# Joint Knowledge Graph Completion and Question Answering

Lihui Liu*, Boxin Du*, Jiejun Xu†, Yinglong Xia‡, Hanghang Tong*
*Department of Computer Science, University of Illinois at Urbana Champaign
† HRL jxu@hrl.com
‡ Meta yxia@fb.com
USA
{lihuil2,boxindu2,htong}@illinois.edu

## ABSTRACT

Knowledge graph reasoning plays a pivotal role in many real-world applications, such as recommendation, computational fact-checking, enterprise knowledge management, and many more. Among these applications, knowledge graph completion (KGC) and multi-hop question answering over knowledge graph (Multi-hop KGQA) are two representative reasoning tasks. In the vast majority of the existing works, the two tasks are considered separately with different models or algorithms. However, we envision that KGC and Multi-hop KGQA are closely related to each other. Therefore, the two tasks will benefit from each other if they are approached adequately. In this work, we propose a neural model named BiNet to jointly handle KGC and multi-hop KGQA, and formulate it as a multi-task learning problem. Specifically, our proposed model leverages a shared embedding space and an answer scoring module, which allows the two tasks to automatically share latent features and learn the interactions between natural language question decoder and answer scoring module. Compared to the existing methods, the proposed BiNet model addresses both multi-hop KGQA and KGC tasks simultaneously with superior performance. Experiment results show that BiNet outperforms state-of-the-art methods on a wide range of KGQA and KGC benchmark datasets.

## CCS CONCEPTS

• **Computing methodologies** → **Reasoning about belief and knowledge**; • **Information systems** → **Data mining**.

## KEYWORDS

Knowledge graph question answering; Knowledge graph completion; Multi-task learning

**ACM Reference Format:**
Lihui Liu*, Boxin Du*, Jiejun Xu†, Yinglong Xia‡, Hanghang Tong*. 2022. Joint Knowledge Graph Completion and Question Answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/XXXXXX.XXXXXX


*KDD '22, August 14–18, 2022, Washington, DC, USA.*

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/XXXXXX.XXXXXX

## 1 INTRODUCTION

A knowledge graph is a graph structure that contains a collection of facts, where nodes represent real-world entities, events and objects, and edges denote the relationship between two nodes. Since its debut in 2012,[1] a variety of knowledge graphs have been generated, including Freebase, Yago, Wikidata and so on. The applications of knowledge graphs are numerous, ranging from network alignment [24], computational fact-checking [12] to recommendation [5]. Among these applications, multi-hop question answering over knowledge graph (Muti-hop KGQA for short) aims to answer natural language questions with the help of knowledge graphs. Knowledge graph completion (KGC), on the other hand, seeks to infer missing facts based on existing information in the knowledge graph.

Recently, KGQA and KGC tasks have attracted great attention from both the academia and the industry, and a multitude of algorithms have been proposed. For example, for KGC, TransE [2] models the relation as a linear transition of points in the embedding space, ComplEx [22] embeds relations and entities in complex space and makes predictions according to an energy function. For multi-hop question answering, Pullnet [19] first extracts question specific subgraphs, and then performs multi-hop reasoning on the extracted subgraph via graph neural networks to find answers, EmbedKGQA [18] uses a pre-trained BERT model to map natural language questions to relation embeddings and finds answers by ComplEx [22]. We note that some work [18] treats the knowledge graph completion task as a single-hop knowledge graph question answering task due to their interchangeable properties. For example, the task of predicting the answer for triple (`Interstellar`, `hasGenre`, ?) in the KGC task could be transformed to answer natural language question "*What is the genre of Interstellar?*".

Despite the potential close relationship between multi-hop KGQA and KGC tasks [18], existing works usually treat them as two separate tasks without considering their reciprocal benefits. That is, most existing multi-hop KGQA methods have implicitly assumed the background knowledge graph is complete [19, 20], whereas the existing KGC methods only exploit the existing information of the input incomplete knowledge graphs, e.g., RESCAL [17] and ComplEx [22].

Different from existing methods, we envision that knowledge graph completion (KGC) and knowledge graph question answering (KGQA) are inherently complementary with each other due to the following reasons. First, *(R1) KGQA helps KGC.* This is because new knowledge could be inferred from the KGQA task, which in turn could be used to complete the knowledge graph. For example, given

[1] https://en.wikipedia.org/wiki/Knowledge_graph

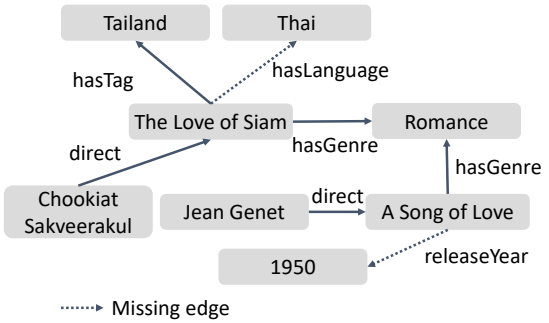



**Figure 1: An illustrative example of incomplete knowledge graph. The dashed line represents missing links. The solid lines are relations in the existing knowledge graph.**

the knowledge graph in Figure 1, if we want to answer "*which year was A Song of Love released?*", there is no way this question could be answered because `A song of Love` only has two relations around it: `direct` and `hasGenre`. Even a human could not answer this question only based on the existing knowledge graph. However, if we are provided the answer (`1950`) to the question "*which years were all the films directed by Jean Genet released?*", we can infer that the release year of `A song of Love` is 1950 because `Jean Genet` only directed one film in his life. This suggests that KGQA could indeed help KGC. Second, *(R2) KGC helps KGQA*. This is because KGC could help improve the performance of KGQA by providing a KG with more complete knowledge triples of high quality. For example, because the movie `The Love of Siam` is linked to `Tailand` via the `hasTag` relation, an ideal KGC model can infer that the movie might be intended for the `Thai` audiences and augment the knowledge graph with appropriate language information for the movie. Subsequently, the question "*what is the language of the film The Love of Siam*" can then be trivially answered.

Armed with this key insight, we propose to jointly address multi-hop KGQA and KGC tasks. We formulate it as a multi-task learning problem. First, in order to leverage multi-hop KGQA for the KGC task, we propose an encoder-decoder-based model which transforms the natural language questions into relation paths to facilitate KGC. Second, in order to leverage KGC for multi-hop KGQA, we let multi-hop KGQA and KGC share both the embedding space and the answer scoring module, which allows them to automatically share latent features and reinforce each other.

In summary, the main contributions of this paper are:

- **Problem Definition.** To our best knowledge, we are the first to formulate joint knowledge graph completion and multi-hop question answering as a multi-task learning problem.
- **Algorithm and Analysis**. We propose a multi-task neural model BiNet which could solve KGQA and KGC tasks at the same time and we further analyze its theoretic feasibility.
- **Empirical Evaluations**. The experimental results on several real-world datasets demonstrate that the proposed BiNet consistently achieves state-of-the-art performance in both tasks.

## 2 PROBLEM DEFINITION

Table 1 gives the main notations used throughout this paper. A knowledge graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$ where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of nodes/entities, $\mathcal{R} = \{r_1, r_2, ..., r_m\}$ is the set of relations and $\mathcal{L}$ is the list of triples. Each triple in the knowledge graph can be denoted as $(h, r, t)$ where $h \in \mathcal{V}$ is the head (i.e., subject) of the triple, $t \in \mathcal{V}$ is the tail (i.e., object) of the triple and $r \in \mathcal{R}$ is the edge (i.e., relation, predicate) of the triple which connects the head $h$ to the tail $t$. The embedding of a node or relation is represented by bold lowercase letters, e.g., $\mathbf{e}_i$, $\mathbf{r}_i$.

