Relevance Search over Schema-Rich Knowledge Graphs

Yu Gu
ygu@smail.nju.edu.cn
National Key Laboratory for Novel Software Technology
Nanjing University, China

Tianshuo Zhou
tianshuo.zhou@smail.nju.edu.cn
National Key Laboratory for Novel Software Technology
Nanjing University, China

Gong Cheng
gcheng@nju.edu.cn
National Key Laboratory for Novel Software Technology
Nanjing University, China

Ziyang Li
zyli@smail.nju.edu.cn
National Key Laboratory for Novel Software Technology
Nanjing University, China

Jeff Z. Pan
jeff.z.pan@abdn.ac.uk
Department of Computing Science
University of Aberdeen, UK

Yuzhong Qu
yzqu@nju.edu.cn
National Key Laboratory for Novel Software Technology
Nanjing University, China

ABSTRACT

Relevance search over a knowledge graph (KG) has gained much research attention. Given a query entity in a KG, the problem is to find its most relevant entities. However, the relevance function is hidden and dynamic. Different users for different queries may consider relevance from different angles of semantics. The ambiguity in a query is more noticeable in the presence of thousands of types of entities and relations in a schema-rich KG, which has challenged the effectiveness and scalability of existing methods. To meet the challenge, our approach called RelSUE requests a user to provide a small number of answer entities as examples, and then automatically learns the most likely relevance function from these examples. Specifically, we assume the intent of a query can be characterized by a set of meta-paths at the schema level. RelSUE searches a KG for diversified significant meta-paths that best characterize the relevance of the user-provided examples to the query entity. It reduces the large search space of a schema-rich KG using distance and degree-based heuristics, and performs reasoning to deduplicate meta-paths that represent equivalent query-specific semantics. Finally, a linear model is learned to predict meta-path based relevance. Extensive experiments demonstrate that RelSUE outperforms several state-of-the-art methods.

CCS CONCEPTS

• **Information systems → Data mining**; **Retrieval models and ranking**; • **Computing methodologies → Semantic networks**;

KEYWORDS

Relevance search, knowledge graph, meta-path, reasoning

ACM Reference Format:
Yu Gu, Tianshuo Zhou, Gong Cheng, Ziyang Li, Jeff Z. Pan, and Yuzhong Qu. 2019. Relevance Search over Schema-Rich Knowledge Graphs. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290970

1 INTRODUCTION

A knowledge Graph (KG), aka a heterogeneous information network, represents different types of relations between various kinds of entities as a graph with meaningful labels. Relevance search is an established task over KGs [4, 6, 8, 11, 13–15, 17, 18, 20, 21, 23]. It has found application in Web search, recommender systems, and many other domains. Given a query entity in a KG, the problem is to find its most relevant entities in the graph. For example, a user may search the KG in Fig. 1 for actors relevant to `Frank Oz`.

Relevance search is challenging mainly due to the ambiguities in queries. Users may have different kinds of hidden relevance behind a query entity. For example, by querying a director `Frank Oz`, a user may seek: (1) actors starring in movies directed by him, (2) actors starring in comedies (but not other genres of movies) directed by him, or (3) actors starring in movies that star him as he is also an actor. To resolve ambiguity, [4, 20, 23] assume the intent of a query can be characterized by a set of meta-paths. Their relative importance is learned from user preference, e.g., a small number of user-provided examples of relevant entities [4, 23]. A meta-path is a path at the schema level, consisting of an alternating sequence of entity types and relation types. For example, assume the following three meta-paths are predefined for Fig. 1:

$$\begin{aligned}
&\mathcal{P}_1 : \texttt{Director} \xrightarrow{\texttt{directed}} \texttt{Film} \xleftarrow{\texttt{acted\_in}} \texttt{Actor}\,,\\
&\mathcal{P}_2 : \texttt{Director} \xrightarrow{\texttt{directed}} \texttt{Comedy} \xleftarrow{\texttt{acted\_in}} \texttt{Actor}\,,\\
&\mathcal{P}_3 : \texttt{Actor} \xrightarrow{\texttt{acted\_in}} \texttt{Film} \xleftarrow{\texttt{acted\_in}} \texttt{Actor}\,.
\end{aligned} \tag{1}$$

They correspond to the above-mentioned three intents. If a user provides `Steve Martin` and `Bill Murray` as examples, we will know that $\mathcal{P}_1$ and $\mathcal{P}_2$ better characterize the user's intent because the query entity and user-provide examples are connected by paths in the KG that follow $\mathcal{P}_1$ and $\mathcal{P}_2$ but not $\mathcal{P}_3$. Therefore, `Kevin Kline` will be returned as the top-ranked answer entity as it is also connected to the query entity by a path that follows $\mathcal{P}_1$ and $\mathcal{P}_2$. However, relying on predefined meta-paths, such methods could hardly scale to a schema-rich KG like DBpedia and YAGO, where thousands of types of entities and relations are involved.

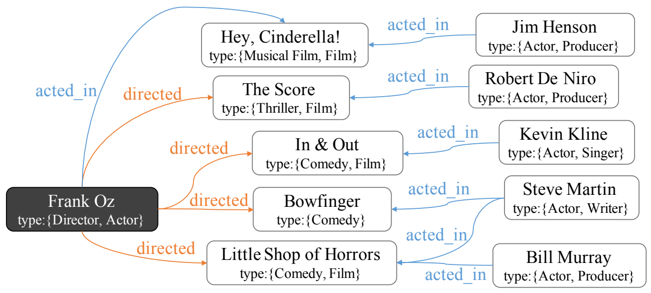

Figure 1: A running example of KG.

To extend this line of work to schema-rich KGs which are becoming increasingly popular, we propose a new approach called RelSUE. Its novelty is threefold, compared with existing methods.

- RelSUE automatically searches a KG for diversified significant meta-paths that best characterize the relevance of the user-provided examples to the query entity. It overcomes the shortcoming of relying on predefined meta-paths [4, 8, 18, 20, 23] which are impracticable for schema-rich KGs.
- RelSUE leverages distance and degree-based heuristics to dynamically reduce the search space and scale to large schema-rich KGs. It breaks through the limitation of bounded exhaustive search [6, 11, 21] which lose the expressiveness to represent the intents of complex queries.
- RelSUE exploits all the many types of entities and relations in a schema-rich KG, and performs reasoning to infer missing types and remove the side effect of semantically duplicate meta-paths. It has greater potential to accurately capture the intent of a query than those being restricted to single-typed entities [6, 15, 21] or disregarding relation types [13, 14].

Our proposed RelSUE outperforms several state-of-the-art methods in an extensive experiment based on two large schema-rich KGs.

The paper is structured as follows. Section 2 discusses related work. Section 3 formulates the problem. Section 4 describes RelSUE. Section 5 reports experiments. Section 6 concludes the paper.

