PAGE: Answering Graph Pattern Queries via Knowledge Graph Embedding

Sanghyun Hong[1], Noseong Park[2*], Tanmoy Chakraborty[3], Hyunjoong Kang[4], and Soonhyun Kwon[4]

[1] University of Maryland, College Park, Maryland, USA
[2] University of North Carolina, Charlotte, North Carolina, USA
[3] Indraprastha Institute of Information Technology Delhi, Delhi, India
[4] Electronics and Telecommunications Research Institute, Daejeon, South Korea
shhong@cs.umd.edu[1], npark2@uncc.edu[2], tanmoy@iiitd.ac.in[3], hjkang@etri.re.kr[4], kwonshzzang@etri.re.kr[4]

Abstract. Answering graph pattern queries have been highly dependent on a technique—i.e., subgraph matching, however, this approach is ineffective when knowledge graphs include incorrect or incomplete information. In this paper, we present a method called `PAGE` that answers graph pattern queries via knowledge graph embedding methods. `PAGE` computes the energy (or uncertainty) of candidate answers with the learned embeddings and chooses the lower-energy candidates as answers. Our method has the two advantages: 1) `PAGE` is able to find latent answers hard to be found via subgraph matching and 2) presents a robust metric that enables us to compute the plausibility of an answer. In evaluations with two popular knowledge graphs, Freebase and NELL, `PAGE` demonstrated the performance increase by up to 28% compared to baseline KGE methods.

Keywords: Graph Databases; Graph Query Answering; Knowledge Graph Embedding

1 Introduction

Graphs/networks are widely used in various fields, e.g., knowledge graphs (KGs) in the Semantic Web, social networks in Social Analytics, protein-protein interaction (PPI) networks in Bioinformatics, etc. As their applications are diverse, many different graph mining paradigms have been proposed: the Semantic Web has its own knowledge graph query language called SPARQL [14], and Neo4j [10], the market-leading graph database management system, also has a graph query language called Cypher. Unfortunately, that progress has been made at search *subgraph patterns* from underlying graphs via subgraph isomorphism, often hard to find answers when the graphs are incomplete or carry incorrect information [11].

Graph embedding methods have come into the light nowadays because of their promising performance in various tasks such as community detection [7], link prediction in the social network [12, 17], and query answering on knowledge

*Corresponding author.

graphs [2]. Those methods learn *latent vector representations (or embeddings)* of vertices and relations[5]. Prior works have reported that using embeddings can provide a way to answer factoid queries[6] even with incorrect and incomplete information [2, 3, 4]. However, KGE methods have only considered simple queries consisting of a single edge or multiple unidirectional edges—i.e., it has not been explored whether we can use them to answer general graph queries.

In this paper, we introduce `PAGE` (**P**attern query **A**nswering through knowledge **G**raph **E**mbedding) that delivers a new paradigm of querying KGs. To the best of our knowledge, we are the first effort to combine them and explore the potential of KGE methods in answering graph queries. Advantages of the proposed approach are twofold:

1. *Our method can discover latent patterns which remain hidden in the incomplete or incorrect KGs.* Rather than relying on the subgraph matching, `PAGE` chooses candidate answers for a graph query based on the energy computed with embeddings, which enables our method to return complete answers.
2. *We present a robust metric that can compute the plausibility of an answer.* This metric aids in post-processing after querying KGs. There can be numerous subgraph patterns matched to a graph query; processing all of them is computationally too expensive. In `PAGE`, we only identify highly plausible subgraph patterns and provide them as candidate answers.

In evaluations, we conduct two experiments: 1) factoid query answering and 2) graph query answering with two popular KGs, Freebase and NELL. Our result demonstrates that `PAGE` outperforms baseline KGE methods by at most 28% in terms of standard metrics such as mean rank and Hits@10/100/1000. The evaluation results show the potential of using KGE methods for answering graph queries in KGs even though the KGs carry incomplete information.

