Knowledge Graph Question Answering with Ambiguous Query

Lihui Liu
lihuil2@illinois.edu
University of Illinois at Urbana champaign
Urbana, Illinois, USA

Yuzhong Chen
yuzchen@visa.edu
Visa Research
USA

Mahashweta Das
mahdas@visa.edu
Visa Research
USA

Hao Yang
haoyang@visa.edu
Visa Research
USA

Hanghang Tong
htong@illinois.edu
University of Illinois at Urbana champaign
Urbana, Illinois, USA

ABSTRACT

Knowledge graph question answering aims to identify answers of the query according to the facts in the knowledge graph. In the vast majority of the existing works, the input queries are considered perfect and can precisely express the user's query intention. However, in reality, input queries might be ambiguous and elusive which only contain a limited amount of information. Directly answering these ambiguous queries may yield unwanted answers and deteriorate user experience. In this paper, we propose PReFNet which focuses on answering ambiguous queries with pseudo relevance feedback on knowledge graphs. In order to leverage the hidden (pseudo) relevance information existed in the results that are initially returned from a given query, PReFNet treats the top-k returned candidate answers as a set of most relevant answers, and uses variational Bayesian inference to infer user's query intention. To boost the quality of the inferred queries, a neighborhood embedding based VGAE model is used to prune inferior inferred queries. The inferred high quality queries will be returned to the users to help them search with ease. Moreover, all the high-quality candidate nodes will be re-ranked according to the inferred queries. The experiment results show that our proposed method can recommend high-quality query graphs to users and improve the question answering accuracy.

CCS CONCEPTS

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

KEYWORDS

Knowledge graph question answering

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583316

ACM Reference Format:
Lihui Liu, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Knowledge Graph Question Answering with Ambiguous Query . In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583316

1 INTRODUCTION

Knowledge graphs are ubiquitous, which have been used in a variety of applications, such as recommendation [6], alignment [27], fact checking [12] and many more. A knowledge graph is a graph data structure which contains a multitude of triples denoting real world facts. Each triple contains a head entity (e.g., `Harvard_University`), a tail entity (e.g., `United_States`) and a relation between them (e.g., `locatedIn`).

Knowledge graph question answering (short for KGQA) aims to find entities in the knowledge graph which can correctly answer the given question. This problem has attracted great attention from both academia and industry, and abundant algorithms have been proposed recently. For example, RnG-KBQA [28] adopts a rank-and-generate approach to transform the input natural language question to a query graph and finds the answer according to the query graph. This strategy of finding answers based on the query graph can also be referred to as subgraph matching [13]. Key-Value Memory Network (KVMem) [16] stores KG facts in a memory table and uses it to retrieve entities which have the most similar embedding as the input query. EmbedKGQA [20] embeds both the input question and entities in the knowledge graph to points in the embedding space and finds answers according to their embedding similarity.

Despite the great progress, most works focus on answering *defectless* queries on knowledge graphs. These queries are assumed to be perfect and can precisely express users' query intentions. However, this is not true most of the time in real cases for the following reasons. First, the vocabulary of different users can vary dramatically. According to a prominent study on the human vocabulary problem [8], about 80-90% of the times two persons will give different representations when they are asked to name the same concept [21]. This means the input queries of different users could be very different from each other. Second, some KGQA methods (e.g., [28] [4]) need to transform the natural language questions to graph queries, and then search the results according to these query graphs. The transformation algorithm may generate queries with

inaccurate graph structure. Last but not the least, allowing users to input query graphs directly may introduce additional structural noise or inaccuracy due to their lack of full background knowledge of the underlying KG [21].

To address these issues, query ambiguity and vagueness need to be correctly resolved, which in turn requires new information in addition to the query itself. Relevance feedback (short for ReF) is one promising solution. The general idea behind relevance feedback is to take the results that are initially returned from a given query, to gather user feedback, and to use information about whether or not those results are relevant to form a new query [1]. The user feedback can be explicit (e.g., clicking like or dislike), implicit (e.g., the duration of time spent viewing a document) or pseudo feedback (no feedback is given). In spite of the fact that relevance feedback has been studied extensively and proven to be effective in information retrieval, it has not been well studied in graph query system. Due to the unique characteristics of graph query, traditional relevance feedback methods are not directly applicable. How to use relevance feedback to improve query performance on graph data largely remains an open problem.

In this paper, we propose PReFNet which applies pseudo relevance feedback (short for PReF) to graph data to infer the true query structure from an ambiguous query and improve answering performance. More specifically, guided by Bayes rule, the proposed PReFNet decomposes the question answering problem into several components and models each component as a neural network. To infer the true query, PReFNet utilizes variational Bayesian inference. To boost the quality of the inferred query, a neighborhood embedding based VGAE model is used to prune inferior query. Finally, the newly inferred queries will be used to re-rank the original candidate answers. The experiment results show that our methods can achieve better question answering accuracy compared with the state-of-the-art baselines and better infer the user's query intention.

In summary, the main contributions of this paper are:

- **Problem Setting.** To our best knowledge, we are the first to study using neural network based pseudo relevance feedback on knowledge graphs to infer queries and improve question answering performance at the same time.
- **Algorithm** We propose to decompose the question answering problem into several components and model each component as a neural network model.
- **Empirical Evaluations**. The experimental results on several real-world datasets demonstrate the effectiveness of the proposed PReFNetframework.

The rest of the paper is organized as follows. Section 2 introduces notations used in this paper and gives the problem definition. Section 3 introduces the overall framework of the proposed algorithm and its technical details. The experiment results are presented in Section 4, and the related work is reviewed in Section 5. Finally, the paper is concluded in Section 6.

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. For example, (`New_York`, `locatedIn`, `United_States`) and (`London`, `locatedIn`, `United_Kingdom`) are two triples in knowledge graph Yago.