2 RELATED WORK

2.1 Relevance Search

Early methods adopt random surfer models [2, 3, 10]. They define the relevance of an entity v to a query entity q as the stationary probability that a random surfer starting from q is at v. This probability is equivalent to a linear combination of the probabilities over all the paths from q to v. However, different types of paths have different semantics, implying different kinds of relevance. The semantics of a path can be represented by the meta-path it follows. PathSim [20] considers a linear combination of the probabilities only over the paths that follow predefined meta-paths which are manually weighted. Users can choose appropriate semantics and obtain more accurate results by specifying meta-paths and assigning them different weights. Weights can also be learned from user-provided examples of relevant entities [4, 23]. PReP [18] extends PathSim and represents cross-meta-path synergy by a generative model that goes beyond a linear combination of meta-paths. With a similar goal, Huang et al. [8] generalizes meta-path to meta-structure which is a directed acyclic graph.

Discussion. The above methods could not fit schema-rich KGs because they require predefined meta-paths (or meta-structures). It is difficult for a non-expert user to specify appropriate meta-paths by retrieving from thousands of types of entities and relations in a schema-rich KG. It is also impracticable for domain experts to predefine meta-paths for all types of information needs that users may have over a schema-rich KG.

To address the problem, PRA [11] and RelSim [21] automatically enumerate all possible length-bounded meta-paths. FSPG [15] greedily selects a subset of important meta-paths. Fang et al. [6] generalizes meta-path to meta-graph, and enumerates all possible size-bounded meta-graphs. With these methods, meta-paths (or meta-graphs) are not required to be predefined. However, as the number of possible meta-paths grows exponentially with their length, the length bound has to be set very small for schema-rich KGs. That would fail to represent the intents of complex queries.

Discussion. Our approach belongs to this line of research, but is specifically designed for schema-rich KGs. It dynamically searches for length-unbounded significant meta-paths that best characterize the relevance of the user-provided examples to the query entity, and its implementation is scalable to large KGs having thousands of types of entities and relations. Rather than being restricted to single-typed entities [6, 21] or heuristically transforming multi-typed entities into single-typed [15], our approach exploits all the types of an entity in meta-path search. This increases expressiveness, though it may lead to semantically duplicate meta-paths as a side effect; for deduplication, reasoning is performed in our approach.

Recently, KG embedding techniques have gained much attention. They represent entities and relations in a low-dimensional continuous vector space [22]. Although one can simply measure the relevance of an entity to a query entity by calculating the distance between their vectors, a more promising strategy is to directly learn an embedding model for relevant entity pairs. ProxEmbed [13] and D2AGE [14] adopt this idea. They sample paths between each pair of entities based on random walks, and use entity types as features.

Discussion. Although these methods can work on a schema-rich KG, their practicability would be debatable. They have many parameters to learn and hence to achieve good results, they need a large number of training examples which a user is unlikely to provide for a search task. Another concern is their running time, which may cause intolerable delays in user interaction. In fact, they are more suitable for offline tasks like link prediction where a large number of training examples are available. For relevance search, our approach is more suitable due to its quick response and better performance given a small number of user-provided examples.

2.2 Entity Set Expansion

Entity set expansion (ESE) is to find entities that are similar to a given small set of seed entities. Lim et al. [12] trains a classifier by learning a support vector machine from seed entities. Features are shortest-path distance from distinct types of entities. MP_ESE [25] heuristically finds a set of important meta-paths that connect many pairs of seed entities. These meta-paths are supposed to also connect seed entities to their similar entities. ESER [24] extends MP_ESE and additionally weights meta-paths by their discriminability. Weights can also be learned from seed entities [5]. A similar line of research is termed query by example [7, 9, 16].

Discussion. These methods can be adapted for relevance search in our context, by treating user-provided examples as seed entities to be expanded. However, the query entity may have to be ignored in ESE; this disconnection between the query entity and user-provided examples may distort the interpretation of the user's intent.

3 PROBLEM

3.1 Preliminaries

Let $\mathcal{T}$ and $\mathcal{R}$ be the set of all entity types and the set of all relation types, respectively. Entities and their relations to each other are represented as a knowledge graph.

Definition 3.1 (Knowledge Graph). A knowledge graph (KG) is a directed labeled graph denoted by $G = \langle V, E, \Phi, \Psi \rangle$ where

- V is a set of entities as nodes in the graph,
- E is a set of directed edges in the graph representing binary relations between entities,
- $\Phi : V \mapsto \mathsf{P}(\mathcal{T})$ is a labeling function that assigns each entity $v \in V$ a set of entity types $\Phi(v) \subseteq \mathcal{T}$, where $\mathsf{P}(\mathcal{T})$ represents the power set of $\mathcal{T}$, and
- $\Psi : E \mapsto \mathcal{R}$ is a labeling function that assigns each edge $e \in E$ a relation type $\Psi(e) \in \mathcal{R}$.

The small KG in Fig. 1 is used as a running example in this paper.

Relation types, entity types, and their subsumption relationships to each other form the schema of a KG.

Definition 3.2 (Schema of KG). For a KG $G = \langle V, E, \Phi, \Psi \rangle$, its schema is a triple $\langle \mathcal{T}_G, \mathcal{R}_G, \sqsubseteq_G \rangle$ where

- $\mathcal{T}_G \subseteq \mathcal{T}$ is the set of entity types that appear in G:
$$\mathcal{T}_G = \bigcup_{v \in V} \Phi(v)\,, \tag{2}$$
- $\mathcal{R}_G \subseteq \mathcal{R}$ is the set of relation types that appear in G:
$$\mathcal{R}_G = \bigcup_{e \in E} \{\Psi(e)\}\,, \tag{3}$$
- $\sqsubseteq_G \subseteq (\mathcal{T}_G \times \mathcal{T}_G)$ represents the subsumption relationships between entity types, which is reflexive and transitive.

For example, we define $\texttt{Comedy} \sqsubseteq_G \texttt{Film}$ in the schema of Fig. 1.

We call G a *schema-rich* KG if its schema contains many (e.g., more than hundreds of) entity types and/or relation types. Otherwise, G is *schema-simple*. DBLP is a typical schema-simple KG, compared with schema-rich KGs like DBpedia and YAGO which contain thousands of entity and relation types.

In G, a path is written as $v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_l} v_l$. The edges in a path are not required to all go the same direction. To consistently use right arrows, we rewrite $\xleftarrow{e}$ in the form of $\xrightarrow{e^{-1}}$, subject to $\Psi(e^{-1}) = \Psi^{-1}(e)$ which represents the converse of $\Psi(e)$. For example, the following path is present in Fig. 1:

$$\texttt{Frank Oz} \xrightarrow{\texttt{directed}} \texttt{In \& Out} \xrightarrow{\texttt{acted\_in}^{-1}} \texttt{Kevin Kline}\,. \tag{4}$$

Given a path in a KG, by substituting the entities in the path with their types, we obtain a meta-path at the schema level.

