On Integrating Knowledge Graph Embedding into SPARQL Query Processing

Hyunjoong Kang[‡], Sanghyun Hong[*], Kookjin Lee[*], Noseong Park[†], Soonhyun Kwon[‡]
Electronics and Telecommunications Research Institute[‡], Daejeon, Korea
University of Maryland[*], College Park, USA
University of North Carolina[†], Charlotte, USA
kanghj@etri.re.kr, {shhong,klee}@umd.edu, npark2@uncc.edu, kwonshzzang@etri.re.kr

***Abstract*—SPARQL is a standard query language for knowledge graphs (KGs). However, it is hard to find correct answer if KGs are incomplete or incorrect. Knowledge graph embedding (KGE) enables answering queries on such KGs by inferring unknown knowledge and removing incorrect knowledge. Hence, our long-term goal in this line of research is to propose a new framework that integrates KGE and SPARQL, which opens various research problems to be addressed. In this paper, we solve one of the most critical problems, that is, optimizing the performance of nearest neighbor (NN) search. In our evaluations, we demonstrate that the search time of state-of-the-art NN search algorithms is improved by 40% without sacrificing answer accuracy.**

***Keywords*-SPARQL Query Processing; Knowledge Graph Embedding; Nearest Neighbor Searching**

I. Introduction

SPARQL is a standard language for querying knowledge graphs (KGs). It finds answers based on existing knowledge in KGs. However, it is hard to expect complete answers from SPARQL if underlying KGs are incomplete or incorrect. Thus, the ultimate goal of our research is to make the framework more resilient against such cases.

Recent researches on knowledge graph embedding (KGE) provide a way to infer missing knowledge and to clean incorrect knowledge on KGs [1]–[3]. **To our knowledge, however, no works attempt to integrate KGE and SPARQL to handle the aforementioned problem.** We try to integrate them for the first time. We envision many brand-new applications based on this approach. However, there are multiple problems that should be solved to completely combine them. ***In this Work-In-Progress paper, we narrow down our scope to solve the most critical problem — i.e., searching nearest neighbors (NNs) in an embedding space.*** We propose a new loss function that maximizes the pruning effects of existing state-of-the-art NN search algorithms while preserving the inference accuracy. It is well-known that the NN search in a high-dimensional space is notoriously hard and has been an open problem for the last few decades. Thus, our solution is generating embeddings that are suitable for the NN search algorithms rather than proposing a new NN search algorithm. We do not consider approximated NN search methods because they do not lead to correct answers after embedding KGs.

Our experiments show that embeddings generated by our loss function can decrease the NN search response time by at most 40%.

† Corresponding author; This work was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIP) [No. CRC-15-05-ETRI].

II. Related Work and Preliminaries

In a KG, relationships between entities are represented by triples (v, r, u), where v and u are entities (or vertices) and r is a relation (or edge label). For example, the knowledge "Trump is the president of the USA" can be expressed as a triple (Trump, isPresidentOf, USA). A collection of such triples compose a huge connected graph denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathsf{L})$, where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is a set of edges (or relationships), and each edge is labeled by a relation $r \in \mathsf{L}$.

One important application of KGs is question answering (QA). In the previous example, Trump should be returned to the question (or query) "Who is the president of the USA?". However, the answering task will be challenging if the exact answer does not exist in $\mathcal{G}$.

Recent works about KGE [1]–[10] show the possibility to accomplish the challenging task. KGE maps vertices and relations onto a d-dimensional vector space and infers missing triples. `TransE` is the most popular and pioneering method. `TransE` measures the *energy* of a triple that indicates the uncertainty (or error) of the triple — i.e., the higher the energy is, the larger the uncertainty (or error) is. Hereinafter, we use a bold character $\mathbf{v}$ (resp., $\mathbf{r}$) to denote a $d \times 1$ column vector representation of a vertex v (resp., a relation r).