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

Table 1: Notations and definitions

Symbols	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$	the knowledge graph
$\mathcal{V}$	the entity set
$\mathcal{R}$	the relation set
$\mathcal{L}$	the fact set
v_i	the i^{th} entity/node in knowledge graph
r_i	the i^{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 ambiguous question
h_Q	the ambiguous question embedding
T	the true query graph of Q
$\overline{Q}$	the natural language question training set
A_Q	the answer set of question Q
a	a candidate answer of question Q
v_Q	the topic entity in question
z	the latent path
p_i	the i-th relation in latent path
w_i	the i^{th} word in Q
r_Q	the relation of query

Knowledge graph question answering aims to answer a question with the help of knowledge graphs. According to the study in [21], most users formulate queries using their own knowledge and vocabulary during the search process. They might not have a fairly good understanding of the underlying data schema and the knowledge graph structure. This means that the users' true intentions behind the queries may be frequently misinterpreted or misrepresented. For example, if a user inputs query "thomas jefferson role in declaration independence", the phrase "role in" should be interpreted as predicate "profession" or "occupation", and the phrase "declaration independence" refers to the pronouncement and founding document adopted by the Second Continental Congress. While in query "John Litel role in declaration independence", the phrase "role in" should be interpreted as predicate "film actor" or "act in", and the phrase "declaration independence" represents a movie. The same phrases "role in" and "declaration independence" have totally different interpretations in the two queries. Unless the transformation algorithm can accurately understand the user's intention, directly answering these ambiguous queries may yield unwanted answers and deteriorate user experience. Here, a promising way to disambiguate the query is via relevance feedback as follows. With the help of relevance feedback, we can infer from the top rank candidates (e.g., `White_House`, `Politician`, `President` for "thomas jefferson" and `United_States`, `Actor`, `Albany_Wisconsin` for "John Litel") that "thomas jefferson" is a statesman and "John Litel" is a actor. These candidates provide clear and useful interpretation to users. The new interpretation can not only help users understand the background knowledge graph better, but also return more accurate answers. More examples can be found in Subsection 4.5B.

In this paper, we focus on answering ambiguous one-hop question over knowledge graph. We assume the input ambiguous query Q contains a topic/anchor entity $v_Q \in \mathcal{V}$ and a sequence of words $Q = (w_1, w_2, ..., w_{|Q|})$. Ideally, each question can be mapped to a unique relation r_Q in the knowledge graph. The goal of question answering over knowledge graph is to identify a set of nodes $A_Q \subseteq \mathcal{V}$ which can answer the ambiguous question. We assume that all the answer entities exist in the knowledge graph, each question only contains a single topic/anchor entity $v_Q \in \mathcal{V}$ and v_Q is given.

To disambiguate the query and improve query accuracy, a natural idea is to effectively acquire more query-specific information. This can be achieved by relevance feedback which has been extensively studied in Information Retrieval Systems. In this paper, we focus on utilizing pseudo relevance feedback to infer true query intention and improve question answering accuracy. More specifically, the proposed PReFNet treats the top-k candidate answers of a KGQA system as most relevant information and uses them to derive the true query so that more accurate answers can be found. The derived queries will in turn help the users better formulate their queries and obtain better understanding of the background knowledge.

Formally, the problem this paper studies is defined as follows.

Problem Definition. *Answering Ambiguous Query:*

Given: *(1) A knowledge graph $\mathcal{G}$, (2) an ambiguous one-hop natural language question;*

Output: *(1) The answer of the question, (2) Top-*k *most likely correct query relations of the input query.*

3 PROPOSED METHOD

In this section, we first introduce the framework and basic idea behind our method. Then, the details of each specific component are elaborated.

3.1 Model Overview

The key idea of PReFNet is as follows. We treat the top-k results of the KGQA system as potentially correct answers, and use them to predict the true query relation. Given the ambiguous query and its anchor entity, we give the following lemma to decompose the problem of question answering over knowledge graph (KGQA).

Lemma 1. *(KGQA Decomposition) Given an ambiguous query Q and its anchor node v_Q, let $Pr(T|Q, v_Q)$ denote the probability that query relation T is generated from Q and let $Pr(a|T, v_Q)$ denote the probability that candidate answer a found by T is the true answer, we have*
$Pr(T|Q, v_Q)Pr(a|T, v_Q) \propto Pr(Q|T, v_Q)Pr(T|a, v_Q)Pr(a|v_Q)$.

Proof. The probability of inferring the true query relation can be expressed as the following equation according to Bayes rule:

$$\begin{aligned} Pr(T|Q, v_Q) &= \frac{Pr(Q, T|v_Q)}{Pr(Q|v_Q)} \\ &\propto Pr(Q, T|v_Q) \\ &= Pr(Q|T, v_Q)Pr(T|v_Q), \end{aligned} \tag{1}$$

where $Pr(Q|v_Q)$ can be ignored since it is irrelevant to inferring T. Furthermore, given a candidate answer a of ambiguous query Q, we have

$$Pr(a|T, v_Q) = \frac{Pr(a, T|v_Q)}{Pr(T|v_Q)},$$

This means

$$\begin{aligned} Pr(T|v_Q) &= \frac{Pr(a, T|v_Q)}{Pr(a|T, v_Q)} \\ &= Pr(a|v_Q)\frac{Pr(T|a, v_Q)}{Pr(a|T, v_Q)}. \end{aligned} \tag{2}$$

Plugging Eq. (2) to Eq. (1), we can get

$$Pr(T|Q, v_Q) \propto Pr(Q|T, v_Q)Pr(a|v_Q)\frac{Pr(T|a, v_Q)}{Pr(a|T, v_Q)}. \tag{3}$$

Consequently, we obtain

$$Pr(T|Q, v_Q)Pr(a|T, v_Q) \propto Pr(Q|T, v_Q)Pr(T|a, v_Q)Pr(a|v_Q). \tag{4}$$

□

In Eq. (4), $Pr(a|v_Q)$ (**prior of candidate answer**) can be treated as the prior probability that node a is the correct answer given v_Q. It describes experts' beliefs when giving no evidence. The intuition is that if a is close to v_Q (e.g., the shortest distance of a to v_Q on the KG is small), it may have a high probability to be the answer. On the other hand, if a is far from v_Q, its probability to be the answer will be low. $Pr(Q|T, v_Q)$ (**likelihood of ambiguous query**) refers to the probability that given the anchor entity v_Q and the true query relation T, how likely the users will input ambiguous query Q. In general, if Q is similar to T, $Pr(Q|T, v_Q)$ should be high. Otherwise, it should be low. $Pr(T|a, v_Q)$ (**posterior of true query**) presents that given v_Q and candidate answer a, how likely T is the correct relation that reflects the true relationship between them.