2 Background

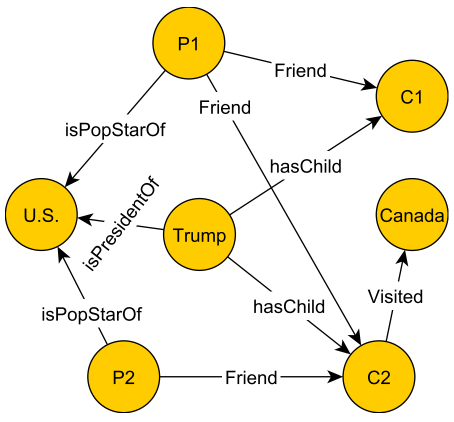
In this section, we introduce the basic concepts of graph pattern query answering and KGE. Let $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ be a KG and L be a set of relations. $\mathcal{V}_\mathcal{G}$ is a set of vertices and $\mathcal{E}_\mathcal{G}$ is a set of edges labeled by one of the relations in L. A relation in L means a certain type of relationship between vertices.

2.1 Graph Query Answering

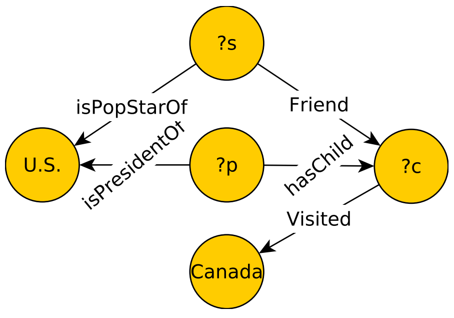
Given a KG $\mathcal{G}$ and a graph query $\mathcal{Q} = (\mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q})$, the task of conventional graph query answering is to find all subgraph patterns of $\mathcal{G}$ matched to $\mathcal{Q}$ via subgraph isomorphism [16].

[5]A relation is an edge label. A KG is an edge-labeled graph.

[6]Factoid queries are the simplest type of graph queries such as “who visited Canada?”, denoted as $?x \xrightarrow{visited} Canada$.



(a) Knowledge Graph $\mathcal{G}$



(b) Graph Pattern Query $\mathcal{Q}$

Fig. 1: An example KG (a) and graph pattern query (b). (Note that $?p$, $?c$, and $?s$ are variables in the query (b), and the answers are $Trump$, $C2$, and $P2$ of $\mathcal{G}$.)

Example 1 (Graph Pattern Query with Projection). In Figure 1, the graph query searches for a child $?c$ of the president $?p$ of the United States, who had visited Canada before and is a friend of a pop star $?s$. The answer is C2 because $\{?c = \text{C2}, ?p = \text{Trump}, ?s = \text{P1}\}$ is a valid subgraph that matches the query.

In query languages, a graph pattern query is expressed as a series of path queries. We write the example query in Neo4j's Cypher query language as follows:

Query 1.1: Cypher representation of the query in Figure 1 (b).

```
MATCH (US)<−[:isPresidentOf]−(?p)−[:hasChild]−>(?c)
MATCH (US)<−[:isPopStarOf]−(?s)−[:Friend]−>(?c)
MATCH (?c)−[:Visited]−>(CANADA)
RETURN ?c
```

Note that each path query is projected to a vertex matched to $?c$. As shown in the above expression, *the problem of answering a graph query can be decomposed into: answering each path query, coming up with a set of candidate answers, and choosing the common answer among candidates.* This observation sheds light on how we can answer the general form of graph queries.

2.2 Knowledge Graph Embedding

KGE methods map vertices and relations into a d-dimensional continuous vector space. `TransE`, `SE`, and `SME` [2, 3, 4] are the most popular and pioneering works; those methods answer a factoid query by using the concept of energy. Given a semantic triple $t = (\mathbf{v}, \mathbf{r}, \mathbf{u})$ in a KG, the energy of the triple indicates the uncertainty (or error) such that a high energy level means a high uncertainty of the triple. The v and u stand for vertices, r is a relation between them, and we

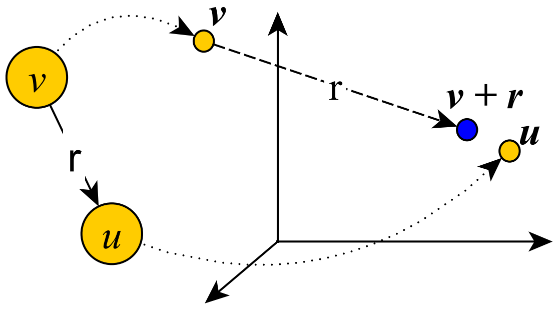


Fig. 2: An example of `TransE` embedding: in `TransE`, a triple v, r, u (on the left side) are mapped into the 3-dimensional space (on the right side). $\mathbf{v}+\mathbf{r}$, represented by the blue vertex, is the prediction of v's neighbor connected through r that minimizes the energy (error) $\|\mathbf{v}+\mathbf{r}-\mathbf{u}\|$, which refers to the distance between the blue vertex and $\mathbf{u}$.