Definition 3.3 (Meta-Path [20]). Given a KG G and its schema $\langle \mathcal{T}_G, \mathcal{R}_G, \sqsubseteq_G \rangle$, a meta-path $\mathcal{P}$ is an alternating sequence of entity types and relation types:

$$\mathcal{P} : T_0 \xrightarrow{R_1} T_1 \xrightarrow{R_2} \cdots \xrightarrow{R_l} T_l\,, \tag{5}$$

where $T_0, \ldots, T_l \in \mathcal{T}_G$ are entity types, $R_1, \ldots, R_l \in \mathcal{R}_G$ are relation types (or their converses), and l is the length of $\mathcal{P}$. A path $v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_l} v_l$ in G is said to *follow* $\mathcal{P}$ if

$$\begin{aligned} &\forall 0 \le i \le l, T_i \in \Phi(v_i)\,, \\ &\forall 1 \le j \le l, R_j = \Psi(e_j)\,. \end{aligned} \tag{6}$$

For example, the path in Eq. (4) follows $\mathcal{P}_1$ and $\mathcal{P}_2$ in Eq. (1).

3.2 Problem Statement

Relevance search takes a query entity as input, and returns top-ranked entities in a KG that are most relevant to the query entity.

Definition 3.4 (Relevance Search). Given a KG $G = \langle V, E, \Phi, \Psi \rangle$, let $\texttt{rel} : (V \times V) \mapsto [0, 1]$ be a relevance function. For $u, v \in V$, $\texttt{rel}(u, v)$ returns the relevance of v to u. The problem of relevance search is to find k top-ranked answer entities in V that are most relevant to a query entity $q \in V$ w.r.t. $\texttt{rel}$.

Relevance search is challenging because $\texttt{rel}$ is hidden and dynamic, varying according to users and queries. Similar to [23], we assume a user provides not only a query entity but also a small number of answer entities as examples. Our focus is then to learn the most likely $\texttt{rel}$ from user-provided examples.

Definition 3.5 (Relevance Search by Examples). Let $\Lambda \subseteq V$ be a set of user-provided examples, and define $\overline{\Lambda} = V \setminus \Lambda$. We transform Definition 3.4 into finding $\texttt{rel}$ such that

$$\forall \lambda \in \Lambda, v \in \overline{\Lambda},\ \texttt{rel}(q, \lambda) \ge \texttt{rel}(q, v)\,. \tag{7}$$

For our running example in Fig. 1, we define $q = \texttt{Frank Oz}$ and $\Lambda = \{\texttt{Steve Martin}, \texttt{Bill Murray}\}$. All the other entities in the graph comprise $\overline{\Lambda}$, in which $\texttt{Kevin Kline}$ is the best answer entity.

4 APPROACH

4.1 Overview of the Approach

We assume the intent of a query can be characterized by a linear combination of a set of n meta-paths: $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. For a candidate entity $v \in V$, its relevance to the query entity q is calculated by

$$\texttt{rel}(q, v) = \sum_{i=1}^{n} w_i \cdot \gamma(q, v | \mathcal{P}_i)\,, \tag{8}$$

where w_i is the weight of $\mathcal{P}_i$, and $\gamma(q, v|\mathcal{P}_i)$ measures the relevance of v to q w.r.t. $\mathcal{P}_i$. In particular, $\gamma(q, v|\mathcal{P}_i) = 0$ if none of the paths from q to v follow $\mathcal{P}_i$. The specific implementation of γ is orthogonal to our research contribution; various measures have been proposed in the literature [11, 17, 19, 20]. We choose PCRW [11] as our γ. It calculates the probability of reaching v from q by a random walk along the paths in G that follow $\mathcal{P}_i$.

In Eq. (8), two problems remain to be solved.

First, in a schema-rich KG, there can be thousands or even millions of kinds of meta-paths. It is resource-consuming, if not impossible, to use all of them in Eq. (8) as done in [6, 11, 21] while users expect answers in real time. Given so many possibilities, it is also impracticable to request a user to manually configure proper meta-paths as assumed in [4, 8, 18, 20, 23]. In Section 4.2, we will introduce how to automatically and efficiently search for n diversified significant meta-paths that best characterize the intent of a query. *Such a quick yet accurate preselection is critical to the design of a scalable meta-path based method for schema-rich KGs, and is regarded as our major technical contribution.*

Second, the weights of the selected n meta-paths are to be automatically determined. In Section 4.3, we will describe how to learn the weights from user-provided positive examples along with automatically generated negative examples. For Eq. (8) we choose a simple linear combination because a user is unlikely to provide many examples for learning a complex model.

4.2 Meta-Path Search

We aim to preselect n diversified significant meta-paths that best characterize the intent of a query. We assess the significance of each meta-path based on user-provided examples, and select n top-ranked ones. The selection is diversified by removing duplicate meta-paths that represent equivalent query-specific semantics. To implement these ideas on a large schema-rich KG, we design two algorithms: exhaustive search and heuristic search.

4.2.1 Significance of Meta-Path. According to Definition 3.5, a good relevance function `rel` is expected to return larger values for Λ than $\overline{\Lambda}$. In Eq. (8), `rel` is decomposed into a linear combination of γ over n meta-paths. So a good meta-path that accurately characterizes the intent of a query is one based on which γ produces higher relevance for Λ than $\overline{\Lambda}$. Therefore, we define the significance of a meta-path $\mathcal{P}$ as follows:

$$\mathtt{sig}(\mathcal{P}|q,\Lambda) = \beta^l \frac{1}{|\Lambda| \cdot |\overline{\Lambda}|} \sum_{\lambda \in \Lambda} \sum_{v \in \overline{\Lambda}} \mathtt{I}(\gamma(q,\lambda|\mathcal{P}) - \gamma(q,v|\mathcal{P})) , \quad (9)$$

where l is the length of $\mathcal{P}$, β is a decay factor for penalizing long overfitted meta-paths, and $\mathtt{I}$ is an indicator function:

$$\mathtt{I}(\gamma(q,\lambda|\mathcal{P}) - \gamma(q,v|\mathcal{P})) = \begin{cases} 1 & \text{if } \gamma(q,\lambda|\mathcal{P}) > \gamma(q,v|\mathcal{P}) , \\ 0 & \text{if } \gamma(q,\lambda|\mathcal{P}) \leq \gamma(q,v|\mathcal{P}) . \end{cases} \quad (10)$$

Note that $\overline{\Lambda}$ can be a very large set, but the computation of `sig` typically involves a mere limited subset because $\gamma(q,v|\mathcal{P}) > 0$ only if v is reachable from q via a path in G that follows $\mathcal{P}$. Also for this reason, we have $\mathtt{sig}(\mathcal{P}|q,\Lambda) > 0$ only if at least one entity in Λ is reachable from q via a path in G that follows $\mathcal{P}$.

4.2.2 Selection of Diversified Meta-Paths. Two meta-paths are somehow indistinguishable if they are followed by the same set of paths in G, e.g., the following two meta-paths in Fig. 1:

$$\begin{aligned} \mathcal{P}_4 &: \mathtt{Director} \xrightarrow{\mathtt{directed}} \mathtt{Thriller} \xleftarrow{\mathtt{acted\_in}} \mathtt{Actor} , \\ \mathcal{P}_5 &: \mathtt{Director} \xrightarrow{\mathtt{directed}} \mathtt{Thriller} \xleftarrow{\mathtt{acted\_in}} \mathtt{Producer} . \end{aligned} \quad (11)$$

In such a case, $\forall v \in V,\ \gamma(q,v|\mathcal{P}_4)$ is equal (or close) to $\gamma(q,v|\mathcal{P}_5)$ for typical measures of γ. In this sense, *they characterize the intent of this particular query from practically (nearly) equivalent angles of semantics.* Such semantically duplicate meta-paths widely exist in schema-rich KGs where an entity has many and various types. Removing them will improve the diversity of the n selected meta-paths, and may form a more thorough understanding of the query.

