Graph Embedding based Query Construction over Knowledge Graphs

Ruijie Wang[1,2], Meng Wang[3*], Jun Liu[4,2], Siyu Yao[1,2], Qinghua Zheng[1,2]
1. Shaanxi Province Key Laboratory of Satellite and Terrestrial Network Tech. R&D,
Xi'an Jiaotong University, Xi'an, Shaanxi, China
2. School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi, China
3. School of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu, China
4. Guang Dong Xi'an Jiaotong University Academy, Shunde, Guangdong, China
xjdwrj@stu.xjtu.edu.cn, wangmengsd@outlook.com, {liukeen,qhzheng}@xjtu.edu.cn, cheryl@stu.xjtu.edu.cn

***Abstract*—Graph-structured queries provide an efficient way to retrieve the desired data from large-scale knowledge graphs. However, it is difficult for non-expert users to write such queries, and users prefer expressing their query intention through natural language questions. Therefore, automatically constructing graph-structured queries of given natural language questions has received wide attention in recent years. Most existing methods rely on natural language processing techniques to perform the query construction process, which is complicated and time-consuming. In this paper, we focus on the query construction process and propose a novel framework which stands on recent advances in knowledge graph embedding techniques. Our framework first encodes the underlying knowledge graph into a low-dimensional embedding space by leveraging the generalized local knowledge graphs. Then, given a natural language question, our framework computes the structure of the target query and determines the vertices/edges which form the target query based on the learned embedding vectors. Finally, the target graph-structured query is constructed according to the query structure and determined vertices/edges. Extensive experiments were conducted on the benchmark dataset. The results demonstrate that our framework outperforms several state-of-the-art baseline models regarding effectiveness and efficiency.**

***Index Terms*—knowledge graph, query construction, knowledge graph embedding, natural language question answering**

I. Introduction

In recent years, an increasing number of large-scale knowledge graphs (KGs), e.g., DBpedia [1] and Wikidata [2], have been published on the Web. Structured query languages, e.g., SPARQL [3] and GraphQL [4], are widely used to access the knowledge contained in KGs. And the structured query can be represented as the graph consisting of vertices, edges, and variables. For example, we illustrate the graph-structured query of the natural language question (NLQ) "*Which actor played in the movies directed by Tim Burton?*" in Fig. 1, where vertices (e.g., *Tim Burton*, *Actor*) and variables (e.g., *?movies*, *?actor*) are linked by edges (e.g., *director*, *type*). Issuing such queries requires users to be precisely aware of the structure and schema of the underlying KG. In order to hide the complexity of query languages, numerous models [5]–[9] have been proposed to construct the graph-structured queries of given NLQs automatically.

*Meng Wang is the corresponding author.

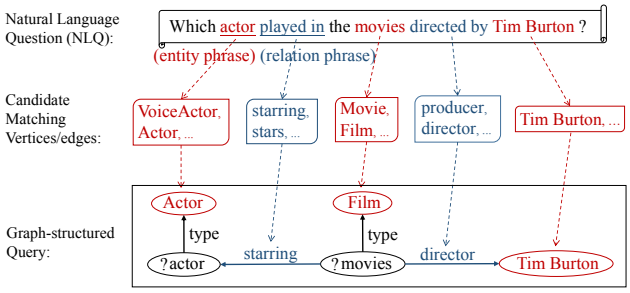


Fig. 1. General query construction process of the example NLQ.

Challenges: In general, constructing the graph-structured query of an NLQ mainly includes two tasks: determining the vertices/edges which form the target query and deducing the structure of the target query. We face the following challenges during the construction process:

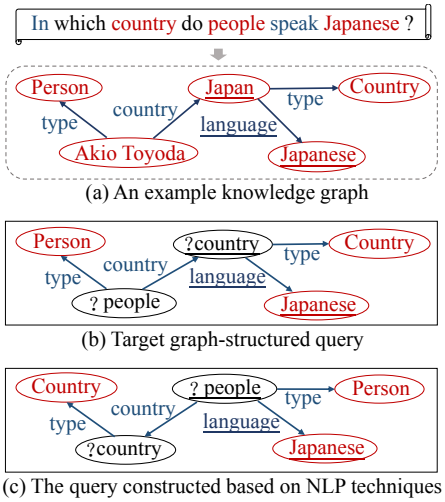
1) To determine the vertices/edges, we need to extract entity/relation phrases from the given NLQ and map them to the matching vertices/edges in the underlying KG. For the NLQ "*Which actor played in the movies directed by Tim Burton?*", the extraction and mapping process is illustrated in Fig. 1. A challenge during this process is that an entity/relation phrase may have multiple candidate matching vertices/edges, and it is hard to select the optimal one. Taking the entity phrase "movies" as an example, both *Movie* and *Film* are plausible candidate matching vertices. Some models [8], [9] determine the optimal candidate vertices/edges during the generation of target queries/answers, which is inefficient since the search space would be expanded at the same time. Other methods [5], [7] prune the candidate vertices/edges before the query generation, but they may filter out potential optimal candidates unexpectedly, which leads to ineffectiveness.

2) To deduce the structure of the target query, most existing models [6]–[9] employ natural language processing (NLP) techniques. However, these models cannot address the "*semantic gap*" between NLQs and KGs which refers to that KGs organize structured information differently from what one can deduce from natural language expressions [10]. Let us consider another NLQ "*In which country do people speak Japanese?*"

978-1-5386-9125-0/18/$31.00 ©2018 IEEE
DOI 10.1109/ICBK.2018.00009

IEEE computer society

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.



