RelSim: Relation Similarity Search in Schema-Rich Heterogeneous Information Networks [Extended Version]

Chenguang Wang[†] , Yizhou Sun[*] , Yanglei Song[‡] , Jiawei Han[‡] , Yangqiu Song[‡] , Lidan Wang[‡], Ming Zhang[†]
[†]School of EECS, Peking University
[*]College of Computer and Information Science, Northeastern University
[‡]Department of Computer Science, University of Illinois at Urbana-Champaign
wangchenguang@pku.edu.cn, yzsun@ccs.neu.edu, {ysong44,hanj,yqsong,lidan}@illinois.edu,
mzhang@net.pku.edu.cn

ABSTRACT

Recent studies have demonstrated the power of modeling real world data as *heterogeneous information networks (HINs)* consisting of multiple types of entities and relations, and the usefulness of using network schema as a high-level description of an HIN. Unfortunately, most of such studies (*e.g.*, similarity search) confine discussions on the networks with only a very small number of entity and relationship types (which we call *HINs with simple schema*), such as DBLP. In the real world, however, the network schema can be rather complex, such as in Freebase. In such *HINs with rich schema*, it is often too much burden to ask users to provide meta-path(s) explicitly for similarity search. It is more desirable to ask users to just provide some simple relation instance examples (*e.g.*, ⟨Barack Obama, John Kerry⟩ and ⟨George W. Bush, Condoleezza Rice⟩) as a query, and have the system generate meta-path(s) that best explains the *latent semantic relation (LSR)* implied by the query (*e.g.*, "president vs. secretary-of-state"). Such meta-paths will guide the system to find other similar relation instances (*e.g.*, ⟨Bill Clinton, Madeleine Albright⟩). In this paper, we systemically study the problem of relation similarity search in schema-rich HINs, and propose a relation similarity search framework that consists of two components: (1) LSR representation and learning: where an LSR is represented as a weighted combination of query-based meta-paths generated based on a query-based network schema and the weights are learned by an optimization model; and (2) an efficient similarity search algorithm: where the algorithm uses a new meta-path-based relation similarity measure, *RelSim*, to compute the relation similarity based on the learned LSR. Five real world datasets are constructed for relation similarity search, and the experiments on the datasets demonstrate the power of our approach.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Relation Similarity Search, Schema-Rich Heterogeneous Information Networks

1. INTRODUCTION

Heterogenous information networks (HINs) have been used recently for modeling real world relationships in many applications [9, 20, 26]. Network schema, a graph of entity types connected by relation types, has also been introduced as an abstractive description of HINs [21]. Previous HIN studies [20, 21] are confined to *simple schemas*, consisting of only a few entity and relation types, such as the DBLP network, with four entity types: *Paper*, *Venue*, *Author* and *Term*, and a few relation types connecting the entity types. However, in the real world, HINs can often be with more *sophisticated network schemas*, containing many more entity types and relation types. For example, the Freebase network[1] contains 1,500+ types of entities, such as *Organization*, *Profession*, *Book*, *Musician*, *Film*, and *Location*, and 35,000+ types of relations among the entity types, such as "is president of" and "is secretary of state of" [7]. We call such HINs with sophisticated network schemas as *schema-rich HINs*.

Many research problems arise with schema-rich HINs. For example, the basic functions, such as similarity search, will need to be re-examined. The concept of meta-path [21], a path on the graph of network schema, that describes the semantic meaning between entities, has been shown its effectiveness in similarity search in HINs with simple schema [16, 21]. One or a set of meta-paths can be easily provided by a user to represent her query intent or interest, *e.g.*, *find similar authors publishing papers at the same venue* can be specified easily as a meta-path *Author-Paper-Venue-Paper-Author*. However, in schema-rich HINs, it is unrealistic to ask users to provide meta-paths explicitly since there are too many possible meaningful paths to be chosen from a complex network schema, especially when the meta-paths needed are long and sophisticated. In such cases, it is more reasonable to ask users to provide a set of simple examples, *e.g.*, ⟨Barack Obama, John Kerry⟩ and ⟨George W. Bush, Condoleezza Rice⟩, as a query, and ask the system to generate the meta-paths that can best explain the *latent semantic relation (LSR)* in the user's query. With such generated meta-paths, the system can further automatically find other similar relation instances satisfying the same LSR "president vs. secretary-of-state," such as ⟨Bill Clinton, Madeleine Albright⟩, and use the new examples to generate more meta-paths iteratively.