Definition 4.1 (Equivalence between Meta-Paths). For a query entity q, two meta-paths $\mathcal{P}_i$ and $\mathcal{P}_j$ are q-equivalent, denoted by $\mathcal{P}_i \sim_q \mathcal{P}_j$, if they are followed by the same set of paths starting from q in G.

This equivalence relation is specific to q. For a different query entity q', two q-equivalent meta-paths may be followed by different paths starting from q' and hence be not q'-equivalent.

Instead of simply choosing the n most significant meta-paths, we additionally require the selected meta-paths not to be q-equivalent. The preselection of n diversified significant meta-paths can be formulated as a multidimensional 0-1 knapsack problem (MKP):

$$\begin{aligned} \text{to maximize } & \sum_{i=1}^{m} y_i \cdot \mathtt{sig}(\mathcal{P}_i|q,\Lambda) , \\ \text{subject to } & \sum_{i=1}^{m} y_i \leq n , \\ & y_i + y_j \leq 1 \text{ for all } i \neq j \text{ and } \mathcal{P}_i \sim_q \mathcal{P}_j , \\ & y_i \in \{0,1\} \text{ for all } 1 \leq i \leq m , \end{aligned} \quad (12)$$

where m is the number of candidate meta-paths, and y_i is a binary variable indicating whether $\mathcal{P}_i$ is selected ($y_i = 1$) or not ($y_i = 0$). This MKP is non-trivial because m can be very large in a schema-rich KG. It may even be impracticable to enumerate all possible meta-paths as candidates. In the following we introduce two algorithms for solving the problem.

4.2.3 Exhaustive Meta-Path Search. The first algorithm we devise is to exhaustively enumerate all the meta-paths between q and Λ as candidates. Other meta-paths are ignored because their $\mathtt{sig} = 0$. Then, the MKP in Eq. (12) is solved by a greedy strategy. However, the number of possible meta-paths grows exponentially with their length, which we have to bound for this algorithm.

In Algorithm 1, SrchP performs bidirectional search to find all the paths between q and Λ in G that are not longer than L, denoted by P (line 1); L is a length bound. GenMP exhaustively generates all the meta-paths that are followed by a path in P, denoted by CMP (line 2). For each candidate meta-path in CMP, we calculate its `sig`, and sort all of them in descending order of `sig` (lines 3–5). To find n diversified significant meta-paths denoted by MP, we check the candidate meta-paths one by one (lines 6–13). In each iteration, a meta-path $\mathcal{P}_i$ is added to MP only if it is not q-equivalent to any meta-path that has been added to MP in previous iterations.

Use of Reasoning. A KG may be incomplete. When G is schema-rich, its original Φ function may assign an entity an incomplete set of types, and hence GenMP may miss some potentially significant meta-paths. For example, `Film` should be a type of `Bowfinger`, which is missing in Fig. 1. We infer missing entity types by performing *deductive reasoning* over the schema of G. Specifically, we use the subsumption relationships between entity types ($\sqsubseteq_G$). We offline precompute the transitive closure of $\sqsubseteq_G$, and use it to add missing high-level entity types. The augmented KG is materialized and replaces the original G.

Algorithm 1: Exhaustive meta-path search.

```
Input: A KG $G$, a query entity $q$, a set of user-provided
       examples $\Lambda$, a number $n$ of meta-paths to select, and
       a length bound $L$.
Output: A set of meta-paths $MP$.
1  $P \leftarrow$ SrchP($G, q, \Lambda, L$);
2  $CMP \leftarrow$ GenMP($P$);
3  foreach $\mathcal{P}_i \in CMP$ do
4      Calculate sig($\mathcal{P}_i | q, \Lambda$);
5  Sort $CMP$ in descending order of sig;
6  $MP \leftarrow \emptyset$;
7  foreach $\mathcal{P}_i \in CMP$ do
8      if $\mathcal{P}_i \sim_q \mathcal{P}_j$ for any $\mathcal{P}_j \in MP$ then
9          Skip $\mathcal{P}_i$ and continue with the next iteration;
10     else
11         $MP \leftarrow MP \cup \{\mathcal{P}_i\}$;
12         if $|MP| = n$ then
13             Terminate the loop;
14 return $MP$;
```

Optimality. *MP* is an optimal solution to Eq. (12) under the length bound. To prove it, recall that q-equivalence is an equivalence relation and induces a partition of *CMP*. Therefore, all the feasible solutions to Eq. (12) define a *matroid*, which guarantees the optimality of our greedy strategy.

Limitations. The algorithm has two limitations. First, it is difficult to choose L. A small value weakens the expressiveness of the approach as some queries are represented by longer meta-paths. A large value enlarges the search space to which SrchP and GenMP may not scale. Second, it is time-consuming to calculate sig and sort all the candidate meta-paths when there are many. We will address these issues in the next algorithm.

4.2.4 Heuristic Meta-Path Search. To address the two limitations of the exhaustive search algorithm, in our second algorithm, the length of meta-paths is not explicitly bounded, but the search space is reduced heuristically. The idea is to perform unidirectional search starting from q, and iteratively expand a *search tree* which is an abstract of G at the schema level. As illustrated in Fig. 2, each node (called a tree-node) represents a set of entities of the same type denoted by Rep, and each edge (called a tree-edge) represents a set of relations of the same type. The search is terminated after finding n diversified significant meta-paths; each corresponds to a path starting from the root in the search tree. For example, the path to Actor in the upper right of Fig. 2 corresponds to $\mathcal{P}_2$ in Eq. 1.

In Algorithm 2, a search tree $\mathcal{ST}$ is initialized with q as the root (line 1). It is iteratively expanded (lines 2–16) until it is fully expanded (line 3) or n diversified significant meta-paths are accepted (lines 15–16). In each iteration, SelTN selects the most promising leaf tree-node to expand, denoted by *stn* (line 4). ExpST expands *stn* and returns its child tree-nodes (line 5). For each of these new tree-nodes, referred to as *tn* (line 6), its promise is evaluated by EvalTN (line 7). If Rep(*tn*) overlaps with Λ, e.g., Rep(Actor) $\cap \Lambda$ = {Steve Martin, Bill Murray} in our running example, a new meta-path $\mathcal{P}_i$ is generated by GetMP, which corresponds to the path in $\mathcal{ST}$ from the root to *tn* and has a non-zero