(a) An example knowledge graph

(b) Target graph-structured query

(c) The query constructed based on NLP techniques

Fig. 2. Example of the *semantic gap* between NLQs and KGs.

which is posed over the KG illustrated in Fig. 2(a). The target structured query of this NLQ is shown in Fig. 2(b). However, NLP technique-based models would generate the structured query shown in Fig. 2(c) since there is an obvious semantic relation "speak" between "people" and "Japanese" from the NLP point of view. These methods cannot infer the relation between "country" and "Japanese", and the constructed query in Fig. 2(c) cannot be evaluated over the underlying KG to obtain answers.

Solutions: In this paper, we focus on the above challenges and propose a novel graph embedding-based framework to construct the graph-structured queries of NLQs automatically. Our framework includes the following processes:

Firstly, the underlying KG is encoded into a low-dimensional embedding space based on the generalized local knowledge graphs (defined in Section II-B). Then, given an NLQ, each entity/relation phrase of the NLQ is mapped to a set of candidate matching vertices/edges in the KG. With the learned embedding vectors, the candidate vertex/edge sets are utilized to compute the structure of the target query. Finally, based on the determined query structure, the optimal matching vertices/edges are selected from the candidate sets, and the target query is generated.

Contributions: In a nutshell, our work makes the following contributions.

1) We propose a novel graph embedding-based framework to construct graph-structured queries of NLQs automatically.
2) We propose a translation-based embedding method which leverages the generalized local knowledge graphs to make the learned embedding vectors applicable to the query construction.
3) We propose an effective and efficient approach to deduce the structure of the target query and determine the optimal matching vertices/edges based on the learned embedding vectors.
4) We conducted extensive experiments on the benchmark dataset to evaluate our framework. The results show that our method outperforms several state-of-the-art baselines regarding both effectiveness and efficiency.

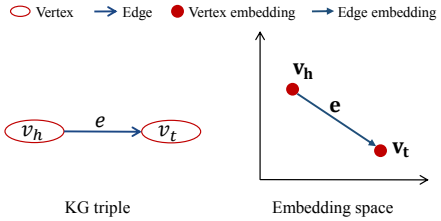


Fig. 3. Illustration of the translation mechanism of TransE.

Organization: The remainder of this paper is organized as follows: Section II introduces our proposed framework in detail. The evaluation of the framework is reported in Section III. Related work is discussed in Section IV. Finally, conclusions and future work are presented in Section V.

II. Proposed Framework

In this section, we first introduce the notations employed in this paper and then elaborate on our framework.

KG Triple and KG. Let $\mathcal{V}$ be a set of vertices (e.g., ⟨Tim Burton⟩[1]), $\mathcal{E}$ be a set of edges (e.g., ⟨director⟩). A KG triple is denoted as (v_h, e, v_t), where $v_h, v_t \in \mathcal{V}$ and $e \in \mathcal{E}$. For example, (⟨Batman⟩, ⟨director⟩, ⟨Tim Burton⟩) is a KG triple. A knowledge graph (KG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a finite set of KG triples.

Entity Vertex and Class Vertex. We divide vertices in the KG into two categories: the entity vertex $v_e \in \mathcal{V}$ represents a specific entity, and the class vertex $v_c \in \mathcal{V}$ represents the type of entity vertices. For example, given the KG triple (⟨Batman⟩, ⟨type⟩, ⟨Film⟩), we can deduce that ⟨Batman⟩ is an entity vertex, and its class vertex is ⟨Film⟩.

NLQ and Entity/Relation Phrase. We denote a given natural language question (NLQ) as $\mathcal{Q}$. The entity phrase (e.g., "actor", "movies", and "Tim Burton") and the relation phrase (e.g., "played in" and "directed by") in $\mathcal{Q}$ are denoted as ent and rel, respectively.

Graph-Structured Query. Let $\mathcal{V}_v$ be a set of variables[2], where each variable $v_v \in \mathcal{V}_v$ is distinguished from the vertices by a leading question mark symbol, e.g., ⟨?movies⟩. A triple pattern is similar to the KG triple but allows the use of variables, e.g., (⟨?movies⟩, ⟨director⟩, ⟨Tim Burton⟩). We define the graph-structured query $\mathcal{G}_\mathcal{Q}$ as a finite set of triple patterns.

KG Embedding. KG embedding methods encode a KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into the embedding space, where the vertex $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ are respectively represented by the vertex embedding $\mathbf{v}$ and edge embedding $\mathbf{e}$.

Translation Mechanism. Our framework adopts the translation mechanism of TransE [12] which represents edges in

[1]We use the term enclosed by a pair of angle brackets to represent a vertex/edge of the KG.

[2]In this paper, we only consider the variables on vertices since the variables on edges rarely appear in real-world structured queries [11].

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.

the KG as translation operations from head vertices to tail vertices in the embedding space. For example, given a KG triple $(v_h, e, v_t) \in \mathcal{G}$, we learn the embedding vectors $\mathbf{v_h}$, $\mathbf{e}$, and $\mathbf{v_t}$ which hold $\mathbf{v_h} + \mathbf{e} \approx \mathbf{v_t}$, as illustrated in Fig. 3.

A. Overview of Proposed Framework

Our framework constructs the graph-structured query of a given NLQ by three modules: phrase mapping, query structure computing, and query generation. We depict an overview of the proposed framework in Fig. 4.

In the first module, our framework maps each entity/relation phrase of the NLQ to a set of candidate matching vertices/edges. We denote the candidate vertex set and the candidate edge set as C_v and C_e, respectively. For example, the candidate vertex set of the entity phrase "actor" (ent_1) is $C_v^1 = \{v_1^1, v_2^1, ...\}$ (i.e., {⟨VoiceActor⟩, ⟨Actor⟩, ...}), as illustrated in Fig. 4(a).