use a bold character to denote an embedding vector. The energy can be defined in various ways:

1. In `TransE`, $e(v,r,u) = \|\mathbf{v}+\mathbf{r}-\mathbf{u}\|_{1 \text{ or } 2}$[7].
2. In `SME`, $e(v,r,u) = g_v(\mathbf{r},\mathbf{v})^{\mathrm{T}} \cdot g_u(\mathbf{r},\mathbf{u})$, where g_v and g_u are linear or bilinear functions.
3. In `SE`, $e(v,r,u) = \|\mathbf{r_l v} - \mathbf{r_r u}\|_{1 \text{ or } 2}$, where $\mathbf{r_l}$ and $\mathbf{r_r}$ are left and right projection matrices[8] representing r.

In training, KGE methods learn the vector representations (or embeddings) of vertices and relations by minimizing the following loss function:

$$\mathcal{L} = \sum_{t^+ \in \mathcal{E}} \sum_{t^- \in \mathcal{N}(t^+)} \max\big(0, \gamma + e(t^+) - e(t^-)\big), \tag{1}$$

where $\mathcal{E}$ is a set of triples, $\mathcal{N}(t^+)$ is a set of negative triples t^- derived from true triples $t^+ \in \mathcal{E}$, γ is a margin, and $e(\cdot)$ is the energy of a triple. For instance, $e(\mathbf{v},\mathbf{r},\mathbf{u}) = \|\mathbf{v}+\mathbf{r}-\mathbf{u}\|$ in `TransE` such that a true triple that exists in the KGs makes the $e(\cdot)$ zero—i.e., $\mathbf{v}+\mathbf{r}=\mathbf{u}$ (see Figure 2). Thus, training with this loss function decreases the energy of the true triples while increasing the energy of false triples such that they differ by at least γ.

With the learned embeddings, existing works answer two types of queries: 1) factoid queries $u \xrightarrow{r} ?x$ and 2) unidirectional path queries $u \xrightarrow{r_1, r_2, \cdots} ?x$[9] [6] by finding the top-k answers that have the smallest energy among all the candidate answers for $?x$.

[7] $\|\cdot\|_1$ (resp. $\|\cdot\|_2$) refers to the ℓ_1-norm (resp. ℓ_2-norm) of a vector.

[8] After vectorization, a matrix can still be represented by a vector.

[9] Note that r_i means an intermediate relation in the path between u and $?x$, and all the relations have the same direction.

Finding a set of candidate answers relying on the energy provides two advantages: 1) this enables us to find the latent answers, and 2) the methods works with KGs that include incomplete information. In this work, we *further extend the KGE methods to answer the general form of a graph pattern query*, which enables to answer the query without subgraph isomorphism (or subgraph matching).

3 Graph Query Answering via KGE

Using existing KGE methods to answer graph pattern queries has a major problem—i.e., those methods are limited to answer factoid or unidirectional path queries (see Sec. 2.2). However, general graph queries involve *bi-directional path queries* as shown in Query 1.1. In addition, it is well-known that considering multiple-hop paths makes KGE methods vulnerable to accumulated errors because an error in an edge can be amplified after multiple hops [6]. This motivates us to present a new query model and a training method. In this section, we introduce `PAGE`, a novel method that enables us to answer graph queries on incomplete KGs via knowledge graph embeddings without relying on subgraph isomorphism.

3.1 `PAGE` Energy Definition and Query Model

Dropping Subgraph Isomorphism. Subgraph isomorphism (or subgraph matching) has traditionally been considered as the key to answer graph pattern queries. However, the quality of the answers drastically decreases when the underlying KG is not complete or contains incorrect knowledge. This is the well-known problem since constructing a KG from web pages or documents is challenging, and the KG usually carries incorrect knowledge representation [15]. To overcome this issue, we drop the subgraph isomorphism in the proposed `PAGE` query model. Instead, we utilize KGE methods that provide high accuracy in answering factoid or unidirectional queries and is able to rectify incorrect knowledge [15].