Algorithm 2: Heuristic meta-path search.

```
Input: A KG $G$, a query entity $q$, a set of user-provided
       examples $\Lambda$, a number $n$ of meta-paths to select, and
       a significance threshold $\tau$.
Output: A set of meta-paths $MP$.
1  Initialize a search tree $\mathcal{ST}$ with $q$ as the root;
2  $MP \leftarrow \emptyset$;
3  while $\mathcal{ST}$ can be expanded do
4      $stn \leftarrow$ SelTN($\mathcal{ST}$);
5      $TN \leftarrow$ ExpST($stn$);
6      foreach $tn \in TN$ do
7          EvalTN($tn$);
8          if $tn$ contains entities in $\Lambda$ then
9              $\mathcal{P}_i$ = GetMP($tn$);
10             if sig($\mathcal{P}_i | q, \Lambda$) $\geq \tau$ then
11                 if $\mathcal{P}_i \sim_q \mathcal{P}_j$ for any $\mathcal{P}_j \in MP$ then
12                     Skip $tn$ and continue with the next
                         iteration;
13                 else
14                     $MP \leftarrow MP \cup \{\mathcal{P}_i\}$;
15                     if $|MP| = n$ then
16                         Terminate the loop;
17 return $MP$;
```

value of sig (lines 8–9). If sig($\mathcal{P}_i$) reaches a predefined threshold τ (line 10), $\mathcal{P}_i$ is called a *sufficiently significant* meta-path, and is added to *MP* only if it is not q-equivalent to any meta-path that has been added to *MP* in previous iterations (lines 11–14). In the following we elaborate several key steps in the algorithm.

Expansion (ExpST). A tree-node *stn* is expanded in two steps. (1) All the edges incident from/to Rep(*stn*) in G, except those leading to a cycle, are grouped according to their relation types. Within each group identified by a distinct relation type $r \in \mathcal{R}_G$, the neighbors of Rep(*stn*) in G are denoted by

$$\mathtt{Nbr}(\mathtt{Rep}(stn), r) = \{v \in V : \exists u \in \mathtt{Rep}(stn),\ (u \xrightarrow{r} v) \in E\}\,. \quad (13)$$

(2) One or more new tree-nodes are generated from Nbr(Rep(*stn*), r) and connected to *stn* via new tree-edges representing r. Each new tree-node represents a distinct type of entities in Nbr(Rep(*stn*), r).

It is unnecessary to create new tree-nodes for every type of entities in Nbr(Rep(*stn*), r). Two new tree-nodes tn_i and tn_j that satisfy Rep(tn_i) = Rep(tn_j) correspond to q-equivalent meta-paths, e.g., the two tree-nodes sharing Rep = {Hey, Cinderella!} in Fig. 2. Only one of them will be kept in $\mathcal{ST}$. From the perspective of reasoning, it amounts to the computation of (locally) equivalent entity types by performing *inductive reasoning*. This early removal of semantically duplicate meta-paths also reduces the search space. Note that when Rep(tn_i) ≠ Rep(tn_j), although tn_i and tn_j correspond to two meta-paths that are not q-equivalent, their descendants in $\mathcal{ST}$ may correspond to q-equivalent meta-paths. Therefore, it is still necessary to check for q-equivalence in the algorithm (lines 11–12).

To efficiently implement ExpST, in the adjacency list representation of G, neighbors are indexed first by relation types and then by entity types. This index also facilitates the computation of our measure γ, i.e., PCRW. To efficiently check for and prevent cycles,

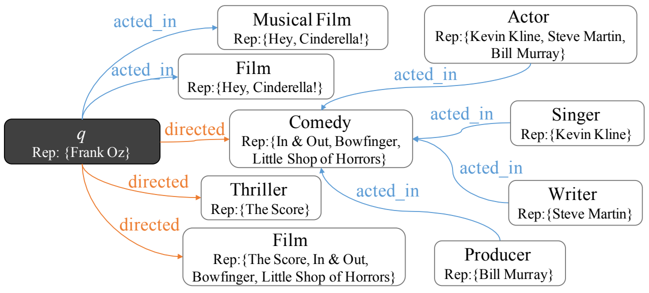

Figure 2: A partially expanded search tree.

for each tree-node we store not only its Rep but also the actual paths from q to Rep in G. These paths are also useful when checking for q-equivalence in the algorithm (lines 11–12).

Evaluation of Promise (EvalTN). SelTN selects the most promising leaf tree-node to expand. The promise of a tree-node tn, denoted by $\texttt{prms}(tn)$, is evaluated by EvalTN, which estimates the largest significance of the meta-paths corresponding to the descendants of tn in $\mathcal{ST}$. An accurate estimation in reasonable time is difficult, if not impossible. We propose two heuristics: *distance* and *degree*.

The first heuristic is based on shortest-path distance. Following Eq. (9), short meta-paths from q to Λ are prioritized. Therefore, for each entity in Λ, we calculate its smallest distance from $\texttt{Rep}(tn)$; then we calculate the average of all the entities in Λ:

$$\texttt{prms}_{\texttt{dist}}(tn) = \texttt{dpth}(tn) + \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \min_{v \in \texttt{Rep}(tn) \setminus \{\lambda\}} \texttt{dist}(v, \lambda) \,, \quad (14)$$

where $\texttt{dpth}(tn)$ denotes the depth of tn in $\mathcal{ST}$, i.e., its distance from the root, and dist returns the shortest-path distance between two entities in G. The most promising tree-node has the smallest value of $\texttt{prms}_{\texttt{dist}}$. For efficient calculation of dist, we implement a distance oracle [1], which allows for reasonably fast distance calculation based on certain precomputed information whose size is much smaller than the size of materializing distances between all pairs of entities, thereby achieving a satisfying trade-off between time and space for handling large KGs.

The second heuristic is used when multiple leaf tree-nodes share the smallest value of $\texttt{prms}_{\texttt{dist}}$. To break ties, we calculate the sum of the degree of $\texttt{Rep}(tn)$ in G:

$$\texttt{prms}_{\texttt{deg}}(tn) = \sum_{v \in \texttt{Rep}(tn)} \texttt{deg}(v) \,, \quad (15)$$

where $\texttt{deg}(v)$ returns the degree of v in G, i.e., the number of edges incident from/to v. A more promising tree-node has a smaller value of $\texttt{prms}_{\texttt{deg}}$. The idea behind this heuristic is twofold. First, expanding entities having a smaller degree is computationally less expensive. Second, more importantly, after expanding such entities, fewer entities in $\overline{\Lambda}$ are reached. In other words, the corresponding meta-paths better discriminate Λ from $\overline{\Lambda}$. They are more likely to be significant according to Eq. (9).

Use of Reasoning. Similar to the first algorithm, we perform offline deductive reasoning to infer missing entity types. In ExpST, as mentioned, we also perform online inductive reasoning to infer (locally) equivalent entity types.

Optimality and Limitations. *MP* is not guaranteed to be an optimal solution to Eq. (12). The selected meta-paths are divesified and sufficiently significant, but may not be the most significant ones. Despite this limitation, the algorithm allows long meta-paths and scales to large schema-rich KGs.

Table 1: Statistics of Datasets

	Entity	Relation	Entity Type	Relation Type
DBpedia	3,480,806	13,301,510	453	645
YAGO	4,295,825	12,430,700	536,648	37

4.3 Relevance Learning

With n preselected meta-paths, the next step is to determine their weights in Eq. (8). Although each meta-path can be simply weighted by its sig value, we aim to fine-tune the weights using machine learning. We formulate a binary classification model. An entity is either relevant or irrelevant to q. To train the model in a supervised manner, in addition to the user-provided positive examples Λ, we automatically generate high-quality negative examples.

