and J. Sun, "Path-enhanced multi-relational question answering with knowledge graph embeddings," 2021, *arXiv:2110.15622*.
[43] J. Shi, S. Cao, L. Hou, J. Li, and H. Zhang, "TransferNet: An effective and transparent framework for multi-hop question answering over relation graph," 2021, *arXiv:2104.07302*.
[44] G. He, Y. Lan, J. Jiang, W. X. Zhao, and J.-R. Wen, "Improving multi-hop knowledge base question answering by learning intermediate supervision signals," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, Mar. 2021, pp. 553–561.
[45] R. Koner, H. Li, M. Hildebrandt, D. Das, V. Tresp, and S. Günnemann, "Graphhopper: Multi-hop scene graph reasoning for visual question answering," in *Proc. Int. Semantic Web Conf.* Albany, NY, USA: Springer, 2021, pp. 111–127.
[46] G. Wan and B. Du, "GaussianPath: A Bayesian multi-hop reasoning framework for knowledge graph reasoning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, 2021, pp. 4393–4401.
[47] W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," 2017, *arXiv:1707.06690*.
[48] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," 2018, *arXiv:1808.10568*.
[49] J. Liao, X. Zhao, J. Tang, W. Zeng, and Z. Tan, "To hop or not, that is the question: Towards effective multi-hop reasoning over knowledge graphs," *World Wide Web*, vol. 24, no. 5, pp. 1837–1856, Sep. 2021.
[50] F. Hill, A. Bordes, S. Chopra, and J. Weston, "The goldilocks principle: Reading children's books with explicit memory representations," 2015, *arXiv:1511.02301*.
[51] Z. Xu, H.-T. Zheng, Z. Fu, and W. Wang, "Enhancing question understanding and representation for knowledge base relation detection," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 1362–1367.
[52] S. Wang and J. Jiang, "A compare-aggregate model for matching text sequences," 2016, *arXiv:1611.01747*.
[53] Y. Yang and M.-W. Chang, "S-MART: Novel tree-based structured learning algorithms applied to tweet entity linking," 2016, *arXiv:1609.08075*.
[54] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," 2015, *arXiv:1503.00075*.
[55] J. Lu, Z. Zhang, X. Yang, and J. Feng, "Efficient subgraph pruning & embedding for multi-relation QA over knowledge graph," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.
[56] L. Zhang, J. Winn, and R. Tomioka, "Gaussian attention model and its application to knowledge base embedding and question answering," 2016, *arXiv:1611.02266*.
[57] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[59] X. Li, M. Alazab, Q. Li, K. Yu, and Q. Yin, "Question-aware memory network for multi-hop question answering in human-robot interaction," 2021, *arXiv:2104.13173*.

IYAD ALAGHA received the Ph.D. degree in computer science from the University of Durham, U.K., in 2009. He worked as a Research Associate at the University of Durham, from 2009 to 2012, in the field of technology enhanced learning. In 2012, he joined as the Faculty of IT, Islamic University of Gaza, as a Lecturer. He was the Dean of the Faculty, from 2019 to 2021. He is currently an Associate Professor with the Department of Software Development. He has published articles in several international journals and conferences. His research interests include deep learning, data analytics, NLP, and semantic web technologies. Besides his academic and research experience, he works as a senior android developer and a trainer. He also works as an IT consultant and a curriculum developer for several institutions.