The left hand side of Eq. (4) shows the main idea of question answering over knowledge graph. The KGQA system first transforms the input natural language question Q to a high quality query relation T, then finds the answer according to T and the anchor node v_Q. The goal of KGQA is to maximize $Pr(T|Q, v_Q)Pr(a|T, v_Q)$, which is equivalent to find query relation T which can maximize $Pr(Q|T, v_Q)Pr(a|v_Q)Pr(T|a, v_Q)$. However, directly calculating Eq. (4) is impossible since we do not know the real world distribution. A general idea is to use $Pr(T|a, v_Q)$ (**posterior of true query**) predict the "hidden relevance" between candidate a and anchor node v_Q. If the "hidden relevance" T is highly related to the original query Q, it can be used to re-rank the top candidate answers. Otherwise, the "hidden relevance" T is considered as noise and ignored.

More specifically, two neural network components are used in PReFNet. The first neural network is used to model $Pr(T|a, v_Q)$ (**posterior of true query**) which performs like a generator to produce high quality candidate query relations (the "hidden relevance" T). The second neural network is used to model $Pr(Q|T, v_Q)$ (**likelihood of ambiguous query**) which behaves like a discriminator to measure the similarity between Q and T, and prunes T if the similarity is too low.

Figure 1 shows the Framework of PReFNet. The yellow part corresponds to the method EmbedKGQA [20] [2]. In the first step, EmbedKGQA is used to find k top candidates based on the input ambiguous query Q. Then, the query inference part ($Pr(T|a, v_Q)$

[2]We use EmbedKGQA because it's one of the state of the art methods.

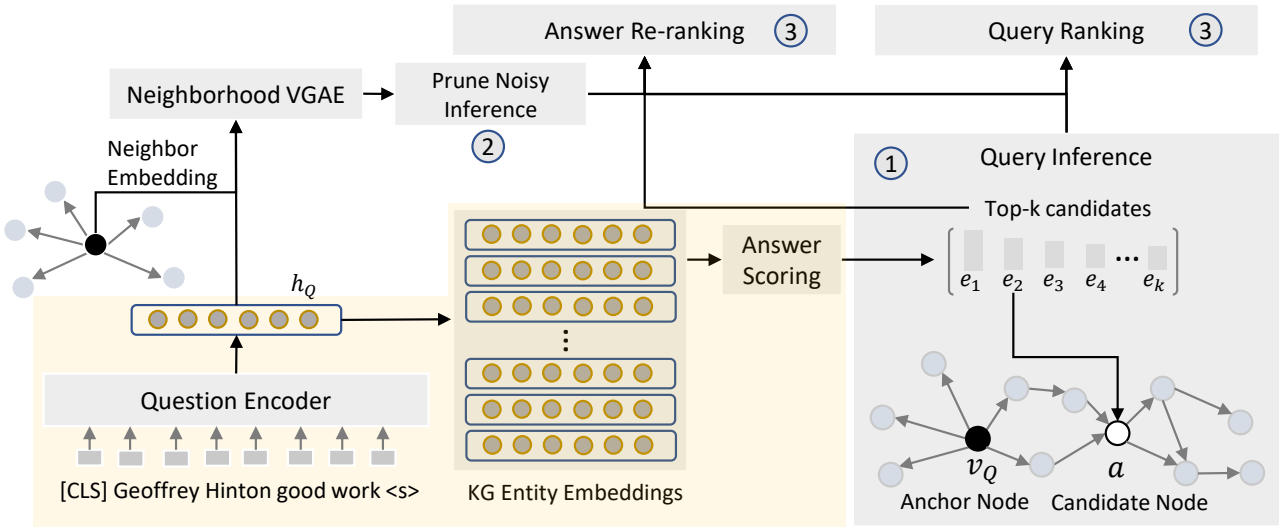

Figure 1: The Framework of PReFNet. The yellow part shows the architecture of EmbedKGQA [20].

subsection 3.2) is used to infer the true relation between anchor node v_Q and each candidate node e_i. Because not every candidate node is good, to boost the quality of those inferred relations, we use a neighborhood embedding based VGAE [10] model to prune low quality relations. Finally, the rest of the high quality inferred relations will be used by Query Ranking ($Pr(Q|T, v_Q)$ subsection 3.4) and Answer Re-ranking (subsection 3.5).

After we have introduced the general idea of PReFNet, in the following subsections, we introduce how to model each component in detail.

3.2 Query Inference: Posterior of True Query

When using pseudo relevance feedback to find out the true intention of the input query, we assume the top-k candidate answers returned by the existing algorithm in the first round is an initial set of most relevant answers. To better understand the query intention, PReFNet infers the query structure according to the anchor entity v_Q, potential/candidate answers, and the background knowledge graph.

When the input query is a one-hop query, this problem is equivalent to the link prediction problem in the knowledge graph. Given an entity pair (v_Q, a), we assume that there is a latent measure for a given entity pair in the KG, i.e. the collection of linked paths; these hidden information can reveal the underlying semantics between these two entities. More specifically, the posterior of the true query $\log Pr(T|a, v_Q)$ can be calculated by

$$\log Pr(T|a, v_Q) = \log \sum_z Pr_\theta(z|a, v_Q) Pr_\eta(T|z, a, v_Q) \quad (5)$$

where η and θ are model parameters, z is the latent path between v_Q and a in the knowledge graph. $Pr_\theta(z|a, v_Q)$ can be treated as the prior distribution denoting how likely z can represent the relation between a and v_Q, and $Pr_\eta(T|z, a, v_Q)$ can be treated as the likelihood that T is the true query between a and v_Q given latent variable z.