**Table 1: Notations and definitions**

| Symbols | Definition |
|---|---|
| $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$ | the knowledge graph |
| $v_i$ | the $i^{\text{th}}$ entity/node in knowledge graph |
| $r_i$ | the $i^{\text{th}}$ relation/edge in knowledge graph |
| $\mathbf{e}_{v_i}$ / $\mathbf{e}_i$ | the embedding of node $v_i$ |
| $\mathbf{r}_i$ | the embedding of relation $r_i$ |
| $Q$ | the multi-hop natural language question |
| $\overline{Q}$ | the multi-hop natural language question training set |
| $A_Q$ | the answer set of question $Q$ |
| $v_Q$ | the topic entity in question |
| $\mathbf{e}_Q$ | the embedding of topic entity $v_Q$ |
| $\mathbf{Q}$ | the embedding of natural language question $Q$ |
| $w_i$ | the $i^{\text{th}}$ word in $Q$ |
| $P$ | the path decoded from question embedding |
| $P[i]$ | the $i^{\text{th}}$ relation in path $P$ |

Given a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$ and a natural language question $Q$ which contains a topic entity $v_Q \in \mathcal{V}$ and a sequence of words $Q = (w_1, w_2, ..., w_{|Q|})$, multi-hop question answering over knowledge graph aims to identify a set of nodes $A_Q \subseteq \mathcal{V}$ to answer this question. Following the definition in [18], we assume that all the answer entities exist in the knowledge graph and each question in multi-hop KBQA only contains a single topic entity $v_Q \in \mathcal{V}$ and $v_Q$ is given. For example, `The Love of Siam` is the topic entity of "*what is the language of the film The Love of Siam*". Ideally, each question can be mapped to a unique path $P = (r_1, r_2, ..., r_{|P|})$ in the knowledge graph.

Knowledge graph completion intends to infer missing facts/triples based on existing information in the knowledge graph. Typically, KGC contains three kinds of sub-tasks: (1) given a partial triple $(h, r, ?)$, predict the corresponding tail entities; (2) given a partial triple $(?, r, t)$, predict the corresponding head entities; (3) given a head entity and a tail entity, predict the relationship between them. In this paper, we only consider the first sub-task. When predicting the tail entities, the KGC method will produce a probability score for each entity in the knowledge graph, the probability score of a candidate entity $v$ denotes how likely the triple $(h, r, v)$ is true.

Many real-world knowledge graphs are incomplete. That is, some key information may not exist in the input knowledge graph. Performing KGQA on an incomplete knowledge graph could lead to wrong answers. However, if the knowledge graph is complete, the KGQA task is more likely to find the correct answers. On the other hand, completing an existing knowledge graph without any extra information might be hard. However, the question and answer pairs in the KGQA task could provide auxiliary information to help complete the knowledge graph. Based on this observation, we aim to jointly handle KGC and multi-hop KGQA, which can be formally defined as follows.

**Problem Definition 1.** *Jointly multi-hop KGQA and KGC*

***Given:*** *(1) A set of training triples of the KGC task, (2) a set of training multi-hop natural language questions of the KGQA task;*

***Output:*** *(1) The answer for test triples of the KGC task, and (2) the* top-$k$ *answers for test multi-hop natural language questions of the KGQA task.*

## 3 PROPOSED METHOD

In this section, we present the proposed model to solve Problem 1.

### 3.1 Model Overview

Multi-hop question answering over knowledge graph (Multi-hop KGQA) can be cast as an entity seeking problem on knowledge graph $\mathcal{G}$ by translating $Q$ into a query path and traveling the knowledge graph to find answers. However, in many real-world cases, the knowledge graph is often incomplete. Thus, attempting to find the answer set $A_Q$ directly on $\mathcal{G}$ by path traverse or subgraph matching [14] is impractical. Knowledge graph completion (KGC), on the other hand, often suffers from *knowledge famine* (i.e., insufficient information) or key missing information, which makes it impossible to complete the knowledge graph. For example, in Figure 1, if the relation `releaseYear` between `A Song of Love` and `1950` is missing, this missing information is impossible to be completed without extra information. As mentioned in Section 2, the multi-hop KGQA and KGC are inherently complementary with each other. On one hand, the information in the natural language question and its corresponding answers can help the knowledge graph completion task. On the other hand, a more complete knowledge graph could potentially improve the KGQA accuracy. To mutually reinforce these two tasks, our model is designed to bear the following two properties. First, to help the KGC task, we use an encoder-decoder model to transform the natural language query to a path that contains one or multiple relations. Second, in our model, both KGQA and KGC leverage the same embedding space and the answer scoring module to automatically share latent features.

Figure 2 shows the framework of our proposed model. Given a multi-hop natural language question with a topic entity, the proposed BiNet first generates a question's embedding $\mathbf{Q}$ via a pre-trained BERT [4] model, and then uses a decoder to generate a relation path which is used by both KGQA and KGC tasks. In order to generate a high-quality path, BiNet uses a probability model to choose the path with the highest probability in the knowledge graph that best interprets the question context. After obtaining the path, the answer scoring module ranks all the nodes and selects $k$ candidates which are most likely to answer the question. Finally, the topic entity $v_Q$, the path $P$, and $k$ candidate nodes are treated as the input of a Transformer [23] to generate the final output of the KGQA task. In addition, the decoded path $P$ is used by the answer scoring module to help the knowledge graph completion task. The overall model can be optimized in an end-to-end manner by combining the loss of different parts. Algorithm 1 and Algorithm 2 in Supplementary Material summarize the pseudocodes.

### 3.2 Question Encoder-Decoder

**A - Preprocessing Text for the Question Encoder.** The proposed BiNet decodes a sequence of relations between the topic entity $v_Q$ and an answer set $A_Q$ in a natural language sentence $Q = (w_1, w_2, ..., w_{|Q|})$ where $v_Q \in \mathcal{V}$ and $A_Q \subseteq \mathcal{V}$. Intuitively, each question context $Q$ could be mapped to a relation path in the knowledge graph distinctively. Therefore, we train an encoder to represent the context as a vector.

To mitigate the noise brought by the surface forms of the entities, we introduce a special token [NE] to mask the topic entity inside the question context/surface form (e.g., "*Who starred Interstellar?*" becomes "*Who starred [NE]?*"). Masking the topic entity prevents the encoder from memorizing the surface forms of the entity and helps it generalize to similar questions involving other entities. Besides this, we add two indicator tokens ([CLS] and <s>) to the beginning and end of the question context to signify its boundary.