The Energy of a Bidirectional Path Query. In our model, we consider a graph query as a set of bidirectional path queries (see Sec. 2.1). To answer a graph query, we first need to answer each bidirectional path query via KGE methods. However, most KGE methods have been proposed without considering bidirectional path queries—i.e., the operators used to compute the energy are not invertible. Thus, we define the regular and inverse energy operations as follows:

Definition 1 (Regular Operation). *Given a query $v \xrightarrow{r} ?x$, the regular operation is to find x such that $energy(\mathbf{v}, \mathbf{r}, \mathbf{x}) = 0$, e.g., $\mathbf{x} = \mathbf{v} + \mathbf{r}$ in* `TransE`*. This answers a query $v \xrightarrow{r} ?x$.*

Definition 2 (Inverse Operation). *Given a query $?x \xrightarrow{r} u$, the inverse operation is to find x such that $energy(\mathbf{x}, \mathbf{r}, \mathbf{u}) = 0$, e.g., $\mathbf{x} = \mathbf{u} - \mathbf{r}$ in* `TransE`*. This answers a query $?x \xrightarrow{r} u$.*

For instance, suppose that we compute the energy of a bidirectional 2-hop path query $?c \xleftarrow{r_1} Trump \xrightarrow{r_2} US$ in `TransE`. The inverse operation of the energy is derived in a straightforward manner (see Sec. 3.3). Thus, the energy is computed as $e(\mathbf{e}) = |\mathbf{u} - \mathbf{r2} + \mathbf{r1}|$, where $\mathbf{u}$ is US, and the answers can be C1/C2 close to the vector $\mathbf{e}$. With the operations, we define the energy of a bidirectional path.

Definition 3 (Energy of Bidirectional Path). *Given an h-hop bidirectional path p in a KG, whose left-end is a vertex $u \in \mathcal{V}$ and right-end is a vertex $v \in \mathcal{V}$ with a series of intermediate relations $r_1, \cdots, r_h$, let $\mathbf{x}$ be a vector calculated after a series of regular and inverse operations starting from $\mathbf{u}$ up to r_{h-1}. The energy of the bidirectional path is then defined in* `TransE` *as:*

$$energy(p) = \begin{cases} ||\mathbf{x} + \mathbf{r_h} - \mathbf{v}||, & \textit{if the last edge is} \xrightarrow{r_h} v \\ ||\mathbf{v} + \mathbf{r_h} - \mathbf{x}||, & \textit{if the last edge is} \xleftarrow{r_h} v \end{cases},$$

Note that this bidirectional path energy definition is independent from underlying triple energy definitions. We also test a couple of different triple energy definitions in our experiments. Now we can define the energy of a graph query by using the sum of all the energy of bidirectional paths in the query:

Definition 4 (Energy of a Graph Query). *Let $\mathcal{Q}$ be a pattern query and q be an answer candidate to $\mathcal{Q}$, the energy of the graph query, denoted as $e(q)$, is defined as:*

$$energy(q) = \sum_{p \in paths(q)} energy(p) \tag{2}$$

, where $paths(q)$ returns bidirectional paths of q that are matched to bidirectional path queries of $\mathcal{Q}$.

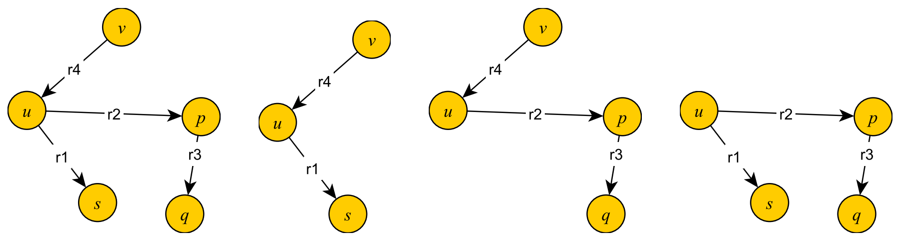
Therefore, answering a graph query $\mathcal{Q}$ is to find a bidirectional paths p in a KG such that $energy(p)$ is minimized.

3.2 Improve the Training Step of KGE Methods

To leverage the new energy definition that supports the regular and inverse operations in `PAGE`, we need to improve the training process of KGE methods. We first sample spanning trees from KGs and decompose each tree into a set of bidirectional paths between two terminal vertices (or degree 1 vertices) in the tree. We then create a set of false paths by altering one terminal vertex of a true path and use both true and false path sets as our training data. This improvement enables KGE methods to learn embeddings of vertices and relations used for answering graph pattern queries.

Sampling Spanning Trees We sample spanning trees from the training sets in Sec. 4.1 by performing the following procedure.