In the second module, our framework utilizes the learned embedding vectors to compute the query structure based on the translation mechanism, as shown in Fig. 4(b). To this end, embedding vectors of the vertices/edges in the same candidate set should be close to each other. For example, we require that $\mathbf{v_1^1} \approx \mathbf{v_2^1}$ since ⟨VoiceActor⟩ (v_1^1) and ⟨Actor⟩ (v_2^1) are in the same candidate set, as shown on the left of Fig. 4(b). Then, we represent each candidate set as a mean embedding vector of the vertices/edges in it and compute the query structure based on the mean embedding vectors of all candidate sets. The query structure is represented by the graph consisting of candidate vertex/edge sets. For example, the computed query structure of the example NLQ is $\{(C_v^2, C_e^1, C_v^1), (C_v^2, C_e^2, C_v^3)\}$, as shown on the right of Fig. 4(b).

In the third module, our framework selects the optimal matching vertex/edge from each candidate set to generate the final structured query. This process is also based on the learned embedding vectors. For instance, the final structured query of the example NLQ is {(⟨?movies⟩, ⟨director⟩, ⟨Tim Burton⟩), (⟨?movies⟩, ⟨starring⟩, ⟨?actor⟩), (⟨?actor⟩, ⟨type⟩, ⟨Actor⟩), (⟨?movies⟩, ⟨type⟩, ⟨Film⟩)}, as illustrated in Fig. 4(c).

B. KG Embedding Learning

As introduced above, the learned embedding vectors of the vertices/edges in the same candidate set should be close to each other, and the translation mechanism should be maintained by our embedding method. In addition, since we represent the variables (e.g., ⟨?movies⟩ and ⟨?actor⟩) of the final structured query by their class vertices (e.g., ⟨Film⟩ and ⟨Actor⟩) during the query construction process, as shown on the left of Fig. 4(c), we require that the learned embedding vectors can be utilized to measure the relation between two class vertices (e.g., ⟨Actor⟩ and ⟨Film⟩) and the relation between a class vertex and an entity vertex (e.g., ⟨Film⟩ and ⟨Tim Burton⟩). However, except the declaration of the types of entity vertices, e.g. (⟨Batman⟩, ⟨type⟩, ⟨Film⟩), the underlying KG only describes the relations between entity vertices, e.g., (⟨Batman⟩, ⟨starring⟩, ⟨Michael Keaton⟩). Therefore, the existing embedding method cannot be directly adopted in our framework, and we need to consider the information generalized based on existing KG triples during the embedding learning. For example, given the KG triples (⟨Batman⟩, ⟨starring⟩, ⟨Michael Keaton⟩), (⟨Batman⟩, ⟨type⟩, ⟨Film⟩), and (⟨Michael Keaton⟩, ⟨type⟩, ⟨Actor⟩), we can deduce the generalized KG triple (⟨Film⟩, ⟨starring⟩, ⟨Actor⟩).

In this section, we propose a novel embedding method which leverages the generalized local knowledge graphs (GL-KGs) of vertices/edges in the underlying KG to learn the required embedding vectors.

Definition II.1 (Local Knowledge Graph)**.** Given a KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the local knowledge graph (L-KG) of the vertex $v \in \mathcal{V}$ is a KG triple set $\mathcal{G}_{\mathcal{L}}(v) = \{(v, e, \hat{v}) | e \in \mathcal{E}, \hat{v} \in \mathcal{V}, (v, e, \hat{v}) \in \mathcal{G}\} \cup \{(\hat{v}, e, v) | \hat{v} \in \mathcal{V}, e \in \mathcal{E}, (\hat{v}, e, v) \in \mathcal{G}\}$. The L-KG of the edge $e \in \mathcal{E}$ is a KG triple set $\mathcal{G}_{\mathcal{L}}(e) = \{(v, e, \hat{v}) | v, \hat{v} \in \mathcal{V}, (v, e, \hat{v}) \in \mathcal{G}\}$.

Definition II.2 (Generalized Local Knowledge Graph)**.** Given a KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the generalized local knowledge graph (GL-KG) of the entity vertex $v_e \in \mathcal{V}$ is the KG triple set $\mathcal{G}_{\mathcal{G}}(v_e) = \{(v_e, e, \hat{v_c}) | (v_e, e, \hat{v_c}) \in \mathcal{G}_{\mathcal{L}}(v_e)\} \cup \{(\hat{v_c}, e, v_e) | (\hat{v_e}, e, v_e) \in \mathcal{G}_{\mathcal{L}}(v_e)\}$, where $\hat{v_c}$ is the class vertex of $\hat{v_e}$. The GL-KG of the class vertex $v_c \in \mathcal{V}$ is the KG triple set $\mathcal{G}_{\mathcal{G}}(v_c) = \{(v_c, e, \hat{v_c}) | (v_e, e, \hat{v_c}) \in \mathcal{G}_{\mathcal{G}}(v_e)\} \cup \{(\hat{v_c}, e, v_c) | (\hat{v_c}, e, v_e) \in \mathcal{G}_{\mathcal{G}}(v_e)\}$, where v_c is the class vertex of v_e. The GL-KG of the edge $e \in \mathcal{E}$ is the KG triple set $\mathcal{G}_{\mathcal{G}}(e) = \{(v_c, e, \hat{v_c}), (v_e, e, \hat{v_c}), (v_c, e, \hat{v_e}) | (v_e, e, \hat{v_e}) \in \mathcal{G}_{\mathcal{L}}(e)\}$, where v_c and $\hat{v_c}$ are the class vertices of v_e and $\hat{v_e}$, respectively.

Taking the entity vertex ⟨Tim Burton⟩ as an example, its L-KG is illustrated in Fig. 5(a), and its GL-KG is illustrated in Fig. 5(b).