**B - Question Encoder.** Given the processed question context, we first pass it through a pre-trained BERT [4] to extract contextual embeddings for each token[2]:

$$[\mathbf{h}_{CLS}, \mathbf{w}_1, ..., \mathbf{w}_{|Q|}, \mathbf{h}_\mathbf{s}] = \text{BERT}([CLS], w_1, ..., w_{|Q|}, < s >) \quad (1)$$

where $\mathbf{h}_{CLS}$ is the embedding of the [CLS] token and $\mathbf{h}_\mathbf{s}$ is the embedding of the <s> token. The final question embedding is obtained from the combination of $\mathbf{h}_{CLS}$ and $\mathbf{h}_\mathbf{s}$ as below, where FFN is a feed forward neural network, and | indicates concatenation.

$$\mathbf{h}_Q = \text{FFN}([\mathbf{h}_{CLS}|\mathbf{h}_\mathbf{s}]) \quad (2)$$

**C - Question Decoder.** Given the question context embedding $\mathbf{h}_Q$, the proposed BiNet decodes $\mathbf{h}_Q$ and generates a sequence of relations $P = (r_1, r_2, ..., r_n)$ using a Long Short-Term Memory (LSTM) [8] model. The initial hidden state $\mathbf{h}_0$ and initial cell state $\mathbf{c}_0$ are obtained from question embedding $\mathbf{h_Q}$ by passing it through two feed forward neural networks, separately.

$$\mathbf{h}_0 = \text{FFN}_h(\mathbf{h}_Q) \qquad \mathbf{c}_0 = \text{FFN}_c(\mathbf{h}_Q) \quad (3)$$

The initial input embedding $\mathbf{x}_0$ could be the question embedding $\mathbf{h_Q}$ or a zero vector. At time step $t$, the hidden state $\mathbf{h}_t$ will be passed through a feed forward neural network with batch normalization and dropout followed by a softmax function to obtain the score of each relation to form a score vector:

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{o}_{t-1})$$
$$\mathbf{a}_t = \text{softmax}(\text{MLP}(\mathbf{h}_t))$$

The new input embedding $\mathbf{x}_{t+1}$ of the decoder at step $t+1$ is the weighted sum of all relation embedding of step $t$.

$$\mathbf{x}_{t+1} = \sum_i \mathbf{a}_t(i)\mathbf{r}_i$$

where $\mathbf{a}_t(i)$ is the $i$-th element of $\mathbf{a}_t$. The final prediction of each step is the relation with the highest score.

**D - Training Question Encoder-Decoder.** Given a natural language question with its answer set, we want to map the question context to its correct relation path. The topic entity $v_Q$ and answer entities can be identified in the knowledge graph according to their surface forms. However, there might exist many paths between them. To identify the correct path, we find all the $k$-shortest paths between each entity pair $(v_Q, v_i)$ where $v_i \in A_Q$. We treat all these shortest paths as potentially correct path candidates. We use Bayes' Rule to infer the probability of whether the shortest path is the correct mapping of the question context. First, all the questions in the

[2]For dataset MetaQA, the surface forms of KGC training triples are treated as the input questions.

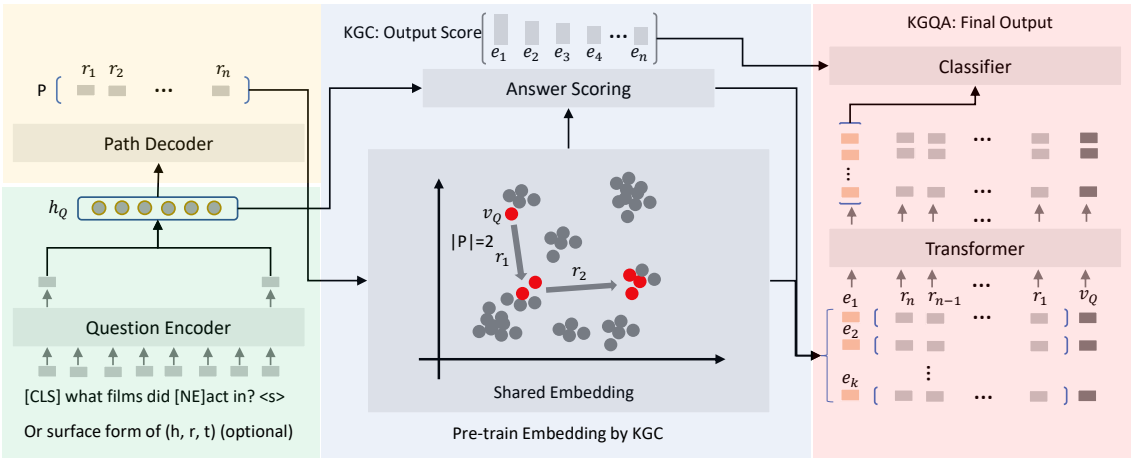



**Figure 2: Architecture of our proposed BiNet Model. The gray part corresponds to Subsectioon 3.3 which is used by both the KGQA and KGC tasks. The red parts are designed for the KGQA task, while the yellow part is intended for the KGC task. The green part will be used by KGQA and KGC tasks. The shared embedding space contains pre-trained embeddings of entities and relations by the training data of KGC. Best viewed in color.**

training set can be divided into $m$ groups: $S_1, S_2, .., S_m$ where $m$ is the number of unique question contexts after masking the topic entity. Each group has a corresponding answer pool $PL_i = \{A_{Q_j} | Q_j \in S_i\}$. Each answer entity $v_i \in A_{Q_j}$ has a corresponding candidate path set $PC(Q_j, v_i) = \{P_i | (v_{Q_j}, P_i, v_i) \in \mathcal{G}\}$. If we assume the occurrence of different paths in $PC$ as i.i.d. random variables, the probability that a path interprets the question context can be expressed as

$$Pr(P_i|S_j, \theta) = \frac{\sum_{Q_j \in S_j} Pr(P_i, Q_j|\theta)}{|PL_j|}$$

where $|PL_j|$ is the number of answer sets in $PL_j$, $Q_j \in S_j$ denotes a specific question $Q_j$ (e.g., "*Who starred Interstellar?*") belongs to $S_j$ (e.g., "*Who starred [NE]?*") and $Pr(P_i, Q_j|\theta)$ is the probability that $P_i \in PC(Q_j, v_i)$ for any $v_i \in A_{Q_j}$, which is defined as follows, where $\mathbb{1}()$ is the indicator function.

$$Pr(P_i, Q_j|\theta) = \frac{\sum_{v_i \in A_{Q_j}} |PC(Q_j, v_i)|^{\mathbb{1}(P_i \in PC(Q_j, v_i))}}{|A_{Q_j}|} \quad (4)$$

After calculating the probability for all the potential paths, we choose the path with the highest probability as the answer. If multiple paths have the highest probability, we treat all of them as correct answers. When training the question encoder and decoder, we use binary cross-entropy loss which is defined as:

$$\mathcal{L}(\hat{P}, P) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{|P|} \mathbb{1}(P[j] = r_i) log(Pr(r_i|\mathcal{M})) + (1 - \mathbb{1}(P[j] = r_i)) log(1 - Pr(r_i|\mathcal{M}))$$

where $\mathcal{M}$ is the parameters of the question encoder-decoder model, $N$ is the number of relations in $\mathcal{G}$. $P$ is the path obtained by Equation (4). $\hat{P}$ is the output of path decoder. Note that the output of path decoder at time step $t$ is a probability distribution over all relations in $\mathcal{G}$.

## 3.3 Answer Scoring

When answering multi-hop natural language questions, traditional knowledge graph traversal or subgraph matching method is infeasible due to the incompleteness of the knowledge graph. When using learning-based methods to solve this problem, we aim to find a function $f()$ which takes in the topic entity $v_Q$, the path $P = (r_1, r_2, ..., r_{|P|})$, the knowledge graph $\mathcal{G}$ and a candidate entity $v$ to output a score which is used to denote how likely it is to travel from $v_Q$, according to path $P$, to reach $v$.