1. Randomly choose a vertex from a KG $\mathcal{G}$.



(a) Spanning Tree (b) Path 1 (c) Path 2 (d) Path 3

Fig. 3: An example spanning tree (a), and its three terminal-to-terminal decompositions (b), (c), and (d).

2. Perform the Join 3(b) of the FFSM [8] e times so that a spanning tree with e edges will be sampled[10].
3. Repeat 1 and 2 until all vertices and edges of the graph $\mathcal{G}$ are covered by at least c different sampled spanning trees.

To ensure a set of comprehensive samples, we utilize the sampling procedure with multiple e values—i.e., we use e up to 4 in our experiments. This method also allows frequently appearing edges in $\mathcal{G}$ to be more sampled than others, which is fair because those edges are more likely to be the part of answers for a graph query. We derive false spanning trees from a true spanning tree by applying the Join 3(b) of the FFSM e times with the edges not in $\mathcal{G}$.

Decomposing Spanning Trees into Bidirectional Paths. Since our method considers each bidirectional paths p from a spanning tree t as a training case, we compute the energy of a path by decomposing the sampled spanning trees. For instance, in Figure 3, we can extract three terminal-to-terminal bidirectional paths from a sampled spanning tree. We utilize the concept introduced in Definition 3 to compute the energy of each case.

Margin-based (Hinge) Loss Function `PAGE` does not only minimize the total energy of training paths decomposed from sampled spanning trees, but also tries to obtain a reasonable energy margin between a training path and false paths derived from the training path. We utilize the same loss function as described in Equation (1) with our own energy definition. Given a training path t^+ and a false path t^- created by randomly modifying one terminal vertex of t^+, t^+'s energy is required to be smaller than that of t^- by a margin of γ. If this is the case for all true and false paths, then the loss function becomes 0, which means a perfect embedding.

[10]The Join 3(b) operation simply appends one random vertex to the terminal position of the current tree such that the extended tree can be also a valid subtree in $\mathcal{G}$

```
Input: KG $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$, Max iteration $max\_iter$, Learning rate $\delta$
Output: Embedding matrix $\mathsf{M}$ for vertices and relations
1 $\mathsf{M} \leftarrow d \times n$ matrix with random initialization
2 $\mathcal{T} \leftarrow$ sampled true spanning trees
3 while $iter \leq max\_iter$ do
4     $\mathcal{T} \leftarrow random\_permute(\mathcal{T})$
5     foreach $\mathcal{T}_{batch} \in minibatch\_split(\mathcal{T})$ do
6         $\mathcal{N}(\mathcal{T}_{batch}) \leftarrow$ sampled false spanning trees from each $t^+ \in \mathcal{T}_{batch}$
7         $\mathsf{M} = \mathsf{M} - \delta \times \nabla_\mathsf{M} \mathcal{L}(\mathcal{T}_{batch}, \mathcal{N}(\mathcal{T}_{batch}))$
8         $\mathsf{M} = \frac{\mathsf{M}}{||M||_F}$
9 return $\mathsf{M}$
```

Algorithm 1: The embedding algorithm used in PAGE

3.3 Infeasible Cases of Existing KGE methods

In this section, we formally prove that some KGE methods cannot answer graph queries because the inverse operation in each method is not unique or cannot defined.

Theorem 1. *The inverse operator of* SME *is not unique.*

Proof. Given a query $?x \xrightarrow{r} u$, $\mathbf{r}$ and $\mathbf{u}$, the inverse operator is defined as finding a vertex x mapped to $?x$ such that $energy(\mathbf{x}, \mathbf{r}, \mathbf{u}) = 0$. In SME, $energy(\mathbf{x}, \mathbf{r}, \mathbf{u}) = g_v(\mathbf{r}, \mathbf{x})^\mathrm{T} \cdot g_u(\mathbf{r}, \mathbf{u})$. Let $X = g_v(\mathbf{r}, \mathbf{x})$ and $Y = g_u(\mathbf{r}, \mathbf{u})$; thus, $energy(\mathbf{x}, \mathbf{r}, \mathbf{u}) = X^\mathrm{T} \cdot Y$. Note that X and Y are both $d \times 1$ column vectors. When $X^\mathrm{T} \cdot Y = 0$, $w \cdot X$, where w is any scalar coefficient, an energy of 0 also arises, which implies that any $\mathbf{x}'$ can be a solution as long as $g_v(\mathbf{r}, \mathbf{x}') = w \cdot X$. There are so many such w that $w \cdot \mathbf{x}'$ can be a solution of the inverse operator.