Generation of Negative Examples. We collect all the entities in G that are reachable from q via a path that follows a preselected meta-path. Excluding Λ from this collection, the remaining entities form candidate entities, denoted by *Cand*, from which k answer entities will be picked because *Cand* are more similar to Λ than other entities in G from the perspective of meta-path. We randomly sample a small subset of *Cand* as negative examples for training. Such entities, if not positive, are high-quality negative examples because they are what a learning model is quite uncertain about.

Model Selection. For the learning model, we choose the linear soft-margin support vector machine which extends Eq. (8). We configure it to penalize false negatives 50 times more than false positives, to respect user-provided positive examples.

5 EXPERIMENTS

5.1 Datasets

We use two well-known encyclopedic schema-rich KGs: DBpedia and YAGO in Table 1. For DBpedia (version 2016-10), we obtain its KG from *Mappingbased Objects*, entity types from *Instance Types*, and subsumption relationships between entity types from *DBpedia Ontology*. For YAGO (version 3.1), these are obtained from *yagoFacts*, *yagoSimpleTypes*, and *yagoSimpleTaxonomy*, respectively.

5.2 Queries and Labeled Answers

Most existing methods are evaluated on schema-simple KGs. Some also use schema-rich KGs [15, 21], but the queries they use are rather simple, most of which can be characterized by a single relation. To perform a more extensive evaluation, we systematically create a broad spectrum of queries according to three dimensions:

sgl/mult whether the intent can be characterized by a single meta-path (**sgl**) or not (**mult**),

s/l whether the intent can be characterized by short meta-paths not longer than one (**s**) or not (**l**), and

bin/rank whether candidate entities can be clearly and unambiguously classified as answers and non-answers (**bin**) or they can only be ranked (**rank**).

In Table 2, for each dataset we create four groups of queries that represent distinct combinations of the three dimensions. For

Table 2: Query Groups for DBpedia (Q_D) and YAGO (Q_Y)

	Query Intent	sgl/mult	s/l	bin/rank
Q_{D1}	grapes grown in q (a chateau)	sgl	s	bin
Q_{D2}	scientists in the same field as q (a scientist)	sgl	l	bin
Q_{D3}	actors starring in many movies directed by q (a director)	sgl	l	rank
Q_{D4}	players drafted by the same team and playing in the same position as q (a player)	mult	l	bin
Q_{Y1}	comedies starring q (an actor)	sgl	s	bin
Q_{Y2}	companies founded by a schoolmate of q (a person)	sgl	l	bin
Q_{Y3}	actors starring in many comedies directed by q (a director)	sgl	l	rank
Q_{Y4}	scientists as both a schoolmate and a colleague of q (a scientist)	mult	l	bin

each group we generate ten random query entities. For each query entity we manually label a set of relevant entities, which later in the experiment are divided into user-provided examples and correct answer entities. Labels are binary for **bin** queries, and are graded in proportion to the number of movies/comedies for **rank** queries Q_{D3}/Q_{Y3}. Labels for the 80 queries are published[1].

5.3 Baselines and Variants of RelSUE

We compare RelSUE with four methods that fit schema-rich KGs. **RelSim** [21] is a state-of-the-art meta-path based method for relevance search. **ProxEmbed** [13] and **D2AGE** [14] are two recent embedding-based methods. **ESER** [24] is a state-of-the-art method for entity set expansion which can be adapted for relevance search.

Configuration of RelSim. RelSim takes pairs of relevant entities as the input, and outputs other pairs of entities having the same kind of relevance. To adapt RelSim for our problem, we pair the query entity and each user-provided example as the input. For each pair of entities it outputs where the query entity is involved, we take the other entity in the pair as an answer entity. To automatically generate negative examples for training, the original implementation of RelSim constructs pairs of irrelevant entities by randomly exchanging entities between input pairs. This is meaningless in our context because all the input pairs contain a common query entity. Instead, we substitute user-provided examples in input pairs with negative examples generated based on meta-paths—in the same way they are generated in RelSUE. We generate $50 \cdot |\Lambda|$ negative examples for each query.

RelSim only supports single-typed entities. For DBpedia, we keep each entity's lowest type in the subsumption hierarchy. For YAGO, that would lead to a highly dispersed distribution of entity types as there are numerous types. So we keep each entity's unique type at the first level of the subsumption hierarchy.

RelSim enumerates all possible length-bounded meta-paths. We use RelSim$_x$ to refer to the version that uses x as the length bound.

Configuration of ProxEmbed and D2AGE. ProxEmbed and D2AGE take triples of entities as training examples. In each triple

[1]http://ws.nju.edu.cn/relevance/relsue

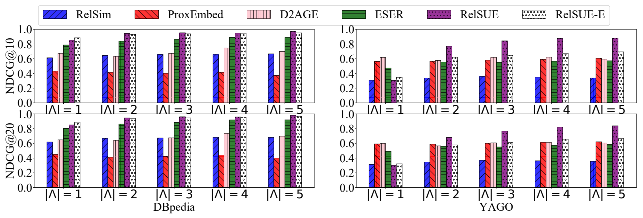

Figure 3: Overall results of NDCG@10 and NDCG@20.

$\langle q, v, u\rangle$, the entity v is more similar to the query entity q than the entity u. To construct such triples, we let v be a user-provided example, and let u be a random entity having the same type as v. As suggested in [13], we generate 100 training triples for each query.

The embedding dimension is 10 for ProxEmbed and 16 for D2AGE, to perform the computation in reasonable time. Other parameters are set to their best-tuned values in [13, 14]. YAGO has too many types of entities to be handled by ProxEmbed and D2AGE due to their one-hot encoding of entity types. We have to only use those at the first level of YAGO's subsumption hierarchy.

ProxEmbed and D2AGE cannot scale to millions of entities. We use ProxEmbed$_x$ and D2AGE$_x$ to refer to their versions that work on the x-hop neighborhood of the query entity.

Configuration of ESER. ESER takes a set of seed entities as the input, and outputs other entities similar to the seed entities. To adapt ESER for our problem, we treat user-provided examples as seed entities. ESER does not specify how it chooses an "anchor entity" from which meta-paths are generated to connect seed entities. We enhance ESER by assigning the query entity to the anchor entity.

ESER bounds the length of meta-paths. We use ESER$_x$ to refer to the version that uses x as the length bound.

Other Methods. Other recent methods [6, 15] are not compared in our experiments for the following reasons. Fang et al. [6] only supports symmetric entity types; i.e., the query entity and user-provided examples are required to have the same type, which is often violated in our context. For FSPG [15], we fail to reproduce its results due to some unclarified details in the algorithm. We have sent emails to its authors but have not received any response.

Variants of RelSUE. RelSUE refers to our heuristic search algorithm, and RelSUE-E$_x$ refers to our exhaustive search algorithm that uses x as the length bound. For each query, both methods generate $50 \cdot |\Lambda|$ negative examples for weight learning. Their number of preselected meta-paths is set to $n = 3$ for DBpedia and $n = 10$ for YAGO. We set $\beta = 0.9$ for Eq. (9) and $\tau = 0.5$ for RelSUE. A parameter study is reported later to justify our setting.