**A - Background.** If an algorithm satisfies the transitivity property of knowledge graph, the operation on path $P$ can be expressed as

$$\mathbf{p} = \mathbf{r}_1 \star \mathbf{r}_2 \star \cdots \star \mathbf{r}_n \quad (5)$$

where $\star$ is a composition operation and $\mathbf{p}$ is the embedding of $P$. According to different designs, the composition operation can have different forms. For example, in TransE [2], each relation represents a linear translation operation in the embedding space, so the path representation can be obtained by:

$$\mathbf{p} = \sum_{r_i \in P} \mathbf{r}_i$$

In RotaE [21], each relation represents a rotation in the complex space, and the composition operation is the Hadamard (i.e., element-wise) product which means

$$\mathbf{p} = \mathbf{r}_1 \odot \mathbf{r}_2 \odot \cdots \odot \mathbf{r}_n$$

However, naively using Equation (5) to calculate the path embedding and finding answers may suffer from low accuracy. This is because noise often exists in the embedding space, and with the increase of the path length, the cascading error will become larger [10] [13]. So, it is necessary to use the intermediate candidates to adjust the search process.

**B - Probabilistic Reasoning Model.** We address this issue by using a probability model $\Theta$. Considering a relation sequence $P = (r_1, ..., r_{|P|})$ originated from topic entity $v_Q$, the model predicts the likelihood $\Theta : (r, v) \rightarrow [0, 1]$ of following a certain edge in a relation sequence from $v_Q$ to any node in $A_Q$. Specifically, we

compute the likelihood of $v$ by multiplying the likelihood of all intermediate steps traversed by $P$ in $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$ by:

$$Pr(v|P, v_Q, \mathcal{G}) \propto \prod_{i=1}^{|P|} \Theta(r_i, v_i|P_{1\to i-1}, v_Q, \mathcal{G}) \tag{6}$$

where $P_{1\to r_i}$ is the subsequence of $P$ up to the $i$-th relation and $P_{1\to 0}$ means empty set. Finding the best answer is equivalent to maximizing the probability function. A naive way to find answers is by choosing the intermediate entity $v_i$ which can maximize the probability at each time step. However, due to the incompleteness of the knowledge graph, it may fail to find the correct answer. Iterating all the intermediate candidates can make sure to find the correct answer with a high probability, but it could hamper the efficiency. In order to strike a good balance between effectiveness and efficiency, as well as mitigate the cascading error [10], at each step, we select the *top-k* candidates with maximum likelihood $Pr(v|P_{1\to i-1}, v_Q, \mathcal{G})$. This can be done by an efficient search algorithm, such as beam search starting from $v_Q$. In the last step, we choose the candidate with the highest probability.

$$o = \max_{v_i \in \mathcal{V}}(Pr(v_i|P, v_Q, \mathcal{G}))$$

Note that, the goal of probability model $\Theta$ is to estimate how likely an entity is the answer. It is used by both the KGQA task and the KGC task to allow them automatically share information. Many methods could be used to model $\Theta$, e.g., bilinear transformation [25], convolutional networks [26] and so on. In our experiment, we use ComplEx [22] for its simplicity. Given $v_h, v_t \in \mathcal{V}$ and $r_i \in \mathcal{R}$, and their embedding $\mathbf{e}_h, \mathbf{e}_i, \mathbf{r}_i$, the probability score of $v_t$ that is reachable from $v_h$ by $r_i$ is calculated by:

$$Pr(v_t|r_i, v_h, \mathcal{G}) = Re(< \mathbf{r}_i, \mathbf{e}_h, \bar{\mathbf{e}}_i >) \tag{7}$$

where $Re()$ is the real part of ComplEx [22] output.

**C - Connection with Existing Methods.** Most existing algorithms which satisfy the transitivity property of knowledge graph are special cases of our model. For example, in TransE [2], each relation represents a linear translation operation in the embedding space, so the probability score of $(r_i, v_t)$ can be calculated by:

$$\Theta(r_i, v_t|P_{1\to i-1}, v_Q, \mathcal{G}) \propto ||\mathbf{e}_{v_Q} + \sum_{r_j \in P_{1\to i}} \mathbf{r}_j - \mathbf{e}_t||_2^{-\mathbb{1}(i=|P|)}$$

and similarly in RotatE [21], the probability score of $(r_i, v_t)$ can be calculated by:

$$\Theta(r_i, v_t|P_{1\to i-1}, v_Q, \mathcal{G}) \propto ||\mathbf{e}_{v_Q} \odot \mathbf{r}_1 \odot ... \odot \mathbf{r}_i - \mathbf{e}_t||_2^{-\mathbb{1}(i=|P|)}$$

## 3.4 Answer Refinement

In most cases, the set of candidate answers found by model $\Theta$ may already be a reasonable estimate of $A_Q$. Sometimes, however, noise may exist in the candidate set and has a higher probability than true answers. One possible reason is that relations in KG exhibit various types of patterns and properties. For example, relations like `FatherOf` and `liveIn` are asymmetric, but relations like `IsFriendWith` are symmetric property. Ideally, a good model should be able to learn all combinations of different properties, like symmetry, antisymmetry, and transitivity. However, due to the incompleteness and complexity of the knowledge graph, it is almost impossible to find a perfect model which can satisfy all the properties and find answers without errors. If we could refine the results, the accuracy of the model may be further increased.

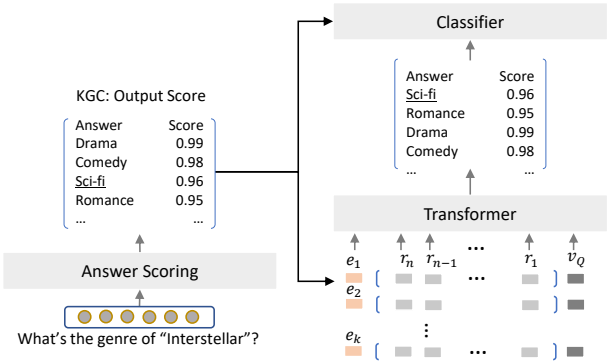



**Figure 3: Answer Refinement.**

Based on this observation, we propose an answer refinement model to re-order the *top-k* candidates of the answer scoring module. Given the *top-k* candidates which have the highest scores found by the answer scoring module, we want to re-select the high quality entities from them. Inspired by [23] [9], given a topic entity $v_Q$ and a path $P$, we concatenate them with each of the candidates to get $k$ sequences. We treat each sequence as a language sentence and pass it through a Transformer [23] to predict the soundness of this sentence.

$$h_i = \text{TRANSFORMER}([\mathbf{e}_{v_Q}|\mathbf{r}_1|...|\mathbf{r}_n|\mathbf{e}_{v_i}])$$

where $h_i$ is the output embedding of entity $v_i$. The final score is predicted by passing $h_i$ through a feed forward neural network with sigmoid function:

$$Pr(v_i|P, v_Q, \mathcal{G}) = \text{Sigmoid}(\text{FFN}(h_i))$$

In the last step, a classifier will be used to return the answer predicted by the answer scoring module or the transformer module. The architecture is shown in Figure 3. Note that the Refinement step is optional.