Theorem 2. *In some variations of* TransE*, the inverse operator cannot be defined or is not computationally desired.*

Proof. Due to space constraints, we sketch a proof. The key idea is: i) to prove the existence (or uniqueness) of the inverse of a generative model (TransG) and ii) to discuss the inverse matrix computation time during the loss minimization (TransH, TransD and so on). For instance, TransG learns multiple vector representations for a relation r. Thus, $energy(v, r, u)$ is a weighted sum of several different energy levels. Each vector representation leads to a different energy level, which can be simply written as $\sum w_i \cdot energy_i(v, r, u)$, where $energy_i(v, r, u)$ is the energy level defined by the i_{th} vector representation of r. Given a query edge $?x \xrightarrow{r} u$, there are many such candidates of $?x$ that the weighted sum equals zero. Thus, the inverse operator solution of TransG also cannot be uniquely defined.

3.4 Embedding Algorithm

Let M be a $d \times n$ embedding matrix, where $n = |\mathcal{V}_G| + |\mathsf{L}|$—i.e., each column of M is an embedding of a vertex or a relation. We use the *projected stochastic gradient*

Database	Vertices	Relations	Training Edges	Testing Queries	Validating Queries
FB15K	14,951	1,345	483,142	50,000	59,071
Nell186	14,463	186	31,134	5,000	5,000

Table 1: Statistics of the *FB15K* and *Nell186* databases

descent (SGD) method described in [7] to compute the M that minimizes the loss function in Eq. 1. In Algorithm 1, we first randomly initialize M (line 1) and sample spanning trees from $\mathcal{G}$ (line 2). In each iteration, we randomly permute sampled spanning trees $\mathcal{T}$ (line 4) and update M w.r.t. the gradient of the loss term (line 7). The SGD computation is efficiently done by various deep learning platforms with the support of GPUs[11]. At the end of each iteration (line 8), we project M onto the unit sphere to prohibit M from being extremely large during iterations.

4 Evaluation

We evaluate `PAGE` in two tasks: 1) factoid query answering and 2) graph query answering. We expect that `PAGE` based on KGE methods outperform in the aforementioned two tasks than the baseline KGE methods.

Baselines. We use `TransE` and `SE` as our baseline methods because they support both regular and inverse operations. Other KGE methods such as `SME` and some variations of `TransE` are excluded because they cannot define unique inverse operations from their energy definitions (see Sec. 3.3). In our experiments, we denote the `TransE` improved by the proposed training process as `PAGE-TransE` and the improved `SE` as `PAGE-SE`.

Experimental Setup. We implement `PAGE` in Python 2.7 with Theano deep learning library[12]. In evaluations, we run `PAGE` on Amazon EC2 instances of type `g2.2xlarge` equipped with an Intel Xeon E5-2670 processor that has eight processor cores, 15GB of RAM, and a Nvidia Tesla GPU with 4GB of video memory and 1,536 CUDA cores.

4.1 Databases and Evaluation Metrics

In this subsection, we discuss our databases and metrics.

Databases. We conducted experiments on datasets from two popular KGs: *FB15K* [3] is a subset of Freebase, and *Nell186* [5] is a subset of NELL containing the most frequent 186 relations. In both KGs, we have well-defined training

[11]We used the Theano [1], one of the most popular deep learning platform.

[12]Theano is one of the most popular deep learning platforms. Optimizing the loss function with the SGD method can be performed efficiently with the support of GPUs.

Database	Metric	Type	`TransE`	`PAGE-TransE`	`SE`	`PAGE-SE`
FB15K	Mean Rank	Micro	181.76	**178.98**	408.69	375.48
		Macro	109.02	**106.09**	412.04	364.24
	Hits@10/100	Micro	43.4% / **76.6%**	**44.2%** / 76.2%	21.9% / 59.2%	22.3% / 59.3%
		Macro	49.2% / **81.3%**	**49.4%** / 80.9%	27.6% / 62.8%	28.9% / 62.8%
Nell186	Mean Rank	Micro	885.54	**784.02**	3412.0	3752.5
		Macro	885.54	**784.02**	4492.2	4736.8
	Hits@10/100	Micro	**41.5% / 74.6%**	38.6% / 72.4%	10.1% / 15.75%	9.2% / 14.5%
		Macro	**41.5% / 74.6%**	38.6% / 72.4%	3.3% / 8.0%	3.0% / 7.1%

Table 2: Mean ranks and Hits@10/100/100 for the factoid query task. (The best values are marked in bold font.)

graphs and factoid testing/validating queries. We sample spanning trees from training graphs and also create random graph pattern queries as follows.