We define the conditional probability of v given its GL-KG $\mathcal{G}_{\mathcal{G}}(v)$ as follows:

$$P(v|\mathcal{G}_{\mathcal{G}}(v)) = \frac{\exp(f_1(v, \mathcal{G}_{\mathcal{G}}(v)))}{\sum_{v' \in \mathcal{V}} \exp(f_1(v', \mathcal{G}_{\mathcal{G}}(v)))}, \tag{1}$$

where $f_1(v', \mathcal{G}_{\mathcal{G}}(v))$ is the function that measures the correlation between an arbitrary vertex $v' \in \mathcal{V}$ and $\mathcal{G}_{\mathcal{G}}(v)$. The above equation can be considered as the compatibility between the vertex v and its GL-KG, and it is formulated as a softmax-like representation, which has been validated [13]. Then, we adopts the translation mechanism of TransE to define the function $f_1(v', \mathcal{G}_{\mathcal{G}}(v))$ as follows:

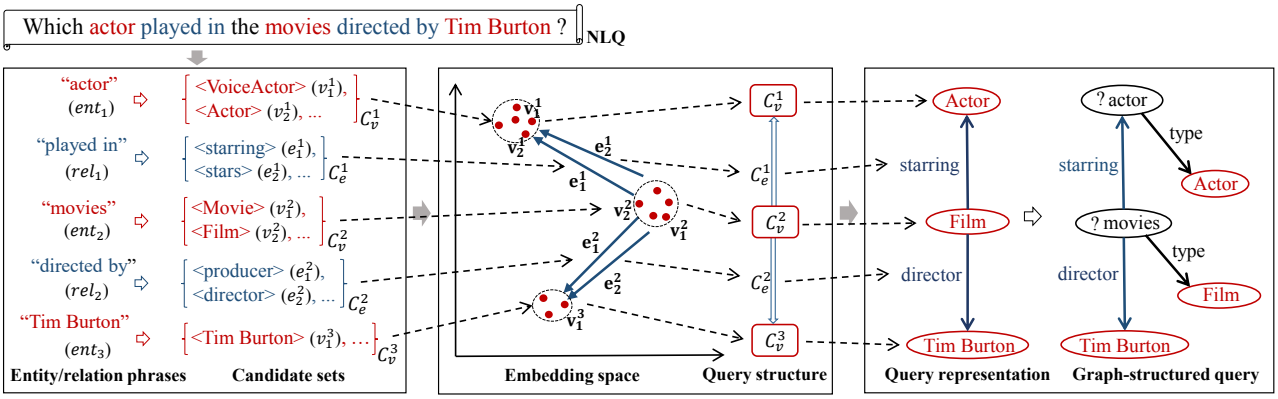
$$f_1(v', \mathcal{G}_{\mathcal{G}}(v)) = -\frac{1}{|\mathcal{G}_{\mathcal{G}}(v)|}\Big(\sum_{(v,e,\hat{v_c}) \in \mathcal{G}_{\mathcal{G}}(v)} \|\mathbf{v'} + \mathbf{e} - \hat{\mathbf{v}}_{\mathbf{c}}\|_2^2 + \sum_{(\hat{v_c},e,v) \in \mathcal{G}_{\mathcal{G}}(v)} \|\hat{\mathbf{v}}_{\mathbf{c}} + \mathbf{e} - \mathbf{v'}\|_2^2\Big), \tag{2}$$

where $|\mathcal{G}_{\mathcal{G}}(v)|$ is the size of $\mathcal{G}_{\mathcal{G}}(v)$.

We learn vertex embeddings by maximizing the joint probability of all vertices in $\mathcal{V}$, which is formulated as follows:

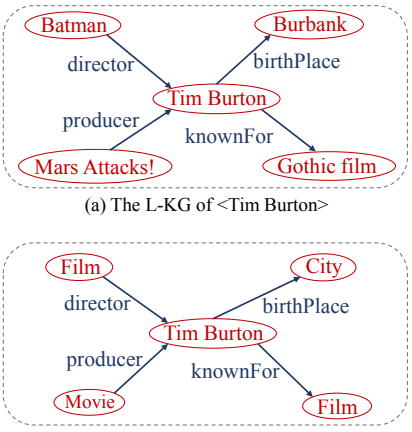
$$O_v = \sum_{v \in \mathcal{V}} \log P(v|\mathcal{G}_{\mathcal{G}}(v)). \tag{3}$$

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.



(a) Phrase Mapping (b) Structure Computing (c) Query Generation

Fig. 4. An overview of our framework.



(a) The L-KG of <Tim Burton>

(b) The GL-KG of <Tim Burton>

Fig. 5. The L-KG and the GL-KG of the entity vertex ⟨Tim Burton⟩.

Analogically, we define the conditional probability of $e \in \mathcal{E}$ given its GL-KG $\mathcal{G}_\mathcal{G}(e)$ as follows:

$$P(e|\mathcal{G}_\mathcal{G}(e)) = \frac{\exp(f_2(e, \mathcal{G}_\mathcal{G}(e)))}{\sum_{e' \in \mathcal{E}} \exp(f_2(e', \mathcal{G}_\mathcal{G}(e)))}, \quad (4)$$

where $f_2(e', \mathcal{G}_\mathcal{G}(e))$ is the function that measures the correlation between an arbitrary edge $e' \in \mathcal{E}$ and $\mathcal{G}_\mathcal{G}(e)$. $f_2(e', \mathcal{G}_\mathcal{G}(e))$ is also formulated based on the translation mechanism:

$$f_2(e', \mathcal{G}_\mathcal{G}(e)) = -\frac{1}{|\mathcal{G}_\mathcal{G}(e)|}\Big(\sum_{(v,e,\hat{v}) \in \mathcal{G}_\mathcal{G}(e)} \|\mathbf{v} + \mathbf{e}' - \hat{\mathbf{v}}\|_2^2\Big), \quad (5)$$

where $|\mathcal{G}_\mathcal{G}(e)|$ is the size of $\mathcal{G}_\mathcal{G}(e)$.