## 3.5 Learning Algorithm

The overall loss function of the proposed BiNet is as follows:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{KGQA} + \mathcal{L}_{KGC} + \mathcal{L}_{Path} + \mathcal{L}_{REG} \\ &= \sum_{Q \in \overline{Q}} \mathcal{J}(\hat{y}, y) + \lambda_1 \sum_{(h,r,t) \in \mathcal{G}} \mathcal{J}(\hat{t}, t) + \lambda_2 \sum_{Q \in \overline{Q}} \mathcal{L}(\hat{P}, P) + \lambda_3 ||\mathbf{W}||_2^2\end{aligned}$$

where $\lambda_1$, $\lambda_2$ and $\lambda_3$ are hyper parameters used to balance the loss. $\overline{Q}$ is the KGQA training set and $(h, r, t)$ belongs to KGC training set. $\mathcal{L}_{KGQA}$ is the KGQA loss, and $\mathcal{L}_{KGC}$ is the KGC loss. Both of them are calculated by $\mathcal{J}$ which is binary cross entropy loss. The first term $\mathcal{J}(\hat{y}, y)$ measures the loss of the KGQA task, $\hat{y}$ is the answer predicted by BiNet and $y$ is the ground truth. The second term $\mathcal{J}(\hat{t}, t)$ calculates the loss in the KGC task, $\hat{t}$ is the answer predicted by BiNet and $t$ is the ground truth. The third item $||\mathbf{W}||_2^2$ is the regularization term for preventing overfitting. The last term $\mathcal{L}(\hat{P}, P)$ is the path decoder loss. Note that the KGQA task $\mathcal{L}_{KGQA}$ contains two parts. The first part is the loss of answer scoring model

before answer refinement. The second part is the loss of answer scoring model after answer refinement.

### 3.6 Proof and Analysis

In this section, we analyze the expressive power of our model. In particular, we show in Lemma 1 and Lemma 2 that KGC and KGQA can indeed mutually benefit each other. We further show in Lemma 3 that if the exact matching of a path exists in the knowledge graph and Equation (7) is equal to 1 for any $(h, r, t) \in \mathcal{L}$, the proposed BiNet is guaranteed to find it.

Lemma 1. *(Benefit of KGQA for KGC) Given a natural language question $Q = (w_1, w_2, ..., w_{|Q|})$ with topic entity $v_Q$ and answer set $A_Q$. Path $P = (r_1, r_2, ..., r_{|P|})$ is a relation sequence of Q. If there is no exact matching between $v_Q$ and any $v \in A_Q$ in the knowledge graph $\mathcal{G}$, suppose $< v_k, r_j, v_m >$ is the missing link in the knowledge graph which is on the path from $v_Q$ to a node $v_a \in A_Q$, minimizing KGQA loss $\mathcal{L}_{KGQA}$ will also minimize KGC loss $\mathcal{L}_{KGC}$.*

Proof. Given an multi-hop path $P = (r_1, r_2, ..., r_{|P|})$, minimizing $\mathcal{L}_{KGQA}$ means maximizing $Pr(v|P, v_Q, \mathcal{G})$ for any $v \in A_Q$. Note that $< v_k, r_j, v_m >$ is the missing link on the path from $v_Q$ to a node $v_a \in A_Q$. Then, maximizing $Pr(v_a|P, v_Q, \mathcal{G})$ means maximizing $Pr(v_m|r_j, v_k, \mathcal{G})$ which is the output of the KGC task on triple $(v_k, r_j, v_m)$. Therefore, we have that $\mathcal{L}_{KGC}$ is minimized. □

Lemma 2. *(Benefit of KGC for KGQA) Given a natural language question $Q = (w_1, w_2, ..., w_{|Q|})$ with topic entity $v_Q$ and answer set $A_Q$. Path $P = (r_1, r_2, ..., r_{|P|})$ is a relation sequence of Q. If there is no exact matching between $v_Q$ and any $v \in A_Q$ in the knowledge graph G, minimizing KGC loss $\mathcal{L}_{KGC}$ will also minimize KGQA loss $\mathcal{L}_{KGQA}$.*

Proof. Minimizing $\mathcal{L}_{KGC}$ means maximizing $Pr(v_t|r_i, v_h, \mathcal{G})$ for any $(v_h, r_i, v_t) \in \mathcal{G}$. Given a path $P = (r_1, r_2, ..., r_{|P|})$ and a topic entity $v_Q$, for any $v_a \in A_Q$, the probability $Pr(v_a|P, v_Q, \mathcal{G})$ can be expressed as Equation 6. Since each item in this formula is maximized, $Pr(v_a|P, v_Q, \mathcal{G})$ is also maximized. Therefore, $\mathcal{L}_{KGQA}$ is minimized. □

If the background knowledge graph is complete, traditional path traverse or subgraph matching methods could find exact matches from knowledge graph $\mathcal{G}$, where each exact match is a subgraph in $\mathcal{G}$ which could be mapped to multi-hop query $(v_Q, P, ?)$ exactly. Ideally, if the answer scoring model in BiNet could output 1 for any triple $(h, r, t)$ in $\mathcal{G}$, BiNet could also find exact matches like subgraph matching methods.

Lemma 3. *(Model Soundness) Given a natural language question $Q = (w_1, w_2, ..., w_{|Q|})$ with topic entity $v_Q$ and answer set $A_Q$. Path $P = (r_1, r_2, ..., r_{|P|})$ is a relation sequence of Q. If Equation (7) is equal to 1 for any $(h, r, t) \in \mathcal{L}$, and if there is an exact match between $v_Q$ and any $v \in A_Q$ in the knowledge graph $\mathcal{G}$, the proposed BiNet is guaranteed to find it as long as $k \geq D^{|P|-1}$, where D is the maximum out-degree of a node on a specific relation type in the knowledge graph.*

Proof. Because $D$ is the maximum out-degree of a node on a specific relation type in the knowledge graph, there are at most $D^{|P|-1}$ exact matching paths of $(r_1, r_2, ..., r_{|P|-1})$ exist in the knowledge graph started from topic entity $v_Q$. If $k \geq D^{|P|-1}$, they will be all included in the *top-k* candidates. Therefore, the exact match can be found. □

Note that $|P|$ is a relatively small value (e.g., usually $|P| \leq 3$).

## 4 EXPERIMENTS

In this section, we evaluate the performance of the proposed BiNet on several public datasets. We first introduce the datasets and baselines used in the paper, and then present the experiment results.

### 4.1 Experimental Setting

Three datasets are used in the paper which are listed below:

- **MetaQA** is a multi-hop question dataset on movie domain which contains more than 400K natural language questions. The background knowledge graph contains more than 100K triples which includes different relationships among directors, movies, genres and actors. All questions can be divided into three categories: 1-hop, 2-hop and 3-hop.
- **WebQuestionsSP** contains about 4,000 questions which could be answered by Freebase. It is a mixture of 1-hop questions and 2-hop questions.
- **SimpleQuestions** [3] is a dataset which consists of more than 100K simple 1-hop natural language questions and their corresponding triples from Freebase. In this paper, we use a subset of SimpleQuestions which contains all the questions that can be answered by Freebase used in WebQuestionsSP.

The statistics of these datasets are shown in Table 8. Their corresponding knowledge graphs are shown in Table 9. In the experiment, we compare the proposed BiNet with baselines in the challenging settings with incomplete KG with 50% and 70% missing edges (we randomly delete 50% and 70% edges from the full knowledge graph). This is because KGQA on the complete KG becomes trivial on these datasets. For example, simply using path traverse or subgraph matching could achieve nearly 100% accuracy, the results of which are shown in Table 8.