Definition 1 (Energy): Given a triple $t = (v, r, u)$, the energy of a triple $e(t)$ is defined as $e(v, r, u) = \|\mathbf{v} + \mathbf{r} - \mathbf{u}\|_{\ell_1 \text{ or } \ell_2}$ in `TransE`

Example 1 (TransE): In Figure 1, v, r, and u (on the left side) are mapped into the 3-dimensional embedding space (on the right side). $\mathbf{v} + \mathbf{r}$, the blue colored vertex, is the predicted vector of v's neighbors connected through r. In this case, the energy of the triple (v, r, u), denoted as $\|\mathbf{v} + \mathbf{r} - \mathbf{u}\|$, indicates the distance between the predicted vector and $\mathbf{u}$ in the space.

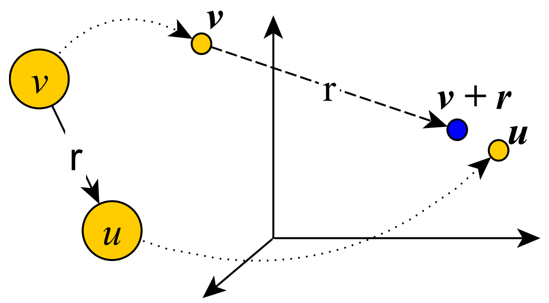


Figure 1: An example `TransE` embedding. The blue vertex $\mathbf{v}+\mathbf{r}$ is the prediction by `TransE`, and $\mathbf{u}$ is the actual answer.

KGE learns vertex and relation vector representations (or embeddings) by minimizing the following max-margin (hinge) loss function:

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

where $\mathcal{E}$ is a set of triples in $\mathcal{G}$ and $\mathcal{N}(t^+)$ is a set of negative (or false) triples derived from a true triple $t^+ \in \mathcal{E}$. $t^- \in \mathcal{N}(t^+)$ differs from t^+ in only one vertex. For instance, two negative triples, (Trump, isPresidentOf, UK) and (Bill Gates, isPresidentOf, USA), can be derived from the true triple (Trump, isPresidentOf, USA).

III. Integrating KGE and SPARQL

We introduce the basic concept of SPARQL query processing, followed by the discussion about the sketch of the integrated framework.

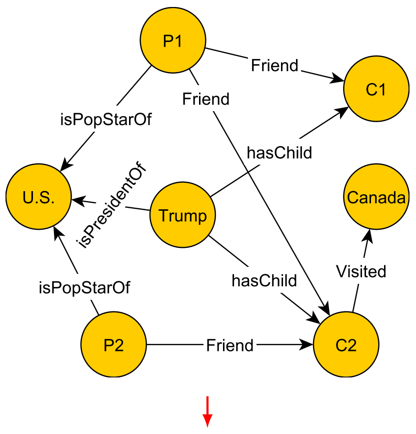
A. *SPARQL Query Processing.*

Triples are typically stored in a three-column table where each column represents sources, relations, or targets (see Figure 2), although there exist some different types of implementations. To reduce response time, the table is usually indexed by the B$^+$-tree algorithm. Let us execute the following SPARQL Query 1 against the KG in Figure 2.

```
SELECT ?p ?c ?s
WHERE {
  ?p isPresidentOf U.S. .
  ?p hasChild ?c .
  ?s isPopStarOf U.S. .
  ?s isFriendOf ?c .
  ?c visited Canada .
}
```

SPARQL Query 1: ?p, ?c, and ?s are variables and should be substituted with real vertices.

The query finds answers about a child $?c$ of the U.S. President $?p$ who has a U.S. pop star $?s$ as a friend and visited Canada. One valid answer is clearly $\{?p = \text{Trump}, ?c = \text{C2}, ?s = \text{P1}\}$. Note that C1 cannot be an answer because C1 did not visit Canada.