The objective of edge embedding learning is to maximize the joint probability of all edges in $\mathcal{E}$, formulated as follows:

$$O_e = \sum_{e \in \mathcal{E}} \log P(e|\mathcal{G}_\mathcal{G}(e)). \quad (6)$$

We jointly maximize the following objective function to learn the required embedding vectors of vertices and edges.

$$O = \lambda_v O_v + \lambda_e O_e, \quad (7)$$

where λ_v and λ_e are weighting hyper-parameters.

Since the embedding vectors are learned based on the GL-KGs which contain generalized KG triples, e.g., (⟨Film⟩, ⟨starring⟩, ⟨Actor⟩), our embedding method can model the relations between class vertices and entity vertices. Then, considering that vertices/edges in the same candidate set usually share common GL-KGs, their embedding vectors would be close to each other according to (1) and (4). For example, the candidate matching vertices ⟨VoiceActor⟩ (v_1^1) and ⟨Actor⟩ (v_2^1) are linked to common vertices such as ⟨Film⟩ and ⟨CreativeWork⟩ in their GL-KGs, and we have $\mathbf{v_1^1} \approx \mathbf{v_2^1}$. In addition, our embedding method maintains the translation mechanism since we adopt the translation mechanism of TransE in (2) and (5). Note that the learned embedding vectors can be utilized in the following query construction of NLQs without any further modification.

It is impractical to directly compute (1) and (4) due to the large scale of the underlying KG. Therefore, we follow [14] to approximate them based on negative sampling. Taking (1) as an example, it can be approximated by the following equation:

$$P(v|\mathcal{G}_\mathcal{G}(v)) \approx \sigma(f_1(v, \mathcal{G}_\mathcal{G}(v))) \cdot \prod_{v' \in \mathcal{V}_\mathcal{N}^v}^{n} \sigma(f_1(v', \mathcal{G}_\mathcal{G}(v))), \quad (8)$$

where n is the number of negative samples, $\sigma(\cdot)$ is the sigmoid function, and v' is the negative vertex which is obtained by sampling vertices from a uniform distribution over the negative vertex set $\mathcal{V}_\mathcal{N}^v$. For each negative vertex $v' \in \mathcal{V}_\mathcal{N}^v$, we require that $\mathcal{G}_\mathcal{G}(v') \cap \mathcal{G}_\mathcal{G}(v) = \varnothing$.

C. Phrase Mapping

Given an NLQ $\mathcal{Q}$, our framework first extracts the entity phrases $\{ent_1, ent_2, ..., ent_n\}$ and the relation phrases $\{rel_1, rel_2, ..., rel_m\}$ from $\mathcal{Q}$. Then, each entity/relation phrase is mapped to a set of candidate matching vertices/edges in the underlying KG, as illustrated in Fig. 4(a). Candidate vertex sets and candidate edge sets are denoted as $\{C_v^1, C_v^2, ..., C_v^n\}$ and $\{C_e^1, C_e^2, ..., C_e^m\}$, respectively. Since the extraction and mapping process are not the focus of this paper, and they have been well studied in previous works [7],

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.

[8], [15], [16], we directly adopt the existing techniques [15] [8] to obtain high-quality candidate sets.

D. Structure Computing

As we have analyzed in Section II-B, the learned embedding vectors of vertices/edges in the same candidate set are close to each other. Therefore, we compute the mean embedding vector to represent a candidate set in the embedding space. Specifically, the mean embedding vectors of the candidate vertex set C_v and candidate edge set C_e are formulated as follows:

$$\mathbf{C_v} = \frac{1}{|C_v|} \sum_{v \in C_v} \mathbf{v}. \tag{9}$$

$$\mathbf{C_e} = \frac{1}{|C_e|} \sum_{e \in C_e} \mathbf{e}. \tag{10}$$

Intuitively, we can regard each candidate vertex/edge set as an individual vertex/edge and assemble all candidate sets into a graph to represent a possible query structure, as illustrated in Fig. 4(b). We define the structure matrix to denote a possible query structure.

Definition II.3 (Structure Matrix)**.** Given n candidate vertex sets $\{C_v^1, C_v^2, ..., C_v^n\}$ and m candidate edge sets $\{C_e^1, C_e^2, ..., C_e^m\}$. A possible query structure of the target graph is denoted as a structure matrix:

$$M_S = \begin{bmatrix} m_{1,1} & m_{1,2} & ... & m_{1,n} \\ m_{2,1} & m_{2,2} & ... & m_{2,n} \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ m_{n,1} & m_{n,2} & ... & m_{n,n} \end{bmatrix}.$$

For each element $m_{i,j}$ in M_S, if $m_{i,j} = k$, the candidate vertex set C_v^i is linked to C_v^j by the candidate edge set C_e^k. And if $m_{i,j} = 0$, C_v^i is not linked to C_v^j. The structure matrix M_S should satisfy the following constraints:

1) If $i = j$, $m_{i,j} = 0$.
2) If $m_{i,j} > 0$, $m_{j,i} = 0$.
3) The number of non-zero elements in M_S is m.
4) For an integer α, if $0 < \alpha < n + 1$,$\sum_{i=1}^{n} m_{i,\alpha} + \sum_{j=1}^{n} m_{\alpha,j} > 0$.
5) For an integer β, if $0 < \beta < m + 1$, there is an element $m_{i,j} = \beta$ in M_S.

For example, the structure matrix of the query structure shown in Fig. 4(b) is:

$$M_S = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}.$$

Since graph-structured queries of given NLQs usually consist of less than three triple patterns besides type constraints [10], the size of the structure matrix is limited, and it is feasible to construct all possible structure matrices which represent all possible query structures. Then, we can evaluate the possible structure matrices based on the translation mechanism to deduce the optimal query structure, formulated as follows:

$$f_3(M_S) = \sum_{m_{i,j} > 0} \|\mathbf{C_v^i} + \mathbf{C_e^{m_{i,j}}} - \mathbf{C_v^j}\|_2^2. \tag{11}$$

Equation (11) computes the cost of a structure matrix. The structure matrix M_S' which has the minimum cost $f_3(M_S')$ represents the query structure of the target query.