A - Estimate $Pr_\theta(z|a, v_Q)$. When calculating $Pr_\theta(z|a, v_Q)$, neighborhood of anchor node v_Q and candidate answer a usually provides us extra information which can imply their similarity to z. For example, if v_Q is surrounded by relations like `bornAt`, `liveIn`, `hasChild`, etc., it alludes that v_Q is a person. And if a is surrounded by `hasMajor`, `locatedIn`, `university_founder`, etc, we can infer that a might be a university. Therefore, if z is a latent path which expresses the relation between a person and a university, $Pr_\theta(z|a, v_Q)$ should have a high value. Otherwise, $Pr_\theta(z|a, v_Q)$ should be low.

In order to encode the neighborhood information of anchor node v_Q and candidate answer a, PReFNet utilizes the relation/edge message passing graph neural network [23] to learn their embeddings as follows

$$m_v^k = \sum_{e \in N(v)} s_e^k,$$
$$s_e^{k+1} = \sigma([m_{v_Q}^k, m_a^k, s_e^i]W^k + b^k), v_Q, a \in N(e),$$

where $[\cdot]$ is the concatenation function, v is an arbitrary node in the knowledge graph, W^k and b^k are the learnable transformation matrix and bias, respectively. s_e^k is the embedding of edge e at iteration k, s_e^0 is the initial edge embedding $\mathbf{r_e}$. Relation/Edge message passing is repeated for $\mathbf{K}$ times. The final message $m_{v_Q}^{\mathbf{K}-1}$ and $m_a^{\mathbf{K}-1}$ are taken as the representation for anchor node v_Q and candidate answer a, respectively.

On the other hand, given a path $z = (p_1, p_2, ..., p_{|z|})$ in the knowledge graph , the path embedding can be calculated by an LSTM as follows:

$$\mathbf{Z} = \text{LSTM}(z), \quad (6)$$

where $\mathbf{Z}$ denotes the embedding with all relational information aggregated throughout path z. The neighborhood information of a

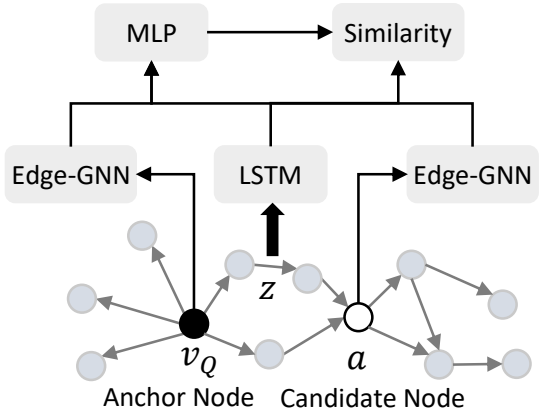

Figure 2: Using relation/edge message passing to estimate $Pr_\theta(z|a, v_Q)$.

and v_Q can be aggregated by

$$\begin{aligned} s_{(v_Q,a)} &= \text{MLP}(m_{v_Q}^{\mathbf{K}-1}, m_a^{\mathbf{K}-1}) \\ &= \sigma([m_{v_Q}^{\mathbf{K}-1}, m_a^{\mathbf{K}-1}]W^{\mathbf{K}-1} + b^{\mathbf{K}-1}). \end{aligned} \tag{7}$$

Finally, the probability $Pr_\theta(z|a, v_Q)$ can be measured by the similarity between $\mathbf{Z}$ and $s_{(v_Q,a)}$:

$$Pr_\theta(z|a, v_Q) = \text{Sigmoid}(s_{(v_Q,a)} \cdot \mathbf{Z}), \tag{8}$$

where $\cdot$ is the dot product. An illustrative example is given in Figure 2.

B - Estimate $Pr_\eta(T|z, a, v_Q)$. For likelihood $Pr_\eta(T|z, a, v_Q)$, it can be modeled by a graph decoder, which takes the embeddings of z, v_Q and a as its initial input and generates a relation. It can be simply modeled as:

$$Pr_\eta(T = r_j|z, a, v_Q) = \text{softmax}(\text{FNN}(s_{(v_Q,a)}||\mathbf{Z}))[j], \tag{9}$$

where FNN is the feedforward neural network.

3.3 Query Inference Training

Directly optimizing the objective in Eq. (5) is challenging, because the summation over all possible paths is intractable. In order to maximize Eq. (5), we resort to variational inference and minimize its negative evidence lower bound:

Proposition 1. *Let $\mathcal{L}(\phi, \theta, \eta)$ be defined as in Eq.* (10)*, we have* $\log Pr(T|a, v_Q) \geq -\mathcal{L}(\phi, \theta, \eta)$

$$\begin{aligned} \mathcal{L}(\phi, \theta, \eta) =& E_{z\sim q_\phi(z|T,a,v_Q)}[-\log Pr_\eta(T|z, a, v_Q)] \\ &+ KL(q_\phi(z|T, a, v_Q)|Pr_\theta(z|a, v_Q)) \end{aligned} \tag{10}$$

where $q_\phi(z|T, a, v_Q)$ is the variational posterior.

Proof. According to Eq. (5), we have

$$\begin{aligned} \log Pr(T|a, v_Q) &= \log \sum_z Pr_\theta(z|a, v_Q) Pr_\eta(T|z, a, v_Q) \\ &= \log \sum_z q_\phi(z|T, a, v_Q) \frac{Pr_\theta(z|a, v_Q) Pr_\eta(T|z, a, v_Q)}{q_\phi(z|T, a, v_Q)} \\ &\geq \sum_z q_\phi(z|T, a, v_Q) \log \frac{Pr_\theta(z|a, v_Q) Pr_\eta(T|z, a, v_Q)}{q_\phi(z|T, a, v_Q)} \\ &= E_{z\sim q_\phi(z|T,a,v_Q)}[\log Pr_\theta(z|a, v_Q) \\ &+ \log Pr_\eta(T|z, a, v_Q) - \log q_\phi(z|T, a, v_Q)] \\ &= -\mathcal{L}(\phi, \theta, \eta) \end{aligned} \tag{11}$$

□

According to Proposition 1, Eq. (10) is the negative lower bound of Eq. (5). Therefore, minimizing Eq. (10) can best approximate $\log Pr(T|a, v_Q)$.