Setting of Bounds (x). For each method, we choose the largest value of x such that the computation can be performed within minutes but not longer. They are: RelSim$_2$, ProxEmbed$_2$, D2AGE$_2$, ESER$_3$, RelSUE-E$_3$ (on DBpedia) or RelSUE-E$_2$ (on YAGO).

5.4 Effectiveness Results

We use NDCG@k as our evaluation criterion.

5.4.1 *Overall Results.*
The average NDCG@10 and NDCG@20 of all the 40 queries on each dataset are presented in Fig. 3.

On DBpedia, RelSUE and RelSUE-E consistently outperform all the four baselines under all the settings of $|\Lambda|$. For example, when $|\Lambda| = 3$ which is a modest number for a user, their NDCG@10 scores

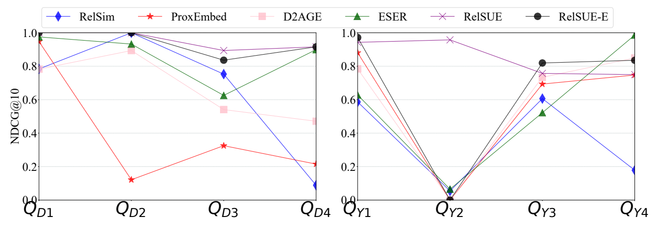

Figure 4: NDCG@10 on different query groups.

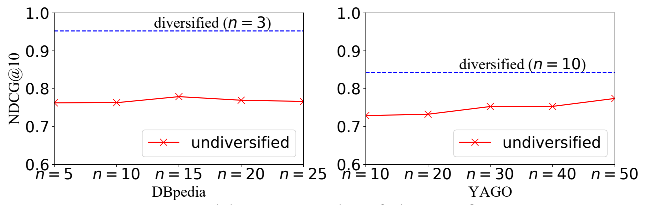

Figure 5: Ablation study of diversification.

are fairly high—in the range of 0.93–0.95, being notably better than ESER and largely superior to the other three methods.

On YAGO, the results are similar, except for $|\Lambda| = 1$ where RelSUE and RelSUE-E are not the best-performing methods. When $|\Lambda|$ increases, RelSUE and RelSUE-E achieve much better results as they can learn a more accurate relevance function from more examples. When $|\Lambda| = 5$, the NDCG@10 of RelSUE reaches 0.88.

RelSUE and RelSUE-E perform equally well on DBpedia, showing the effectiveness of the proposed heuristics. On YAGO, RelSUE even largely surpasses RelSUE-E due to the poor performance of RelSUE-E on one particular group of queries, which will be detailed in the following section.

5.4.2 *Results by Query Groups.*

The results on each query group are presented in Fig. 4. We fix $|\Lambda| = 3$ from now on.

On DBpedia, RelSUE and RelSUE-E outperform all the baselines on all the four query groups. Specifically, Q_{D1} are simple queries, so all the methods achieve good results. For Q_{D2}, ProxEmbed performs not well because it disregards relation types. D2AGE also disregards relation types but its more powerful non-linear expressiveness provides compensation. ESER, though is among the better ones on three groups, notably falls behind RelSUE on Q_{D3} due to its weakness in answering **rank** queries. RelSim, whose model is likely to prioritize one single instead of multiple meta-paths, shows poor performance on Q_{D4}.

On YAGO, the results are generally similar. The superior performance of RelSUE and RelSUE-E is partially due to their capability to exploit all the many types of entities in YAGO. We particularly highlight Q_{Y2}, on which RelSUE is the only method that works. Answering this group of queries relies on a meta-path of length 3, to which RelSim, ProxEmbed, D2AGE, and RelSUE-E cannot scale on YAGO. ESER can find this long meta-path, but its performance is limited by its scoring function.

5.4.3 *Ablation Study.*

We analyze the usefulness of diversification and weight learning.

Diversification. RelSUE selects n diversified meta-paths. In Fig. 5, when diversification is disabled, we witness a notable drop in NDCG because undiversified q-equivalent meta-paths have overlapping semantics and they collectively still miss important meta-paths.

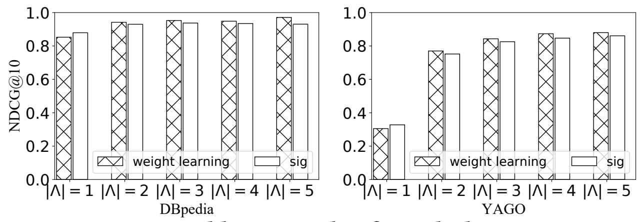

Figure 6: Ablation study of weight learning.

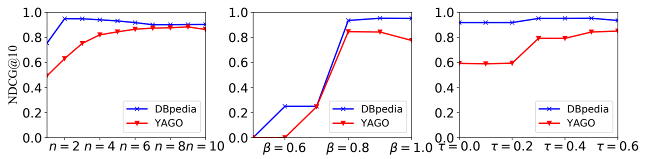

Figure 7: Influence of parameters on RelSUE.

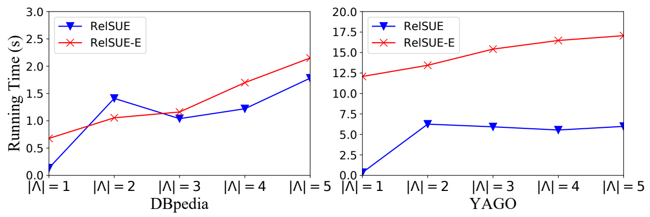

Figure 8: Running time.

The gap is consistently notable when n is increased to fairly large values. It demonstrates the usefulness of diversification.

Weight Learning. RelSUE fine-tunes the weights of preselected meta-paths using supervised learning. One may suggest weighting meta-paths directly by their `sig` values. In Fig. 6, we find learning slightly improves NDCG when Λ contains more than one entity.

5.4.4 *Parameter Study.*

In Fig. 7, we present the influence of the three parameters on RelSUE, namely n, β, and τ.

Number of Preselected Meta-Paths (n). On DBpedia, NDCG reaches a high standard very quickly when n grows. On YAGO, n is also fairly small (< 10) at convergence. The results suggest the high quality of the top-ranked meta-paths preselected by RelSUE.

Decay Factor (β). NDCG is low when β is small because some important but fairly long meta-paths are rejected. It peaks at $\beta = 0.8$ or 0.9 instead of $\beta = 1$ because in that case, some long overfitted meta-paths are selected.

Significance Threshold (τ). On YAGO, NDCG is not high when τ is small because many unimportant meta-paths are selected. However, it quickly converges to a high level when τ grows.

5.5 Efficiency Results

We report running time on a Xeon E5-2643 v4 (3.40GHz).

In Fig. 8, RelSUE and RelSUE-E are very fast on DBpedia. Both of them complete a search task in a few seconds. RelSim and ESER are actually even faster, using less than one second. However, taking NDCG scores into consideration, RelSUE and RelSUE-E are more cost-effective methods. ProxEmbed and D2AGE are relatively slow. They typically use one or several minutes for processing a query.