E. Query Generation

In this module, based on the computed query structure, we select the optimal matching vertex/edge from each candidate set and generate the target graph-structured query.

The query structure of the example NLQ is illustrated in Fig. 6(a). We may construct a set of candidate query representations by selecting a candidate vertex/edge from each candidate set, as illustrated in Fig. 6(b). Specifically, a candidate query representation is denoted as a triple set $\mathcal{Q}_\mathcal{R} = \{(v_h^1, e^1, v_t^1), (v_h^2, e^2, v_t^2), ..., (v_h^m, e^m, v_t^m)\}$. We compute the cost of a query representation by the following equation:

$$f_4(\mathcal{Q}_\mathcal{R}) = \sum_{(v_h, e, v_t) \in \mathcal{Q}_\mathcal{R}} \|\mathbf{v_h} + \mathbf{e} - \mathbf{v_t}\|_2^2. \tag{12}$$

The candidate query representation $\mathcal{Q}_\mathcal{R}'$ which has the minimum cost $f_4(\mathcal{Q}_\mathcal{R}')$ is the optimal one. Finally, we generate the target graph-structured query by replacing class vertices in $\mathcal{Q}_\mathcal{R}'$ with variables, as illustrated in Fig. 6(c).

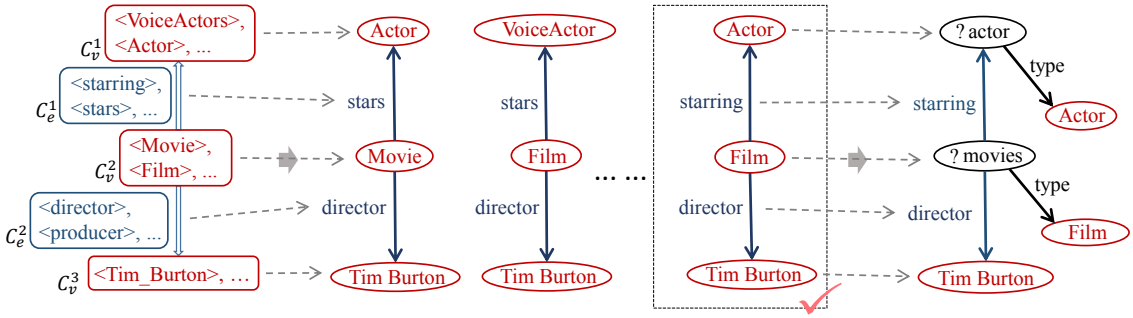
F. Discussion

We discuss two issues which need to be considered during the framework implementation: 1) It is a common scenario where relation phrases are implied in the NLQ. For example, the NLQ "*List actors born in China.*" is usually expressed as "*List Chinese actors.*", where the relation phrase "born in" is implied. We employ the method in [8] to generate candidate edges for implied relation phrases. 2) Other improved translation-based models such as TransH [17] and TransR [18] can also be adopted by modifying (2), (5), (11), and (12) to achieve better embedding performance. However, the framework would be more complicated, and the training time would increase. We will investigate on this part more deeply in the future.

III. Experiments

The graph-structured queries constructed by our framework can be evaluated over KGs to obtain the answers of given NLQs. Therefore, to scrutinize the effectiveness and efficiency of our framework, we compare it with KG-based question answering models including all models participating the first task of QALD-6 [19] and two state-of-the-art models RFF [8] and NFF [8]. All experiments were conducted on a Linux server with an Intel Core i7 3.40GHz CPU and 126GB memory running Ubuntu-14.04.1, and we set the dimension of embedding vectors to 100, $\lambda_v = 0.5$, and $\lambda_e = 0.5$.

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.



(a) Query structure (b) Candidate query representations (c) Graph-structured query

Fig. 6. Generation of the graph-structured query.

A. Dataset

KG Dataset. DBpedia is a large-scale KG which contains structured information extracted from Wikipedia[3]. We employ the version of DBpedia-2015[4] which consists of 6.7M vertices, 1.4K edges, and 583M KG triples.

NLQ Dataset. QALD-6 [19] is the sixth installment of a series of challenges on question answering over KGs. It published 100 test questions over DBpedia for the first task "Multilingual Question Answering" [19]. And the test questions are associated with gold structured queries and answers.

B. Effectiveness Evaluation

In this section, we follow [19] to evaluate the effectiveness of our framework with three metrics: recall, precision, and F-1 measure. The recall refers to the ratio of correct answers obtained by the constructed query over all gold answers. Among all answers obtained by the constructed query, the precision refers to the proportion of correct answers. The F-1 measure is a weighted average between the precision and recall, and it is computed as follows:

$$F-1\ measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \quad (13)$$

We report the evaluation results of our framework, RFF, NFF, and the models [19] participating QALD-6 in Table I, where *Processed* indicates the number of processed NLQs for each model. Note that the recall and precision of each model are computed with respect to the number of processed NLQs, and the F-1 measure is computed with respect to the total number of questions.

We sum up three observations based on Table I: Firstly, our framework is ranked second according to the F-1 measure. Nevertheless, our framework is still the most competitive one since the first-ranked system CANaLI [20] can only answer NLQs expressed by the Controlled Natural Language [20]. Secondly, among the models which processed all test NLQs, our framework achieves the highest recall except CANaLI, which is due to the following reasons: 1) The semantic gap problem between NLQs and KGs is well addressed by our framework since the query structure is computed based on the learned embedding vectors which are essentially the latent representation of the underlying KG. 2) Our framework does not need to prune the candidate sets before the structure deducing and query generation, and the optimal matching vertices/edges are selected during the query generation process. Therefore, the loss of potential optimal matching vertices/edges is effectively avoided. 3) The selected matching vertices/edges of our framework are globally optimal since the selection is based on the computed cost of the entire query. Thirdly, except CANaLI, our framework is ranked third according to the precision. The main reason is that the existing techniques [15] [8] cannot obtain high-quality phrase mapping results for implied relation phrases. Therefore, some constraints in NLQs cannot be translated into structured queries precisely.