Source	Relation	Target
Trump	hasChild	C1
Trump	isPresidentOf	U.S.
⋮	⋮	⋮

Figure 2: An example of knowledge graph and its three-column-table representation.

However, let us assume that C1 had actually visited, but the triple is mistakenly missing. C1 should be an answer in this case, but the existing SPARQL query processing method cannot find it. Thus, we use KGE methods to infer the missing knowledge and attempt to answer the SPARQL query.

B. *Sketch of the Integrated Framework.*

In the integrated framework, KGE inserts candidates for each variable, e.g., $?c$, $?p$, and $?s$ in the previous example, in addition to existing candidates. For this, our framework needs a spatial DBMS and NN search since the energy is defined as the distance in the embedding space (see Example 1). The advantage of our framework is twofold: 1) the framework is different from inexact (or approximate) graph query answering [11], [12] because answers returned from our method are characterized by strong confidence and are not approximated, and 2) we can still use the advanced query features in SPARQL in conjunction with the inference capability.

IV. Proposed KGE Improving Nearest Neighbor Search

The main challenge in integrating KGE methods into SPARQL query processing is to optimize the NN search performance. Note that even if there have been remarkable accomplishments in optimizing the NN search [13], [14], it has been remained as a long-standing open problem.

Given a graph query, our framework first needs to check energy levels and find nearest neighbors in the embedding space. For this, the NN search is required, hence, optimizing the NN search performance can improve the

Databases	Method	KD Tree	Ball Tree	FNNS	OBFS	Mean Rank	Hits@10
FB15K	`TransE` (original)	0.754 sec	0.65 sec	0.703 sec	0.67 sec	197.7	44.61 %
	`TransE` ($\beta = 16$)	**64.7 %**	**68.4 %**	**60.5 %**	**59.7 %**	197.4	**45.25 %**
	`SE` (original)	0.901 sec	0.756 sec	0.661 sec	0.965 sec	**53.0**	25.17 %
	`SE` ($\beta = 8$)	97.3 %	96.9 %	97.2 %	95.9 %	**53.0**	25.18 %
WN18	`TransE` (original)	0.546 sec	0.437 sec	0.446 sec	0.378 sec	337.61	88.93%
	`TransE` ($\beta = 4$)	77.8 %	**82.8 %**	**88.1 %**	**81.8 %**	328.76	89.25%
	`TransR` (original)	0.516 sec	0.61 sec	1.012 sec	1.132 sec	**232.1**	92.31%
	`TransR` ($\beta = 16$)	**76.3 %**	107.3 %	94.2%	93.9 %	**232.1**	**92.34%**

Table I: Average response time of NN search algorithms and query answering accuracy. We show absolute response time in seconds for baseline methods (marked with "original") and relative response time in percentages for the proposed method. Note that the best values are indicated in bold.

overall performance of our framework. Thus, we focus on improving the NN search performance in the rest of this section. As mentioned earlier, we do not design a new NN search algorithm but redesign the loss function used by KGE methods to generate vector representations that can maximize the pruning effects of existing NN search algorithms.

Redesign the Loss Function.: Our solution is to re-design the loss function used for KGE methods to generate vector representations that improve the NN search performance. We propose the following loss term:

$$\mathcal{L}_2 = \mathcal{L} + \alpha \|\mathsf{C} - \mathsf{C}_\mathsf{M}\|_\mathcal{F}, \tag{2}$$

where $\mathcal{L}$ is the loss function in Equation (1), C is the ideal covariance matrix, M is the $d \times |\mathcal{V}|$ matrix containing column vector representations of vertices, C_M is the $d \times d$ covariance matrix of M, and $\|\cdot\|_\mathcal{F}$ denotes the Frobenius matrix norm. The covariance matrix C_M is calculated as follows:

$$\mathsf{C}_M = \mathbb{E}[(\mathsf{M} - \mathbb{E}[\mathsf{M}])(\mathsf{M} - \mathbb{E}[\mathsf{M}])^\intercal]. \tag{3}$$