On YAGO which contains much more entity types than DBpedia, the running time of RelSUE-E increases to more than 10 seconds, but RelSUE is still reasonably fast. Both of them use less time than all the baselines. It demonstrates the scalability of our approach.

6 CONCLUSIONS

RelSUE is one of the first solutions to relevance search that are specifically optimized for handling large schema-rich KGs. Building on meta-path based relevance, it is distinguished from previous methods by: its automated and efficient selection of expressive, diversified, and significant meta-paths guided by user-provided examples, and its full exploitation of multiple entity types with reasoning support. These new technical features have made RelSUE significantly outperform the state of the art in an extensive evaluation where a broad spectrum of queries over two popular schema-rich KGs are used. The lead is taken by using a short running time. With this scalable approach, the application of relevance search can be extended from schema-simple KGs to more general domains, including Web-scale search and recommendation.

One limitation of our approach is the unsatisfying performance given very few examples provided on a YAGO-like KG, which contains hundreds of thousands of types of entities. In this case, relevance search is inherently difficult, but still, we intend to study more robust heuristics or to design efficient approximate algorithms with a performance guarantee. From another point of view, an interesting question would be whether and how a relevance search engine can actively solicit user feedback if the ambiguity of a query is believed to not be resolvable given the current examples.

Besides, to improve the running time of our approach, one direction is to offline build a certain kind of schema-level index, to support more efficient top-ranked meta-path search.

ACKNOWLEDGMENTS

This work was supported by the NSFC (61572247, 61772264), the EU Marie Currie K-Drive project (286348), the Six Talent Peaks (RJFW-011) and the Qing Lan Programs of Jiangsu Province.

REFERENCES

[1] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2013. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*. 349–360. https://doi.org/10.1145/2463676.2465315

[2] Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*. 635–644. https://doi.org/10.1145/1935826.1935914

[3] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. 2004. ObjectRank: Authority-Based Keyword Search in Databases. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*. 564–575.

[4] Shaoli Bu, Xiaoguang Hong, Zhaohui Peng, and Qingzhong Li. 2014. Integrating meta-path selection with user-preference for top-k relevant search in heterogeneous information networks. In *Proceedings of the IEEE 18th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2014, Hsinchu, Taiwan, May 21-23, 2014*. 301–306. https://doi.org/10.1109/CSCWD.2014.6846859

[5] Xiaohuan Cao, Chuan Shi, Yuyan Zheng, Jiayu Ding, Xiaoli Li, and Bin Wu. 2018. A Heterogeneous Information Network Method for Entity Set Expansion in Knowledge Graph. In *Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II*. 288–299. https://doi.org/10.1007/978-3-319-93037-4_23

[6] Yuan Fang, Wenqing Lin, Vincent Wenchen Zheng, Min Wu, Kevin Chen-Chuan Chang, and Xiaoli Li. 2016. Semantic proximity search on graphs with metagraph-based learning. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*. 277–288. https://doi.org/10.1109/ICDE.2016.7498247

[7] Jialong Han, Kai Zheng, Aixin Sun, Shuo Shang, and Ji-Rong Wen. 2016. Discovering Neighborhood Pattern Queries by sample answers in knowledge base. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*. 1014–1025. https://doi.org/10.1109/ICDE.2016.7498309

[8] Zhipeng Huang, Yudian Zheng, Reynold Cheng, Yizhou Sun, Nikos Mamoulis, and Xiang Li. 2016. Meta Structure: Computing Relevance in Large Heterogeneous Information Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 1595–1604. https://doi.org/10.1145/2939672.2939815

[9] Nandish Jayaram, Arijit Khan, Chengkai Li, Xifeng Yan, and Ramez Elmasri. 2015. Querying Knowledge Graphs by Example Entity Tuples. *IEEE Trans. Knowl. Data Eng.* 27, 10 (2015), 2797–2811. https://doi.org/10.1109/TKDE.2015.2426696

[10] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. 538–543. https://doi.org/10.1145/775047.775126

[11] Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning* 81, 1 (2010), 53–67. https://doi.org/10.1007/s10994-010-5205-8

[12] Lipyeow Lim, Haixun Wang, and Min Wang. 2013. Semantic queries by example. In *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*. 347–358. https://doi.org/10.1145/2452376.2452417

[13] Zemin Liu, Vincent W. Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2017. Semantic Proximity Search on Heterogeneous Graph by Proximity Embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 154–160.

[14] Zemin Liu, Vincent W. Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2018. Distance-Aware DAG Embedding for Proximity Search on Heterogeneous Graphs. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

[15] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. 2015. Discovering Meta-Paths in Large Heterogeneous Information Networks. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. 754–764. https://doi.org/10.1145/2736277.2741123

[16] Steffen Metzger, Ralf Schenkel, and Marcin Sydow. 2017. QBEES: query-by-example entity search in semantic knowledge graphs based on maximal aspects, diversity-awareness and relaxation. *J. Intell. Inf. Syst.* 49, 3 (2017), 333–366. https://doi.org/10.1007/s10844-017-0443-x

[17] Chuan Shi, Xiangnan Kong, Philip S. Yu, Sihong Xie, and Bin Wu. 2012. Relevance search in heterogeneous networks. In *15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings*. 180–191. https://doi.org/10.1145/2247596.2247618

[18] Yu Shi, Po-Wei Chan, Honglei Zhuang, Huan Gui, and Jiawei Han. 2017. PReP: Path-Based Relevance from a Probabilistic Perspective in Heterogeneous Information Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. 425–434. https://doi.org/10.1145/3097983.3097990

[19] Yizhou Sun, Rick Barber, Manish Gupta, Charu C. Aggarwal, and Jiawei Han. 2011. Co-author Relationship Prediction in Heterogeneous Bibliographic Networks. In *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2011, Kaohsiung, Taiwan, 25-27 July 2011*. 121–128. https://doi.org/10.1109/ASONAM.2011.112

[20] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *PVLDB* 4, 11 (2011), 992–1003.

[21] Chenguang Wang, Yizhou Sun, Yanglei Song, Jiawei Han, Yangqiu Song, Lidan Wang, and Ming Zhang. 2016. RelSim: Relation Similarity Search in Schema-Rich Heterogeneous Information Networks. In *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*. 621–629. https://doi.org/10.1137/1.9781611974348.70

[22] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Trans. Knowl. Data Eng.* 29, 12 (2017), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

[23] Xiao Yu, Yizhou Sun, Brandon Norick, Tiancheng Mao, and Jiawei Han. 2012. User guided entity similarity search using meta-path selection in heterogeneous information networks. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*. 2025–2029. https://doi.org/10.1145/2396761.2398565

[24] Xiangling Zhang, Yueguo Chen, Jun Chen, Xiaoyong Du, Ke Wang, and Ji-Rong Wen. 2017. Entity Set Expansion via Knowledge Graphs. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 1101–1104. https://doi.org/10.1145/3077136.3080732

[25] Yuyan Zheng, Chuan Shi, Xiaohuan Cao, Xiaoli Li, and Bin Wu. 2017. Entity Set Expansion with Meta Path in Knowledge Graph. In *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I*. 317–329. https://doi.org/10.1007/978-3-319-57454-7_25