1. Merge training and test sets into one KG.
2. Randomly select a vertex v from the test set.
3. Create z paths by iterating the following steps z times.
   (a) Choose a length in between 2 and 4.
   (b) Randomly select a path of the chosen length starting from v. This path should have at least one edge in the original test set.
4. Convert v and all intermediate vertices of the sampled paths into variables and create a graph query Q.
5. The correct answer to the query Q is v, i.e., we are interested only in finding an entity mapped to the variable converted from v.

The statistics of our datasets are summarized in Table 1.

Metrics. We use the same evaluation metrics as in previous studies: 1) the average rank of the correct answers among the entities sorted in ascending order of energy (mean rank), and 2) the proportion of correct answers ranked in the top 10/100/1000 (Hits@10/100/1000).

4.2 Factoid Query Answering

Table 2 summarizes the results of the factoid query answering task. The best performances are shown in `TransE` cases, and `SE` shows the worse performance than `TransE` for all the datasets across all the metrics. Thus, our discussion focuses on the `TransE` and `PAGE-TransE` cases. In terms of the mean rank, `PAGE-TransE` demonstrates at most 13% better performance than the baseline `TransE`. The performance of `TransE` and `PAGE-TransE` in terms of Hits@10/100 is similar in *FB15K* whereas `TransE` performs slightly better in *Nell186*.

Database	Metric	Type	`TransE`	`PAGE-TransE`	`SE`	`PAGE-SE`
FB15K	Mean Rank	Micro	1150.5	**1088**	7493.5	7514.0
		Macro	2509.9	**2362.8**	7571.4	7933.7
	Hits@100/1000	Micro	18.3% / 56.7%	**25% / 60%**	1.7% / 5.0%	1.7% / 10%
		Macro	19% / 58.7%	**24.3% / 61.7%**	2.0% / 6.0%	2.0% / 7.7%
Nell186	Mean Rank	Micro	38.5	**38**	4803.6	4960.1
		Macro	769.4	**491.1**	5240.3	5431.8
	Hits@100/1000	Micro	64.8% / 89.4%	**66.5% / 94.6%**	15.75% / 31.4%	14.5% / 28.9%
		Macro	60.2% / 80.7%	**65.4% / 87.2%**	7.9% / 23.4%	7.1% / 21.3%

Table 3: Mean ranks and Hits@10/100/100 for the graph task. (The best values are marked in bold font.)

`PAGE` that involves proposed training process did not show the best performance in all the factoid query answering tasks. However, `PAGE` demonstrates similar accuracy in terms of Hit@100 and is better in graph query answering (see Sec. 4.3. In more than 70% to 80% of the testing queries, correct answers are part of the top-100 candidates, which means our approach is generally applicable.

4.3 Graph Query Answering

In Table 3, we summarize the results of the graph query answering task. Since the graph query answering is a more difficult task than the factoid query answering, we use Hits@100/1000 instead of Hits@10. Similar to the factoid query answering results, `SE` exhibits worse performance than `TransE`, thus, our comparison focuses on the `TransE` cases. As expected, `PAGE-TransE` significantly outperforms `TransE` in all cases, which implies that considering of terminal-to-terminal bidirectional paths in the training process enables answering graph queries. In detail, `PAGE-TransE` demonstrates 9% to 28% enhancements for Hits@100 (19% to 24.3% in *FB15K* and 60.2% to 65.4% in *Nell186*) compared to the original `TransE`.

5 Discussion

Complexity Issue of the `PAGE` Query Model Dropping the subgraph isomorphism enables to find latent answers since any vertex can be a candidate answer of a variable. However, considering entire vertices as candidate answers is not computationally preferred. For instance, in the mixed-directional path query $?c \xleftarrow{r_1} ?y \xrightarrow{r_2} ?z \xleftarrow{r_3} US$, the number of candidates can be exponentially increased once we decided to search candidates for the intermediate variables $?y$ and $?z$. Instead, our method excludes intermediate variables in a mixed-directional query path from candidates and computes the energy between $?c$ and US by considering only the relations r_1, r_2, and r_3 (and their directions). This approach decreases computations and enables a lightweight query processing time complexity—i.e., k^n rather than k^m, where k is the number of candidates for a variable, m is the number of all variables, and $n \ll m$ is the number of non-intermediate variables.