When approximating the posterior $q_\phi(z|T, a, v_Q)$, we use the same model as prior (Eq. (8)), except that we replace Eq. (8) by

$$\begin{aligned} x &= s_{(v_Q,a)} + e_T \\ q_\phi(z|T, a, v_Q) &= \text{Sigmoid}(x \cdot \mathbf{Z}) \end{aligned} \tag{12}$$

where e_T is the embedding of $T.$, and $\cdot$ is the dot product.

When updating the model, we leverage gradient descent to minimize the loss. The gradient of the loss w.r.t. $q_\phi(z|T, a, v_Q)$ can be calculated as:

$$\begin{aligned} \nabla_\phi \mathcal{L} = E_{q_\phi(z|T,a,v_Q)}[&\nabla_\phi q_\phi(z|T, a, v_Q)(\log Pr_\theta(z|a, v_Q) \\ &+ \log Pr_\eta(T|z, a, v_Q) - \log q_\phi(z|T, a, v_Q))]. \end{aligned} \tag{13}$$

The gradient in Eq. (13) can be approximated by the Monte Carlo method using N samples of the latent variable from variational distribution

$$\begin{aligned} \nabla_\phi \mathcal{L} \approx \frac{1}{N} \sum_{i=1}^{N} [&\nabla_\phi q_\phi(z_i|T, a, v_Q)(\log Pr_\theta(z_i|a, v_Q) \\ &+ \log Pr_\eta(T|z_i, a, v_Q) - \log q_\phi(z_i|T, a, v_Q))]. \end{aligned} \tag{14}$$

When updating other parameters: θ, η, the gradient of $\mathcal{L}(\phi, \theta, \eta)$ with respect to η can be calculated directly, and the gradient of $\mathcal{L}(\phi, \theta, \eta)$ with respect to θ is equivalent to calculating the gradient of KL-divergence of $KL(q_\phi(z|T, a, v_Q)|Pr_\theta(z|a, v_Q))$ with respect to θ.

3.4 Query Ranking: Likelihood of Ambiguous Query

After we introduce how to model $Pr(T|a, v_Q)$ (**posterior of true query**), we now introduce how to model $Pr(Q|T, v_Q)$ (**likelihood of ambiguous query**).

Given the initial top-k answer candidates, a set of potential true query relations are generated by $Pr(T|a, v_Q)$ (**posterior of true query**). Because noise may exist in the initial candidate set, to boost the qualify of the new generated T, we use a neighborhood embedding based VGAE [10] to prune low quality T. The details of the neighborhood embedding based VGAE model is given in Appendix.

After pruning low qualify inferred queries, we use another neural network to model $Pr(Q|T, v_Q)$ which calculates the similarity between T and the input ambiguous query Q. Inspired by [1], $Pr(Q|T, v_Q)$ is approximated according to their distance in the embedding space. Given a ambiguous query, since the query is a natural language question, we use a pre-trained BERT [5] to learn its embedding:

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

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

$$\mathbf{h}_Q = \text{FNN}([\mathbf{h}_{CLS}]).$$

When training the Query Ranking model ($Pr(Q|T, v_Q)$), we randomly sample some negative relations from the knowledge graph. The positive query relations are given in the training data. In summary, given a positive pair and a negative pair, the loss is defined as

$$L = \sum_{e_T} \sum_{e_{T'}} [\gamma + d(h_Q, e_T) - d(h_Q, e_{T'})], \tag{15}$$

where $e_{T'}$ is the embedding of the negative sample T', $d(h_Q, e_T)$ measures the distance between embedding h_Q and e_T, which can be either the L_1 or the L_2-norm, and γ is a margin hyperparameter.

During the test, we use the neighborhood embedding based VGAE to measure the quality of the inferred queries. If the score is greater than 0.5, we think the queries have high quality and return them directly. Otherwise, all the high qualify potential queries generated by query inference $Pr(T|a, v_Q)$ are ranked according to their distances calculated by $d()$ in Eq. (15).

3.5 Answering Re-ranking

After getting high quality inferred queries, we can re-rank all the top-k candidate answers. The ranking function is calculate by $\sum_T Pr(a|T, v_Q)$. Similar to [20], we approximate $\sum_T Pr(a|T, v_Q)$ as

$$\log \sum_T Pr(a|T, v_Q) \propto \gamma * |R_a \cap R_T|,$$

where R_a is the set of relations on the shortest path between anchor entity v_Q and candidate answer a. R_T is the top-ranked potential queries generated by query inference $Pr(T|a, v_Q)$ (**posterior of true query**). To keep the high quality of the inferred query T, we prune those queries which are scored less than 0.2 by the neighborhood embedding based VGAE [10] model.

4 EXPERIMENTS

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

We aim to answer the following questions.

4.1 Experimental Setting

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

- **WebQuestionsSP** contains both the 1-hop questions and 2-hop natural language questions. There are 4,000 questions in total and all questions can be answered by Freebase.
- **SimpleQuestions** contains more than 100K simple 1-hop natural language questions which can be answered by Freebase.
- **MetaQA** is a dataset on movie domain which contains more than 400K natural language questions. The background knowledge graph contains information about directors, movies, genres and actors. We use the 1-hop questions in our experiment.

The statistics of these datasets are shown in Table 10 in Appendix. Their corresponding knowledge graphs are shown in Table 11 in Appendix. In the experiment, we test the effectiveness of PReFNet on complete KG and incomplete KG with 50% and 20% missing edges respectively. All missing edges are randomly deleted.

We test the query ranking performance on 4 baselines, including

- HGNet [4] uses a generation method to transform the input query to a SPARQL query and uses it to find the results.
- RnG-KBQA [28] adopts a bootstrapping strategy to train a query graph ranker and generator, and searches the background knowledge graph according to the generated query.
- VGAE is a variant of variational autoencoder [10]. We model the encoder component with a pre-trained Bert [5] and decoder component with a feedforward neural network. No neighborhood information is used here.
- Neighborhood VGAE is the relation pruning model used in the proposed PReFNet which has been introduced in subsection 3.4. The details are given in Appendix.