The key idea behind our loss redesign is that the covariance of the learned matrix M should have a certain ideal form C such that the vector representations (or embeddings) are well-spread in the early dimensions of the embedding space. Given a query point, most NN search algorithms depend on the strategy that prunes a point if its distance from the query point is greater than the distance between the query point and a temporary nearest neighbor. In such algorithms, the distance calculation between two points starts from the first dimension to make the pruning more efficient. For instance, this strategy is highlighted in red in the optimized brute-force search [13] (or the linear projection search [14]) shown in Algorithm 1. Starting from the first dimension (line 5), the algorithm incrementally updates the distance (line 6). If the intermediate distance is greater than the distance from a temporary nearest point, then the point is pruned without further updates (line 7). Thus, if the learned embeddings are

```
Input: A search point $\mathbf{p}$
Output: The nearest neighbor $\mathbf{x}$ of $\mathbf{p}$
1  $\mathbf{x} \leftarrow$ a random vertex in $\mathcal{V}$
2  $min\_dist \leftarrow \|\mathbf{p} - \mathbf{x}\|_1$
3  foreach vertex $v \in \mathcal{V}$ do
4      $dist \leftarrow 0$
5      for $1 \leq i \leq d$ do
6          $dist \mathrel{+}= |\mathbf{p_i} - \mathbf{v_i}|$ or $|\mathbf{p_i} - \mathbf{v_i}|^2$
7          if $dist \geq min\_dist$ then break
8      if $dist < min\_dist$ then
9          $\mathbf{x} \leftarrow v$
10         $min\_dist \leftarrow dist$
11 return $\mathbf{x}$
```

Algorithm 1: Optimized Brute-Force Search (OBFS) seeks the nearest neighbor x of a point $\mathbf{p}$ in terms of the L1 or squared L2 norm. (this simple algorithm is faster than other methods (see Sec V) and is easy to parallelize.)

well-scattered over the space in the early dimensions, the efficiency of the pruning process will be maximized.

Ideal Covariance Matrix: The ideal covariance matrix should be the diagonal matrix that satisfies the following conditions:

1) All non-diagonal elements should be zero to ensure that all d dimensions are independent of each other.
2) The diagonal elements should be non-zero and decreasing exponentially as a function of the index number i; let c_i, $1 \leq i \leq d$, be the i-th diagonal element of C, we utilize the *squared exponential function* to generate the ideal diagonal elements:

$$c_i = exp(-i^2/\beta^2), \tag{4}$$

where β is a scaling parameter. This ensures that the earlier dimensions are well-scattered because c_i denotes the variance of the i-th row vector of M.

Our idea is closely related to principle component analysis (PCA) in that both assume that there exist principle components of the embedding space and the early dimensions make the points mostly distinguishable. However, there is the key difference that we fix the principle components in an ideal form in Equation (4) and let the covariance matrix C_M close to it whereas PCA searches does not assume any ideal forms. We found in our experiments that the term added to our loss function weighed by α in Equation (2) can be regarded as a regularizer. Even if the term is not designed to have a regularization effect, our experimental results report that the query answering accuracy can be slightly increased in certain cases — we do not claim this point strongly because it is not our main focus. However, this is the most desirable scenario, where our new loss function improves not only the answering accuracy but also reduces the NN search response time. Thus, our new loss function $\mathcal{L}_2$ can be seen as a loss function based on $\mathcal{L}$ with the covariance-based regularizer.

V. Evaluation

We evaluate our method with four search algorithms: KD tree, Ball tree, OBFS, and FNNS [13].

We first calculate vector representations with various KGE methods and utilize them as inputs to the NN search algorithms. Seven KGE methods (`TransE`, `SME`, `SE`, `TransR`, `TransD`, `TransH` and `TransG`) were tested considering their popularity. Three representative cases (`TransE`, `TransR`, and `SE`) are reported in Table I. `SME` shows similar patterns to `SE`. `TransD` and `TransH` are omitted because `TransR` results are similar to their results. `TransG` is also improved a lot by the proposed loss function but its improvement ratio is close to that of `TransE`. Thus, we selected the three representative cases to show.