We compare our method BiNet with 4 baselines on the KGQA task, including:

- GraftNet [20] finds a question-specific subgraph containing KG facts, and then uses a graph neural network to predict the answers.
- PullNet [19] utilizes the shortest path as supervision to train graph retrieval module and conduct multi-hop reasoning with GraftNet on the retrieved sub-graph.
- Key-Value Memory Network (KVMem) [15] maintains a memory table which stores KG facts and uses this for retrieval.
- EmbedKGQA [18] conducts multi-hop reasoning through matching pre-trained entity embeddings with question embedding obtained from RoBERTa.

We compare our method BiNet with 4 baselines on the KGC task, including

[3] The query paths of WebQuestionsSP and SimpleQuestions are given, so we don't need to use Question Encoder-Decoder.

**Table 2: KGQA Hits@1 results of MetaQA on 50% and 30% incomplete knowledge graphs.**

| | 50% KG | | | | 30% KG | | | |
|---|---|---|---|---|---|---|---|---|
| Model | MetaQA-1 | MetaQA-2 | MetaQA-3 | Avg | MetaQA-1 | MetaQA-2 | MetaQA-3 | Avg |
| GraftNet | 64.0 | 52.6 | 59.2 | 58.6 | 48.4 | | | 48.4 |
| PullNet | 65.1 | 52.1 | 59.7 | 59.0 | - | - | - | - |
| KV-Mem | 63.6 | 41.8 | 37.6 | 47.7 | 44.7 | | | 44.7 |
| EmbedKGQA | 83.1 | 91.8 | 70.3 | 81.7 | 77.7 | 81.2 | 69.0 | 76.0 |
| BiNet | 84.2 | 92.8 | 75.9 | 84.3 | 77.8 | 86.4 | 74.3 | 79.5 |

**Table 3: KGQA Hits@1 results of WQSP and SimpleQA on 50% and 30% incomplete knowledge graphs.**

| | 50% KG | | 30% KG | |
|---|---|---|---|---|
| Model | Webqsp | SimpleQA | Webqsp | SimpleQA |
| GraftNet | 32.7 | 39.8 | 34.9 | 25.7 |
| PullNet | 48.2 | - | 34.6 | - |
| KV-Mem | 50.1 | 28.9 | 25.8 | 22.8 |
| EmbedKGQA | 47.3 | 41.7 | 38.8 | 33.5 |
| BiNet | 49.4 | 42.6 | 40.5 | 33.9 |

- RESCAL [17] is a three-way tensor factorization method which embeds each entity as a latent vector and each relation as a matrix.
- DisatMult [25] uses low dimensional vectors to represent nodes and uses bilinear functions to represent relations.
- ComplEx [22] embeds each entity as a complex vector which contains a real part and an imaginary part, and the relations are represented as bilinear functions.

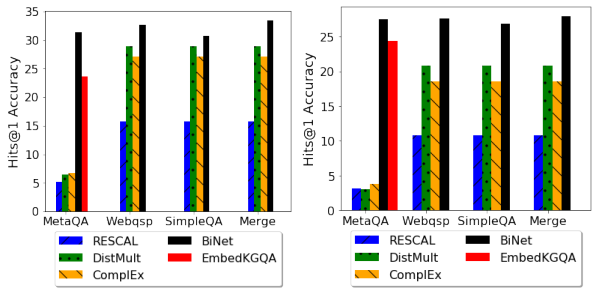
## 4.2 Knowledge Graph Question Answering Performance

Following the standard setup in KGQA [18], we evaluate the accuracy using the Hits@1 metrics. Table 2 shows the results of all baseline methods on the MetaQA dataset with 50% incomplete KG and 30% incomplete KG, respectively. The code of PullNet is not publicly available, so we omit its performance on 30% incomplete knowledge graph. The results of GraftNet and KV-Mem are from [20]. As we can see from the table, when the background knowledge graph becomes sparse, the Hits@1 accuracy of all methods decreases. This means that the quality of the background knowledge graph has a significant impact on the KGQA task. For subgraph retrieval-based methods: GraftNet and PullNet, their performances suffer severely from the incompleteness of the background knowledge graph. This is because, when the KG becomes sparse, the generated subgraphs of these methods are unable to cover the answer entities. Among all the methods, our method achieves the best results. For 50% incomplete KG, our BiNet is about 2.5% better than EmbedKGQA and more than 25% better than all other baseline methods. For 30% incomplete KG, our BiNet is about 3.5% better than EmbedKGQA and 28% better than other baseline methods on average.

Table 3 shows the performance of different methods on WebQuestionsSP and SimpleQuestions datasets. We have the similar results. When the background knowledge graph becomes sparse, the accuracy of all methods decreases. Nonetheless, the proposed BiNet consistently outperforms other baseline methods, and it is about 1.2% better than baseline methods on average.

## 4.3 Knowledge Graph Completion Performance

Figure 4(a) and Figure 4(b) show the accuracy of different knowledge graph completion methods on 50% incomplete knowledge graph and 30% incomplete knowledge graph, respectively. Traditional knowledge graph embedding methods like RESCAL, DistMult and ComplEx do not perform very well on MetaQA knowledge graph. This is because the knowledge graph is very sparse. It only contains about 66,791 edges which is 2.3% that of Freebase. Since EmbedKGQA is not designed for knowledge graph completion, when using EmbedKGQA for knowledge graph completion, we first transform the KG triple $(h, r, v)$ to a natural language question, and then train EmbedKGQA. As we can see, for MetaQA knowledge graph, EmbedKGQA performs quite well. It is higher than other existing knowledge graph completion baseline methods without the question data. When using the question date, EmbedKGQA is about 15% higher than other baselines. Compare with other methods, the proposed BiNet consistently has the highest KGC performance.




(a) Accuracy on 50% Incomplete KG (b) Accuracy on 30% Incomplete KG

**Figure 4: Knowledge Graph Completion Accuracy. 'Merge' dataset is the combination of Webqsp and SimpleQA.**

## 4.4 Ablation Studies

In this section, we evaluate the effectiveness of each component of the proposed BiNet.

**A - Answer Refinement.** In this subsection, we show the effectiveness of the answer refinement module. The results are shown in Table 4. As we can see, the refinement module could improve the prediction accuracy by about 2% on average on both 50% incomplete and 30% incomplete knowledge graphs. This means that the proposed refinement model indeed alleviates the sparsity of the background knowledge graph. Compared with 1-hop natural language questions, e.g., Webqsp and SimpleQA, the accuracy improvement on long path questions is more significant (3.2% vs 2%). This means that when the path becomes longer, the refinement module is even more effective.

**Table 4: Ablation study of Answer Refinement.**

| 50% KG | | | |
|---|---|---|---|
| Model | MetaQA-3hop | Webqsp | SimpleQA |
| BiNet without refinement | 70.3 | 47.2 | 41.8 |
| BiNet with refinement | 75.9 | 49.4 | 42.6 |
| 30% KG | | | |
| Model | MetaQA-3hop | Webqsp | SimpleQA |
| BiNet without refinement | 71.2 | 39.1 | 33.2 |
| BiNet with refinement | 74.3 | 40.5 | 33.9 |

**B - The Power of Knowledge Graph Completion.** In this subsection, we study whether simply completing the knowledge graph first could help KGQA task. When completing the knowledge graph, we use two strategies. The first one is the heuristic based method to complete the knowledge graph according to the natural language questions in the KGQA training and validation sets. This heuristic based method is similar to some rule based knowledge graph completion methods. For example, if `A` is the child of `B` and `C` is the husband of `B`, then `A` is the child of `C`. After we complete the knowledge graph by the heuristic method, we further use the second strategy to complete the knowledge graph. We train ComplEx on the partially completed knowledge graph. Then given a triple $(h, r, ?)$, we use ComplEx to predict the answer. Note that we only keep those triples which satisfy $Pr(v_t|r_i, v_h, \mathcal{G}) >= 0.99$ where 1 is the highest score. We also prune some obvious wrong predictions from the triple list. For example, if the head entity is a movie `Interstellar` and the relation is `starredBy`, but the prediction is not a movie star, we delete this triple even though its probability is greater than or equal to 0.99.