We test question answering performance on 3 baselines, including:

- GraftNet [22] finds a question-specific subgraph containing KG facts, and then uses a graph neural network to predict the answers.
- Key-Value Memory Network (KVMem) [16] maintains a memory table which stores KG facts and uses this for retrieval.
- EmbedKGQA [20] conducts multi-hop reasoning through matching pre-trained entity embeddings with question embedding obtained from RoBERTa [5]. We use EmbedKGQA with relation matching which has better performance compared with the EmbedKGQA without relation matching.

Another related method is GRF [21]. However, neither the code nor the datasets are publicly available, and thus we leave the comparison with GRF to future work.

4.2 Performance of Query Ranking

Traditional KBQA methods usually transform the natural language query to a query graph, and then find the answer according to the query graph. However, because of the ambiguity in the input query, the generated query graph is usually inaccurate. The pseudo relevance feedback, on the other hand, can infer queries according to the top candidate answers. We run the experiments on datasets WebQSP and SimpleQA. Because the data format of the background knowledge graph of MetaQA is different from that required by

Table 2: Query ranking Hit@1.

	Webqsp 50% KG	Webqsp 80% KG	Webqsp full	SimpleQA 50% KG	SimpleQA 80% KG	SimpleQA full
HGNet	52.5	55.3	58.2	-		
RnG-KBQA	61.2	63.3	66.7	-		
VGAE	70.7	71.3	73.7	87.4	89.1	91.6
Neighborhood-VGAE	74.2	76.1	78.1	90.3	92.3	93.5
PREFNET	75.7	77.2	79.0	90.8	92.7	94.5

Table 3: KGQA Hit@1 results of Webqsp on complete, 50% and 80% completeness knowledge graphs.

Model	Webqsp 50%	Webqsp 80%	Webqsp
GraftNet	32.7	39.8	-
KVMem	44.2	-	46.7
HGNet	37.8	45.4	46.1
RnG-KBQA	40.6	48.5	53.1
EmbedKGQA	46.3	52.8	56.7
PREFNET	47.7	53.7	57.2

Table 4: KGQA Hit@1 results of SimpleQA on complete, 50% and 80% completeness knowledge graphs.

Model	simpleQA50%	simpleQA80%	simpleQA
GraftNet	39.8	-	-
KVMem	28.9	-	-
EmbedKGQA	46.5	58.9	64.3
PREFNET	48.8	60.1	64.8

Table 5: KGQA Hit@1 results of MetaQA on complete, 50% and 80% completeness knowledge graphs.

Model	MetaQA 50%	MetaQA 80%	MetaQA
GraftNet	64.0	-	-
KVMem	63.6	-	-
EmbedKGQA	83.1	90.1	95.3
PREFNET	83.9	90.6	95.6

HGNet and RnG-KBQA, we omit the query ranking experiment on MetaQA.

Following the same setup as in [20], we evaluate the accuracy using the Hit@1 metrics which is the fraction of times a correct answer was retrieved within the top-1 positions. Table 2 shows the results of Query Ranking. Four different methods have been used in the experiments. HGNet and RnG-KBQA are knowledge graph question answering algorithms. They first transform the ambiguous natural language question to a query graph, and then search the knowledge graph according to the generated query graph. VGAE is a variational autoencoder and decoder method. Neighborhood-VGAE is the proposed pruning method used in PREFNET.

As observed in Table 2, the proposed PREFNET has the highest query ranking accuracy compared with other baselines. For Webqsp, the performance boosts are 5.0% and 5.9% with 50% and 80% of the underlying knowledge graph respectively, compared with VGAE, and 0.9% on average compared with Neighborhood-VGAE. Similar results can be found in complete knowledge graph as well. Besides, compared with 50% knowledge graph, the query ranking accuracy on 80% and full knowledge graph of all four baselines is higher. This means if the background knowledge graph becomes more complete, all baselines tend to generate more accurate query graphs.

4.3 Performance of Question Answering

Table 3 shows the results of all baseline methods on Webqsp dataset with 50% complete KG, 80% complete KG and complete KG, respectively. As shown in Table 3, the Hit@1 of all methods decreases as the sparsity (incompleteness) of the background knowledge graph increases. This means that the quality of the background knowledge graph has a significant impact on the KGQA task. Among all the methods, EmbedKGQA with relation matching can achieve the highest accuracy. PREFNET further increases the accuracy by 1% on average. Table 4 and Table 5 show the performance of different methods on SimpleQuestions and MetaQA datasets. Similar results can be observed.

4.4 Efficiency

Figure 3 shows the training time and test time of the query inference component on different datasets. As we can see, the runtime of the Query Inference component 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 the Query Inference component is relatively short.

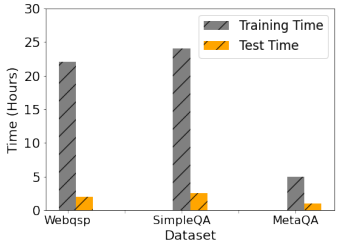

Figure 3: Query Inference Training and Test Time.

Compared with the running time of query inference, the running time of the query ranking (subsection 3.4) and answer re-ranking (subsection 3.5) is negligible (e.g., less than 20 minutes on the whole test dataset).

4.5 Ablation Study

A - Query Inference. In this subsection, we show the effectiveness of the query inference module. We first pretrain the module only on the background knowledge graph of each dataset, and then retrain the module on the question training dataset. During the pretrain and retrain processes, we split all the data into training set (80%), valid set (10%) and test set (10%). As shown in Table 6 [3], the query

[3]For the full background knowledge graph, because we can directly find correct paths between anchor node v_Q and candidate answer node a, we do not need query inference module.

inference module has a very high Hit@K on all datasets. It can achieve 88.8% Hit@1, 95.8% Hit@3 and 96.9% Hit@5 on average.

Table 6: Query Inference Performance. wsp means WebQuestionsSP. sQA means SimpleQuestions.