Datasets.: We use two standard evaluation KGs: FB15K and WN18 that have thousands of validation and testing questions [1].

Metrics: We measure the search response time in seconds and the inference accuracy in terms of mean rank and Hits@10. The mean rank is the average rank of correct answers among candidates sorted in ascending order of energy. Hits@10 is the ratio of test questions where their answers are one of the top-10 lowest-energy candidates. Low mean rank values are desired whereas Hits@10 should be high.

Results: Our results are summarized in Table I. For all the experiments, we set $\alpha = 1/|\mathcal{V}|$ and find β through validation. Algorithms marked with "original" show its original response time when trained with the original loss in Equation (1). For our method, we show the relative response time in comparison with the original response time. In terms of response time, `TransE` mostly achieved the best performance with the Ball tree, FNNS and OBFS when $\beta = 16$ for FB15K and $\beta = 4$ for WN18. However, `TransR` demonstrated the best with the KD tree when $\beta = 16$ for WN18. For the accuracy, we found that `TransE` is improved whereas `SE` and `TransR` are not sensitive to the loss function modifications. Interestingly, our method achieved the mean-rank improvement from 337.61 to 328.76 for WN18 in the `TransE` case, which is non-trivial.

VI. Conclusion

This paper has tackled one of the most significant problems in integrating KGE with SPARQL. Since the embeddings obtained from KGE methods are stored in spatial DBMSs, the B$^+$-tree indexing in SPARQL query processing cannot be used. Thus, we have to utilize the NN search methods in searching answer candidates, which essentially introduces a critical performance bottleneck. To reduce the performance issue, we propose a new loss function for KGE methods that supports the NN search algorithms' pruning idea. In evaluations, we demonstrate that our method significantly improves both the response time and the accuracy in terms of mean-rank and Hits@10.

References

[1] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation." *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[2] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating Embeddings for Modeling Multi-relational Data." in *Proceedings of NIPS'13*, 2013, pp. 2787–2795.

[3] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning Structured Embeddings of Knowledge Bases." in *Proceedings of AAAI'11*, San Francisco, USA, 2011.

[4] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proceedings of AAAI'16*, ser. AAAI'16. AAAI Press, 2016, pp. 985–991.

[5] H. Yoon, H. Song, S. Park, and S. Park, "A translation-based knowledge graph embedding preserving logical property of relations," in *Proceedings of NAACL'16*, 2016, pp. 907–916.

[6] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of AAAI'15*, ser. AAAI'15, 2015, pp. 2181–2187.

[7] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of ACL'15*, 2015, pp. 687–696.

[8] H. Xiao, M. Huang, Y. Hao, and X. Zhu, "Transg : A generative mixture model for knowledge graph embedding." *CoRR*, vol. abs/1509.05488, 2015.

[9] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *Proceedings of AAAI'14*. AAAI Press, 2014, pp. 1112–1119.

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

[11] L. Zou, L. Chen, and M. T. Özsu, "Distance-join: Pattern match query in a large graph database," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 886–897, 2009.

[12] M. Mongiovi, R. Di Natale, R. Giugno, A. Pulvirenti, A. Ferro, and R. Sharan, "Sigma: a set-cover-based inexact graph matching algorithm," *Journal of bioinformatics and computational biology*, vol. 8, no. 02, pp. 199–218, 2010.

[13] Y. Hwang, B. Han, and H. K. Ahn, "A fast nearest neighbor search algorithm by nonlinear embedding," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3053–3060.

[14] Y. Hel-Or and H. Hel-Or, "Real-time pattern matching using projection kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1430–1445, Sept 2005.