**Table 5: The power of knowledge graph completion.**

| 50% KG | | | |
|---|---|---|---|
| Model | EmbedKGQA | KGC + EmbedKGQA | BiNet |
| MetaQA-1hop | 83.1 | 83.2 | 84.2 |
| MetaQA-2hop | 91.8 | 92.4 | 92.8 |
| MetaQA-3hop | 70.3 | 73.5 | 75.9 |
| Webqsp | 47.3 | 47.7 | 49.4 |
| SimpleQA | 41.7 | 41.9 | 42.6 |
| 30% KG | | | |
| Model | EmbedKGQA | KGC + EmbedKGQA | BiNet |
| MetaQA-1hop | 77.7 | 77.8 | 77.8 |
| MetaQA-2hop | 81.2 | 85.1 | 86.4 |
| MetaQA-3hop | 69.0 | 71.1 | 74.3 |
| Webqsp | 38.8 | 39.1 | 40.5 |
| SimpleQA | 33.5 | 33.7 | 33.9 |

Table 5 shows the performance of KGC+EmbedKGQA and the proposed BiNet. The performance of KGC+EmbedKGQA is better than EmbedKGQA on all three kinds of questions. This means that completing the knowledge graph first can indeed improve the KGQA performance. In MetaQA-2hop questions on 30% incomplete KG, it achieves the largest accuracy improvement which is 3.9%. On average, completing the knowledge graph first could improve about 1.2% Hits@1 accuracy. The proposed BiNet further improves the performance by 1.3%.

**C - Path Prediction.** A good path decoder is important for the proposed model BiNet. Before training the path decoder, we manually add ground-truth paths to the training data. When training the model, at each time step the path decoder will predict what the next relation is according to the previous relation decoded. With probability equal to the $\alpha$ ($\alpha = 0.5$), the decoder will use the actual ground-truth relation as the input to the decoder during the next time-step. However, with probability $1 - \alpha$, it will use the relation that the model predicts as the next input to the model, even if it does not match the actual next relation in the ground-truth. Table 11 shows some results of the path decoder. As we can see, the path decoder could generate relation paths with high accuracy.

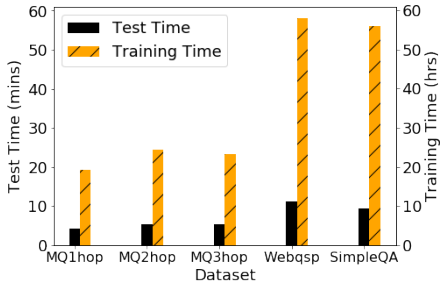



**Figure 5: BiNet Training and Test Time.**

**D - Efficiency.** Figure 5 show the training time and test time of BiNet on different datasets. As we can see, the runtime of BiNet on Webqsp and SimpleQA is much larger than that on MetaQA dataset. This is because the background knowledge graph of Webqsp and SimpleQA is much larger than that of MetaQA. Despite the long training time, the test time of BiNet is relatively short which is less than 15 minutes.

## 5 RELATED WORK

In this section, we review related work for multi-hop knowledge graph question answering, knowledge graph completion, and multi-task learning.

### 5.1 Multi-hop Knowledge Graph Question Answering

Multi-hop knowledge graph question answering aims to answer the question which could be transformed to a relation path in the knowledge graph. Existing methods could be divided into several different categories, e.g, semantic parsing based method, information retrieval based method, embedding based method and so on. For example, GraftNet [20] and PullNet [19] are information retrieval-based methods that retrieve a subgraph of candidate answers from the knowledge base to guide prediction. KV-Mem [15] and EmbedKGQA [18] are embedding and deep learning-based methods that use deep learning networks to embed the question into a point in the embedding space and find answers according to a similarity function. For a comprehensive survey on KGQA, see a recent survey [28]. In this paper, we focus on using deep learning model to answer multi-hop natural language questions.

### 5.2 Knowledge Graph Completion

knowledge graph completion aims to predict missing links or entities based on existing information in the knowledge graph. Most of the existing methods like TransE [2], RESCAL [17] and DistMult [25] embed entities as points in the low dimensional Euclidean and model relations as linear or bilinear transformation in the space. Other methods like RotatE [21] and ComplEx [22] represent entities as points in the complex space and relations as rotation or Bilinear

transfromation. Other methods like BoxE [1] and KG2E [7] use geometry box or gaussian distribution to represent an entity.

### 5.3 Multi-task Learning

Multi-task learning aims to learn a model which could simultaneously learn knowledge from several different tasks. Such approaches could offer advantages like improving model accuracy, preventing over-fitting through shared representations and fast learning. Most existing methods for multi-task learning focus on designing competitive models which enable sharing knowledge among different tasks [6, 16, 27]. For example, in [16], the authors propose Cross-stitch units which allow two deep neural networks to share representations as a linear combination of input activation maps. In [27], the authors propose a model which shares the adjacency matrix between the network alignment task and network completion task. Other methods, like [11] aims to study how to weigh each task in multi-task learning to prevent tuning different weights by hand, while [3] aims to make training deep multitask models easier, and proposes an algorithm that automatically balances training in deep multitask models by dynamically tuning gradient magnitudes. In this paper, we build a new multi-task model for joint KGQA and KGC.

## 6 CONCLUSION

In this paper, we propose to use multi-task learning to solve multi-hop question answering on knowledge graph and knowledge graph completion at the same time. We propose a neural network-based model named BiNet to accomplish this. By allowing the KGQA and KGC to use the same embedding space and a shared answer scoring module, both tasks could learn latent features from each other in a mutually beneficial way. The experiment results show that the proposed BiNet consistently outperforms the state-of-the-art methods on both KGC and KGQA on multiple datasets.

## 7 ACKNOWLEDGEMENT

LL and HT are supported by National Science Foundation under grant No. 1947135, and 2134079 by the NSF Program on Fairness in AI in collaboration with Amazon under award No. 1939725, by DARPA HR001121C0165, INCAS by Agriculture and Food Research Initiative (AFRI) grant no. 2020-67021-32799/project accession no.1024178 from the USDA National Institute of Food and Agriculture, The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## REFERENCES

[1] Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. 2020. BoxE: A Box Embedding Model for Knowledge Base Completion. arXiv:2007.06267 [cs.AI]

[2] A Bordes and Usunier N. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*.

[3] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. arXiv:1711.02257 [cs.CV]

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[5] Boxin Du, Lihui Liu, and Hanghang Tong. 2021. Sylvester Tensor Equation for Multi-Way Association. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 311–321.

[6] Boxin Du, Changhe Yuan, Robert Barton, Tal Neiman, and Hanghang Tong. 2021. Hypergraph Pre-training with Graph Neural Networks. *arXiv preprint arXiv:2105.10862* (2021).

[7] S He, K Liu, G Ji, and J Zhao. 2015. Learning to Represent Knowledge Graphs with Gaussian Embedding *(CIKM '15)*.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (nov 1997), 1735–1780.