Model	wsp50%	wsp80%	sQA50%	sQA80%	MetaQA50%	MetaQA80%
Hit@1	82.4	87.4	92.6	95.9	86.2	88.4
Hit@3	91.8	93.4	98.8	99.6	98.1	93.2
Hit@5	93.7	94.7	99.3	99.9	99.9	94.1

B - Query Ranking.

In this subsection, we show the effectiveness of the query ranking module. Some examples are shown in Table 7. As we can see, when the input question is ambiguous, it is very hard to correctly predict its true query intention. For example, for query "thomas jefferson role in declaration independence", the query generator VGAE thinks the query intention is "film.actor.film". Because the phrase "role in" is usually used to describle "someone plays/acts a role in a movie". However, with pseudo relevance feedback, the query inference module can discover "thomas jefferson" is a stateman/diplomat, and that helps the module correctly predict the true intention "people.person.profession" and finds the correct answer (statesperson) in the background knowledge graph.

C - Question Answering.

Table 8 shows the ablation study on question answering task. EmbedKGQA - RM stands for EmbedKGQA without relation matching. EmbedKGQA+QI denotes that we add the query information inferred by query inference module to relation matching process in EmbedKGQA. More specifically, the relation matching process of EmbedKGQA finds the shortest path between the anchor node v_Q and candidate answer a, and order the candidate answers according to their shortest paths. We add the relation inferred by query inference module to the shortest path when ordering the candidate answer. It is observed that the proposed PReFNet achieves the best performance.

5 RELATED WORK

5.1 Knowledge Graph Question Answering

Knowledge graph has many applications [2, 3, 6, 7, 9, 12, 15, 19, 24–26]. Among them, knowledge Graph Question Answering has been studied for a long time. When the input query is a natural language sentence, a general strategy to answer the question is to transform the question to a query graph, and search the answer according to the query graph. For example, in [28], Xi et.al. propose a model which contains candidate query graphs ranking component and a true query graph generation component. By iteratively updating these two components, both components' performance can be improved. The query graph is finally generated by the second component and can be used to search the KG. In [14], Liu et.al. propose a multi-task model to tackle KGQA and KGC at the same time. Other methods, e.g., [20], [16], directly learn an embedding from the natural language sentence and search answers in the embedding space. When the input query is a graph query, [18] models different operations in the query graph as different neural network and transform the query process to an entity search problem in the embedding space. In principle, all of them can be used as the query system in our method. That is, the top-k answers of these methods can be treated as the pseudo relevance feedback of our method.

5.2 Relevance Feedback

Relevance feedback is a widely studied topic in Information Retrieval. However, it has not been well studied for graph data. In [21], Su et.al. use relevance feedback to infer additional information and use them to enrich the query. The original ranking function is re-tuned according to the results in relevance feedback. In [11], Matteo et.al. concentrate on assisting the user by expanding the query according to the additional information in relevance feedback to provide a more informative (full) query that can retrieve more detailed and relevant answers. However, different from our work which aims to infer the true intention of users, they expand the query graph at each round until they find the answer. In other words, the setting is different.

5.3 Variational Inference

The goal of variational inference is to approximate difficult-to-compute posterior density. In [29], Zhang et.al. treat the topic entity in the input question as a latent variable and utilize variational reasoning network to handle noise in questions, and learn multi-hop reasoning simultaneously. In [17], Qu et.al. propose a probabilistic model called RNNLogic which treats logic rules as latent variables, and simultaneously trains a rule generator as well as a reasoning predictor with logic rules. These logic rules are similar to the latent paths in our model. There are many other works using variational inference. Different from these works, we are the first to utilize variational inference in relevance feedback on graph data.

6 CONCLUSION

In this paper, we propose a model PReFNet which utilizes pseudo relevance feedback to answer ambiguous query over knowledge graph. Guided by Bayes rule, we decompose the question answering problem to several components and model each component with a neural network. To infer hidden queries, we resort to variational inference to update intractable model with latent variables. To ensure the high quality of the inferred queries, we propose a neighborhood embedding based VGAE model to prune inferior queries. The experiment results show that the proposed PReFNet consistently boosts the state-of-the-art methods on both query ranking and question answering tasks on multiple datasets.

ACKNOWLEDGMENTS

The LL and HT are partially supported by NSF (1947135, 2134079 and 1939725), DARPA (HR001121C0165), NIFA (2020-67021-32799) and ARO (W911NF2110088).

REFERENCES

[1] A Bordes and Usunier N. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26 (NeurIPS '21).*

[2] Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. [n. d.]. Variational Knowledge Graph Reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational.* Association for Computational Linguistics.

Table 7: Results of relation found by different methods.

Question	Query Relation found by PReFNet	Query Relation found by VGAE
thomas jefferson role in declaration independence	people.person.profession	film.actor.film
state mount st. helens in	location.location.containedby	geography.mountain.mountain_type
monarchy japan	location.country.form_of_government	schema.administrative_area.administrative_children
first gulf war fought	time.event.locations	military_command.military_commander
buddha come	people.person.place_of_birth	people.person.places_lived
part country new england	location.location.containedby	base.locations.countries.continent
drink john pemberton create	inventor.inventions	symbols.name_source.namesakes
jesus after he died cross	people.deceased_person.place_of_death	people.person.profession
charles babbage discover	inventor.inventions	base.argumentmaps.innovator.original_ideas
country code mexico	location.country.internet_tld	location.location.adjoin_s

Table 8: Ablation study of question answering.

Model	webqsp50%	webqsp80%	simpleQA50%	simpleQA80%
EmbedKGQA - RM	44.8	50.3	46.5	58.9
EmbedKGQA	45.9	52.8	46.6	58.8
EmbedKGQA+QI	46.3	52.8	46.7	58.9
PReFNet	47.7	53.7	48.8	60.1

[3] Xiusi Chen, Yu Zhang, Jinliang Deng, Jyun-Yu Jiang, and Wei Wang. [n. d.]. *Gotta: Generative Few-shot Question Answering by Prompt-based Cloze Data Augmentation.*

[4] Yongrui Chen, Huiying Li, Guilin Qi, Tianxing Wu, and Tenggou Wang. 2021. Outlining and Filling: Hierarchical Query Graph Generation for Answering Complex Questions over Knowledge Graph. arXiv.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

[6] 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.