TABLE I
RESULTS ON THE NLQ DATASET (TOTAL NUMBER OF QUESTIONS: 100)

	Processed	Recall	Precision	F-1
CANaLI	100	0.89	0.89	0.89
Our framework	**100**	**0.73**	**0.85**	**0.79**
NFF	100	0.70	0.89	0.78
UTQA	100	0.69	0.82	0.75
KWGAnswer	100	0.59	0.85	0.70
RFF	100	0.43	0.77	0.55
NbFramework	**63**	0.85	0.87	0.54
SemGraphQA	100	0.25	0.70	0.37
UIQA(with manual)	**44**	0.63	0.54	0.25
UIQA(without manual)	**36**	0.53	0.43	0.17

C. Efficiency Evaluation

The average time cost of our framework to construct the structured query and obtain answers for a given NLQ is 609.2ms. The average time cost of each module is reported in Table II. The phrase mapping module spends much more time than the other modules due to the large scale of DBpedia which also increases the time cost of the query evaluation process.

[3]https://www.wikipedia.org/
[4]http://wiki.dbpedia.org/develop/datasets

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.

TABLE II
AVERAGE TIME COST OF EACH MODULE

Module	Avg.Time Cost (ms)
Phrase Mapping	475.1
Structure Computing	29.3
Query Generation and Evaluation	104.8

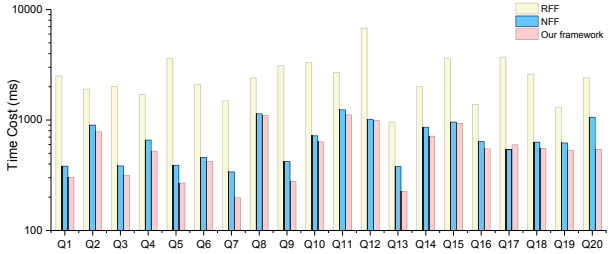


Fig. 7. Time costs of our framework, RFF, and NFF.

The time costs of our framework, RFF, and NFF to answer twenty randomly selected NLQs are illustrated in Fig. 7. Obviously, the time cost of our framework is less than the two baselines. This is reasonable since our framework leverages the learned embedding vectors to construct structured queries. Both the query structure deducing and the selection of matching vertices/edges are performed in the embedding space by numerical calculations. In addition, compared with the subgraph matching process of RFF and NFF, the evaluation of structured queries is more efficient.

In summary, the evaluation results on effectiveness and efficiency demonstrate that our framework can facilitate to construct the graph-structured queries of given NLQs.

D. Failure Analysis

In this section, we analyze the failure causes of our framework. Given an NLQ, if the structured query constructed by our framework cannot retrieve all the gold answers, or unrelated answers are retrieved meanwhile, we consider that the framework cannot answer this NLQ correctly. Among the 100 test NLQs, our framework can correctly answer 70 NLQs. For the rest 30 NLQs, we divide them into three categories according to which module of our framework should be responsible for the failure. The analysis result is summarized in Table III. We can observe that the phrase mapping module should be responsible for most failures. There are mainly two causes: 1) The entity/relation phrases of the NLQ may be implied or over-expressed. For example, it is hard to extract the useful relation phrase from the NLQ "*Which space probes were sent into orbit around the sun?*". 2) The existing techniques cannot obtain the matching vertices/edges of some extracted entity/relation phrases. Both the structure computing module and the query generation module are responsible for a very limited number of failure NLQs, which further proves the effectiveness of our embedding-based approach. In addition, ten NLQs cannot be correctly answered by our framework due to other causes such as complex operation requirements for the target queries, e.g., "*Give me a list of all critically endangered birds.*".

TABLE III
FAILURE ANALYSIS OF OUR FRAMEWORK ON THE NLQ DATASET

Failure Module	# (Ratio)	Sample Failure NLQ
Phrase Mapping	13 (43.4%)	Which space probes were sent into orbit around the sun?
Structure Computing	4 (13.3%)	Who was the doctoral supervisor of Albert Einstein?
Query Generation	3 (10.0%)	Who was Vincent van Gogh inspired by?
Other	10 (33.3%)	Give me a list of all critically endangered birds.

IV. RELATED WORK

In this section, we discuss related researches in the following aspects: query construction models for NLQs and KG embedding techniques.

Most existing models construct graph-structured queries of given NLQs based on NLP techniques including syntactic parsing [8], [9], [21], semantic parsing [22]–[25], and templates [26]–[29]. Hu et al. [8] extract semantic relations from the dependency tree [30] of the NLQ to construct semantic query graphs which can be matched in the KG to obtain answers. Given an NLQ, the model in [22] first constructs a set of candidate queries, each of which is associated with a canonical realization in natural language. Then, the model selects the optimal candidate query by comparing the associated realizations with the given NLQ using a paraphrase model. Unger et al. [27] mirror the internal structure of the given NLQ by templates which can be instantiated by entity identification and predicate detection to construct target queries. These NLP technique-based models are complicated and time-consuming.