Qualitative Comparison with Approximated Graph Query Model Many approximated graph query answering models have been proposed [9, 13]. These models partially ignore the subgraph isomorphism by allowing missing edges in a KG or considering only semantically similar edges. However, there is a case in which the answers from such models cannot be one of the top-rank answers whereas our model ranks any low-energy candidate highly. For instance, in the worst case, suppose that the query is "Who is the athlete who won the U.S. Open against Roger Federer and is a teammate of Andy Murray?". The correct answer is Andy Roddick, however, the following two triples are not contained in the training set of *Nell186* [5]:

$$\textit{Andy Roddick} \xrightarrow{won} \textit{US Open}$$
$$\textit{Andy Roddick} \xrightarrow{isTeammateOf} \textit{Rodger Federer}$$

No existing approximate query model can answer this query correctly because all query edges are not matched for Andy Roddick, however, our PAGE model had listed Andy Roddick as one of the top-20 candidates (more precisely, the 18th candidate in terms of energy) among all the vertices.

6 Conclusion

This paper is the first work that tackles the problem of subgraph matching by utilizing KGE methods. We propose PAGE, a novel query model that enables to answer general graph queries on incorrect or incomplete KGs, which provides a new paradigm of querying KGs. Our work has two contributions to data mining and KGE research: 1) we demonstrated that a graph query (or a complex form of a query) can be answered through KGE methods by decomposing the query into multiple mixed-directional path queries, and 2) we achieved the same performance in simple query answering task and the better performance in graph query answering task with two popular KGs. In evaluations, the performance enhancement is at most 28% compared to the baseline KGE methods.

References

[1] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: A cpu and gpu math compiler in python. In: Proc. 9th Python in Science Conf. pp. 1–7 (2010)

[2] Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation. Machine Learning **94**(2), 233–259 (2014)

[3] Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. In: NIPS. pp. 2787–2795 (2013)

[4] Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning Structured Embeddings of Knowledge Bases. In: AAAI. AAAI Press, San Francisco, USA (2011)

[5] Guo, S., Wang, Q., Wang, B., Wang, L., Guo, L.: Semantically Smooth Knowledge Graph Embedding. In: ACL. pp. 84–94. The Association for Computer Linguistics (2015)

[6] Guu, K., Miller, J., Liang, P.: Traversing knowledge graphs in vector space. In: Empirical Methods in Natural Language Processing (EMNLP) (2015)

[7] Hong, S., Chakraborty, T., Ahn, S., Husari, G., Park, N.: Sena: Preserving social structure for network embedding. In: Proceedings of the 28th ACM Conference on Hypertext and Social Media. pp. 235–244. ACM (2017)

[8] Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraphs in the presence of isomorphism. In: Proc. Third IEEE Int. Conf. Data Mining 2003. pp. 549–552 (Nov 2003). https://doi.org/10.1109/ICDM.2003.1250974

[9] Khan, A., Wu, Y., Aggarwal, C.C., Yan, X.: Nema: fast graph search with label similarity. In: Proc. of the 39th international conference on Very Large Data Bases. pp. 181–192 (2013)

[10] Neo4j: The worlds leading graph database (2017)

[11] Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web (Preprint), 1–20 (2016)

[12] Perozzi, B., Al-Rfou', R., Skiena, S.: DeepWalk: online learning of social representations. In: KDD. pp. 701–710. ACM (2014)

[13] Pienta, R., Tamersoy, A., Tong, H., Chau, D.H.: Mage: Matching approximate patterns in richly-attributed graphs. In: BigData Conference. pp. 585–590 (2014)

[14] Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (2008)

[15] Ren, X., Wu, Z., He, W., Qu, M., Voss, C.R., Ji, H., Abdelzaher, T.F., Han, J.: Cotype: Joint extraction of typed entities and relations with knowledge bases. In: Proceedings of the 26th International Conference on World Wide Web. pp. 1015–1024. Geneva, Switzerland (2017). https://doi.org/10.1145/3038912.3052708

[16] Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23**(1), 31–42 (Jan 1976). https://doi.org/10.1145/321921.321925

[17] Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1225–1234. ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939753