[7] Dongqi Fu and Jingrui He. 2021. SDG: A Simplified and Dynamic Graph Neural Network. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2273–2277. https://doi.org/10.1145/3404835.3463059

[8] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The Vocabulary Problem in Human-System Communication. (1987).

[9] Zijie Huang, Zheng Li, Haoming Jiang, Tianyu Cao, Hanqing Lu, Bing Yin, Karthik Subbian, Yizhou Sun, and Wei Wang. 2022. Multilingual Knowledge Graph Completion with Self-Supervised Adaptive Graph Alignment. In *Annual Meeting of the Association for Computational Linguistics (ACL).*

[10] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. arXiv.

[11] Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis. 2020. *Graph-Query Suggestions for Knowledge Graph Exploration.* Association for Computing Machinery, New York, NY, USA, 2549–2555. https://doi.org/10.1145/3366423.3380005

[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] L. Liu, B. Du, and H. Tong. 2019. GFinder: Approximate Attributed Subgraph Matching *(BigData '19).*

[14] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint Knowledge Graph Completion and Question Answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1098–1108.

[15] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural Network-Based Question Answering over Knowledge Graphs on Word and Character Level. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1211–1220. https://doi.org/10.1145/3038912.3052675

[16] 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]

[17] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. 2020. RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs. arXiv. https://doi.org/10.48550/ARXIV.2010.04029

[18] Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space using Box Embeddings. In *International Conference on Learning Representations.*

[19] Marta Sabou and Konrad Höffner. 2017. Survey on Challenges of Question Answering in the Semantic Web. *Semant. Web* 8, 6 (jan 2017), 895–920. https://doi.org/10.3233/SW-160247

[20] 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.

[21] Yu Su, Shengqi Yang, Huan Sun, Mudhakar Srivatsa, Sue Kase, Michelle Vanni, and Xifeng Yan. 2015. Exploiting Relevance Feedback in Knowledge Graph Search *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2783258.2783320

[22] 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]

[23] Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2021. Relational Message Passing for Knowledge Graph Completion. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21).*

[24] Ruijie Wang, Zheng Li, Danqing Zhang, Qingyu Yin, Tong Zhao, Bing Yin, and Tarek Abdelzaher. 2022. RETE: Retrieval-Enhanced Temporal Event Forecasting on Unified Query Product Evolutionary Graph. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. 462–472.

[25] Ruijie Wang, zheng li, Dachun Sun, Shengzhong Liu, Jinning Li, Bing Yin, and Tarek Abdelzaher. 2022. Learning to Sample and Aggregate: Few-shot Reasoning over Temporal Knowledge Graphs. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=1LmgISIDZJ

[26] 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.

[27] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. BRIGHT: A Bridging Algorithm for Network Alignment. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 3907–3917. https://doi.org/10.1145/3442381.3450053

[28] Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. RnG-KBQA: Generation Augmented Iterative Ranking for Knowledge Base Question Answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

[29] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2017. Variational Reasoning for Question Answering with Knowledge Graph. arXiv.

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/PrefNet.

Baselines. The implementation code of HGNet, RnG-KBQA, GraftNet, Key-Value Memory Network and EmbedKGQA can be found from the Table below.

Table 9: Baseline Source.

Model	github
HGNet	https://github.com/Bahuia/HGNet
RnG-KBQA	https://github.com/salesforce/rng-kbqa
GraftNet	https://github.com/haitian-sun/GraftNet
KV-Mem	https://github.com/jojonki/key-value-memory-networks
EmbedKGQA	https://github.com/malllabiisc/EmbedKGQA

THE NEIGHBORHOOD EMBEDDING BASED VAGE

In this section, we introduce the details of the neighborhood embedding based VAGE model in PREFNET. The goal of Neighborhood-VGAE is to score the quality of a relation r_i w.r.t. the given ambiguous query Q. Given a ambiguous query Q, a pretrained Bert model is used to learn its embedding, which is given below.

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

To denoise the query, we utilize the idea of variational autoencoder to generate the mean μ and variance δ from the query embedding, and randomly sample a point from the distribution.

$$\mu = \text{MLP}(\mathbf{h}_Q)$$
$$\delta = \text{MLP}(\mathbf{h}_Q)$$
$$\Lambda = \mu + \delta \odot \epsilon$$

where $\epsilon \sim N(0, \mathbf{I})$.

The quality of a relation r_i is calculated by

$$f(r_i) = \text{softmax}(\text{FNN}(\Lambda || H(v_Q)))[i],$$

where $H(v_Q)$ is the neighborhood embedding of anchor node v_Q.

STATISTICS OF DATASETS

Table 10 represents the details of the datasets used in this paper.

Table 10: Number of queries in each dataset.

Dataset	Train	Valid	Test
MetaQA	96,106	9,992	9,947
WebQSP	2,998	100	1,639
SimpleQA	15,3188	2,105	4,345

Table 11 shows the statistics of different KGs used in the experiments.

Table 11: Statistics of the 50% KG, 80% KG and complete KG for the three datasets. Note that WebQSP and SimpleQA use the same background knowledge graph.

50% KG				
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
80% KG				
MetaQA	43,234	18	109,520	4,000
WebQSP	1,886,683	1,144	4,624,196	20,000
SimpleQA	1,886,683	1,144	4,624,196	20,000
full KG				
MetaQA	43,234	18	133,582	4,053
WebQSP	1,886,683	1,144	5,780,246	20,000
SimpleQA	1,886,683	1,144	5,780,246	20,000

LIMITATIONS AND FUTURE WORKS

One limitation of the proposed PREFNET model lies in its generalizabilitys. In this paper, PREFNET can only support one hop queries. To generalize multihop queries, one way is to let the output of the Query Inference be the distribution of multihop queries. More specifically, we can replace Eq. (9) as a path decoder. For future improvements, the inferred queries can be extended to handle multihop queries, and the variational inference model can be swapped with Bayesian Variational Inference to enhance its robustness.