Along with the scales of KGs rapidly growing, the applications of KGs are increasingly compromised by the data sparsity and computational inefficiency. To address this problem, KG embedding models [12], [13], [17], [18], [31] which encode KGs into low-dimensional embedding spaces have been proposed. TransE [12] and its variants [17], [18], i.e., translation-based models, are the mainstream embedding models [31]. They cannot satisfy our requirement that the embedding vectors of vertices/edges in the same candidate set should be close to each other. TCE [13] and GAKE [31] are context-based embedding models which satisfy this requirement. However, they do not consider the generalized information, which makes them inapplicable to the query construction. Therefore, we cannot directly adopt the existing KG embedding models in our framework.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel framework which leverages the embedding techniques to construct the graph-structured queries of NLQs given by users over KGs. Our framework utilizes the learned embedding vectors to compute the query structure and determine the vertices/edges which

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.

form the target query. In order to learn the required embedding vectors, we adopt the translation mechanism and propose an embedding method which considers the generalized information based on GL-KGs of vertices/edges. Extensive experiments have been conducted on the benchmark dataset. The results demonstrate that our framework outperforms other state-of-the-art models in terms of effectiveness and efficiency. The main failure cause of our framework is the errors during the phrase mapping process. Therefore, we intend to explore more effective phrase mapping methods in the future. And, we are trying to improve the performance of our embedding method by adopting more sophisticated translation mechanisms.

ACKNOWLEDGMENT

This work was supported by National Key Research and Development Program of China (2016YFB1000903), National Natural Science Foundation of China (61532015, 61532004, 61672419, and 61672418), Innovative Research Group of the National Natural Science Foundation of China (61721002), Innovation Research Team of Ministry of Education (IRT_17R86), Project of China Knowledge Centre for Engineering Science and Technology, Science and Technology Planning Project of Guangdong Province (No. 2017A010101029), and Teaching Reform Project of XJTU (No. 17ZX044).

REFERENCES

[1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[2] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[3] S. Harris, A. Seaborne, and E. Prudhommeaux. (2013) Sparql 1.1 query language. [Online]. Available: http://www.w3.org/TR/sparql11-query/

[4] H. He and A. K. Singh, "Graphs-at-a-time: query language and access methods for graph databases," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 405–418.

[5] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao, "Natural language question/answering: Let users talk with the knowledge graph," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 217–226.

[6] P. Yin, N. Duan, B. Kao, J. Bao, and M. Zhou, "Answering questions with complex semantic constraints on open knowledge bases," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1301–1310.

[7] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum, "Natural language questions for the web of data," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 379–390.

[8] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 824–837, 2018.

[9] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao, "Natural language question answering over rdf: a graph data driven approach," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 313–324.

[10] D. Diefenbach, V. Lopez, K. Singh, and P. Maret, "Core techniques of question answering systems over knowledge bases: a survey," *Knowledge and Information Systems*, pp. 1–41, 2017.

[11] A. Bonifati, W. Martens, and T. Timm, "An analytical study of large sparql query logs," *Proc. VLDB Endow.*, vol. 11, no. 2, pp. 149–161, 2017.

[12] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[13] J. Shi, H. Gao, G. Qi, and Z. Zhou, "Knowledge graph embedding with triple context," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 2299–2302.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[15] M. Dubey, D. Banerjee, D. Chaudhuri, and J. Lehmann, "Earl: Joint entity and relation linking for question answering over knowledge graphs," *arXiv preprint arXiv:1801.03825*, 2018.

[16] R. Mihalcea and A. Csomai, "Wikify!: linking documents to encyclopedic knowledge," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, 2007, pp. 233–242.

[17] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *AAAI Conference on Artificial Intelligence*. AAAI, 2014, pp. 1112–1119.

[18] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion." in *AAAI Conference on Artificial Intelligence*, 2015, pp. 2181–2187.

[19] C. Unger, A.-C. N. Ngomo, and E. Cabrio, "6th open challenge on question answering over linked data (qald-6)," *Semantic Web Evaluation Challeng*, pp. 171–177, 2016.

[20] G. M. Mazzeo and C. Zaniolo, "Answering controlled natural language questions on rdf knowledge bases." in *International Conference on Extending DB Technology*, 2016, pp. 608–611.

[21] C. Giannone, V. Bellomaria, and R. Basili, "A hmm-based approach to question answering against linked data." in *CLEF (Working Notes)*, 2013.

[22] J. Berant and P. Liang, "Semantic parsing via paraphrasing," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2014, pp. 1415–1425.

[23] S. Hakimov, C. Unger, S. Walter, and P. Cimiano, "Applying semantic parsing to question answering over linked data: Addressing the lexical gap," in *NLDB*. Springer, 2015, pp. 103–109.

[24] A. Marginean, "Question answering over biomedical linked data with grammatical framework," *Semantic Web*, vol. 8, no. 4, pp. 565–580, 2017.

[25] K. Xu, S. Zhang, Y. Feng, and D. Zhao, "Answering natural language questions via phrasal semantic parsing," in *Natural Language Processing and Chinese Computing*. Springer, 2014, pp. 333–344.

[26] H. Bast and E. Haussmann, "More accurate question answering on freebase," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1431–1440.

[27] C. Unger, L. Buhmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over rdf data," in *Proceedings of the 21st International Conference on World Wide Web*. ACM, 2012, pp. 639–648.

[28] W. Zheng, L. Zou, X. Lian, J. X. Yu, S. Song, and D. Zhao, "How to build templates for rdf question/answering: An uncertain graph similarity join approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1809–1824.

[29] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over rdf data," in *International World Wide Web Conferences*. ACM, 2012, pp. 639–648.

[30] M.-C. De Marneffe and C. D. Manning. (2010) Stanford typed dependencies manual. [Online]. Available: https://nlp.stanford.edu/software/dependencies_manual.pdf

[31] J. Feng, M. Huang, Y. Yang *et al.*, "Gake: Graph aware knowledge embedding," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 641–651.

Authorized licensed use limited to: TRINITY COLLEGE LIBRARY DUBLIN. Downloaded on September 27,2022 at 15:21:10 UTC from IEEE Xplore. Restrictions apply.