[9] Weiqiang Jin, Hang Yu, Xi Tao, and Ruiping Yin. 2022. Improving Embedded Knowledge Graph Multi-hop Question Answering by introducing Relational Chain Reasoning. arXiv:2110.12679 [cs.CL]

[10] P Liang K Guu, J Miller. [n.d.]. Traversing Knowledge Graphs in Vector Space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

[11] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. arXiv:1705.07115 [cs.CV]

[12] Lihui Liu, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2021. KompaRe: A Knowledge Graph Comparative Reasoning System. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 3308–3318.

[13] Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang Tong. 2021. Neural-Answering Logical Queries on Knowledge Graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1087–1097.

[14] L. Liu, B. Du, J. xu, and H. Tong. 2019. G-Finder: Approximate Attributed Subgraph Matching. In *2019 IEEE International Conference on Big Data (Big Data)*. 513–522.

[15] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. arXiv:1606.03126 [cs.CL]

[16] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-Stitch Networks for Multi-Task Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[17] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* (Bellevue, Washington, USA) *(ICML'11)*. Omnipress, Madison, WI, USA, 809–816.

[18] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 4498–4507.

[19] Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. 2019. PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text. arXiv:1904.09537 [cs.CL]

[20] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. arXiv:1809.00782 [cs.CL]

[21] Z Sun, Z Deng, J Nie, and J Tang. [n.d.]. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. *CoRR* ([n. d.]).

[22] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

[24] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic Knowledge Graph Alignment. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 5 (May 2021), 4564–4572.

[25] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. arXiv:1412.6575 [cs.CL]

[26] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. arXiv:1412.6575 [cs.CL]

[27] Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. 2017. iNEAT: Incomplete Network Alignment. In *2017 IEEE International Conference on Data Mining (ICDM)*. 1189–1194. https://doi.org/10.1109/ICDM.2017.160

[28] Xiaohan Zou. 2020. A Survey on Application of Knowledge Graph. *Journal of Physics: Conference Series* 1487, 1 (mar 2020), 012016. https://doi.org/10.1088/1742-6596/1487/1/012016

## SUPPLEMENTARY MATERIAL: REPRODUCIBILITY

**Reproducibility.** All experiments are performed on a machine with an Intel(R) Xeon(R) Gold 6240R CPU, 1510GB memory and NVIDIA-SMI Tesla V100-SXM2. The details of datasets, machine and parameters can be found in Section 4. All datasets are publicly available. The source code of this paper can be found at https://github.com/lihuiliullh/BiNet.

**Baselines.**The implementation code of GraftNet, Key-Value Memory Network and EmbedKGQA can be found from the Table below. Because the source code of PullNet is not publicly available. We obtain its results from [20].

**Table 6: Baseline Source.**

| Model | github |
|---|---|
| GraftNet | https://github.com/haitian-sun/GraftNet |
| KV-Mem | https://github.com/jojonki/key-value-memory-networks |
| EmbedKGQA | https://github.com/malllabiisc/EmbedKGQA |

For the knowledge graph completion baselines, we use LibKGE https://github.com/uma-pi1/kge. The training and testing configure file can be found from the repository.

**Dataset.**All the datasets used in this paper are publicly available which could be found in the Table below.

**Table 7: Dataset Source.**

| Model | github |
|---|---|
| MetaQA | https://github.com/malllabiisc/EmbedKGQA |
| Webqsp | https://github.com/malllabiisc/EmbedKGQA |
| SimpleQA | https://github.com/davidgolub/SimpleQA/tree/master/datasets |

Table 8 and Table 9 show the details of the datasets. Table 10 shows the background knowledge graph used in the experiment which contains 70% missing edges. The number of entities, number of relations and number of test edges are the same as 50% incomplete knowledge graph. However, the number of train edges is much smaller.

**Training Parameters.** When training BiNet on MetaQA, we use embedding dimension 200 and batch size 128. The learning rate is set to 0.0005. For datasets Webqsp and SimpleQA, we set embedding dimension to 200 and batch size to 16. The learning rate is set to 0.00002.

**Table 8: Summary of datasets. Coverage is the accuracy of subgraph matching. As we can see, simply applying edge traverse on the complete knowledge graph could achieve nearly 100% accuracy.**

| Dataset | Train | Valid | Test | Coverage |
|---|---|---|---|---|
| MetaQA 1-hop | 96,106 | 9,992 | 9,947 | 100% |
| MetaQA 2-hop | 118,948 | 14,872 | 14,872 | 100% |
| MetaQA 3-hop | 114,196 | 14,274 | 14,274 | 99% |
| WebQSP | 2,998 | 100 | 1,639 | 99% |
| SimpleQA | 15,3188 | 2,105 | 4,345 | 99% |

**Table 9: Statistics of the 50% KG for the three datasets. Note that WebQSP and SimpleQA use the same background knowledge graph.**

| Dataset | Entities | Relations | Train Edges | Test Edges |
|---|---|---|---|---|
| MetaQA | 43,234 | 18 | 66,791 | 4,000 |
| WebQSP | 1,886,683 | 1,144 | 2,872,880 | 20,000 |
| SimpleQA | 1,886,683 | 1,144 | 2,872,880 | 20,000 |

**Table 10: Statistics of the 30% KG for the three datasets. Note that WebQSP and SimpleQA use the same background knowledge graph.**

| Dataset | Entities | Relations | Train Edges | Test Edges |
|---|---|---|---|---|
| MetaQA | 43,234 | 18 | 40,074 | 4,000 |
| WebQSP | 1,886,683 | 1,144 | 1,764,663 | 20,000 |
| SimpleQA | 1,886,683 | 1,144 | 1,764,663 | 20,000 |

**The Pseudocodes.** The pseudocodes of BiNet are listed below.

**Algorithm 1** Knowledge Graph Question Answering Training

**Input:** knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$, training dataset $\{Q, A_Q\}_{train}$, hyper parameter $k$
**Training:**
Obtain question embedding **Q** from pre-trained BERT
Calculate scores for all entities by Answer Scoring Module
Select *top-k* candidates $S$
Decode question embedding **Q** to obtain path $P$
**for** each $v_i \in S$ **do**
    Concatenate topic entity $v_Q$, path $P$ and $v_i$
    Use Transformer to calculate the score of $v_i$
**end for**
**Return** sorted *top-k* candidates

**Algorithm 2** Knowledge Graph Completion Training

**Input:** knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$, training datasets $\{v_Q, P, A_Q\}_{train}$, $\{(h, r, t)\}_{train}$ for KGC, hyper parameter $k$
**Training:**
Pre-train shared entity and relation embedding by $\{(h, r, t)\}_{train}$
Head = topic entity $v_Q$
**for** each $r_i \in P$ **do**
    Beam search according to Head and $r_i$
    Head = *top-k* candidates
**end for**
**Return** the best sorted candidate

**The Results of Path Decoder.** Table 11 shows the results of the path decoder.

**Table 11: Results of Path Decoder.**

| Question | Path |
|---|---|
| the movies starred by [Tanner Maguire] were in which genres | starred_actors_reverse \| has_genre |
| when did the movies written by [Cristian Nemescu] release | written_by_reverse \| release_year |
| the films acted by [Benjamin Pitts] were released in which years | Starred_actors_reverse \| release_year |
| who are movie co-writers of [Ray Ashley] | written_by_reverse \| written_by |
| who co-starred with [Mary McDonnell] | starred_actors_reverse \| starred_actors |
| who are movie co-directors of [Jack Hazan] | directed_by_reverse \| directed_by |