The above process turns the meta-path generation problem to

[1]http://www.freebase.com/

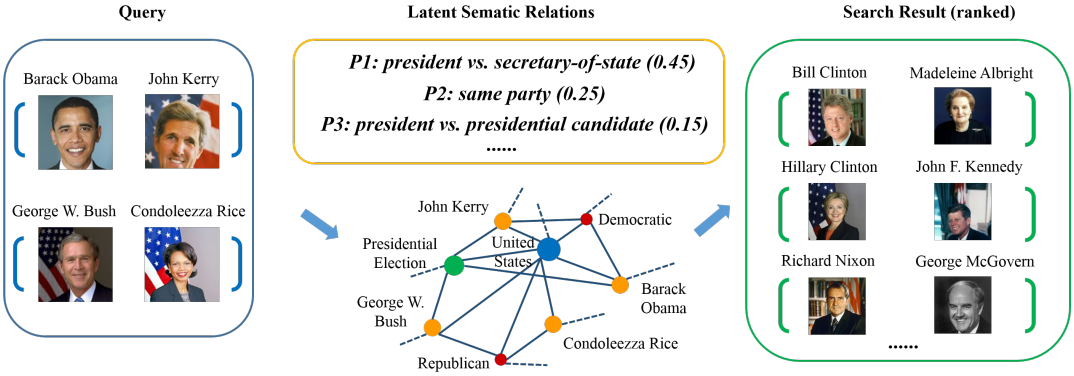


Figure 1: Relation similarity search in schema-rich HIN. Left: a user query; middle: different query-based meta-paths associated with corresponding weights ($\mathcal{P}_1 = \textit{President} \xrightarrow{\text{is president of}} \textit{Country} \xleftarrow{\text{is secretary of state of}} \textit{Secretary of State}$, $\mathcal{P}_2 = \textit{Politician} \xrightarrow{\text{is member of}} \textit{Party} \xleftarrow{\text{is member of}} \textit{Politician}$, $\mathcal{P}_3 = \textit{President} \xrightarrow{\text{is president of}} \textit{Country} \xleftarrow{\text{is presidential candidate of}} \textit{Presidential Candidate}$); right: ranked similar relation instances.

another fundamental problem: searching similar relation instances for a query in schema-rich HINs. As shown in Figure 1, our goal is to find similar relation instances based on the query $Q = \{\langle$Barack Obama, John Kerry$\rangle, \langle$George W. Bush, Condoleezza Rice$\rangle\}$. We address the following challenges:

1. *How to find the most likely LSR implied by the query?* Diverse LSRs are implied by the query. For example, except for LSR "president vs. secretary-of-state," $\langle$Barack Obama, John Kerry$\rangle$ also satisfies LSR "president vs. presidential candidate." Only in the semantic relation of "president vs. secretary-of-state," $\langle$Barack Obama, John Kerry$\rangle$ and $\langle$Bill Clinton, Madeleine Albright$\rangle$ are similar.

2. *How to find the other similar relation instances that imply the same LSR* (*e.g.*, $\langle$Bill Clinton, Madeleine Albright$\rangle$)?

Previous studies cannot solve the above challenges. First, although similarity search has been widely studied, most of previous studies are on entity similarity [16, 21]. For example, given two entities $v^{(1)}$ and $v^{(2)}$, $sim(v^{(1)}, v^{(2)})$ aims to measure the similarity between $v^{(1)}$ and $v^{(2)}$. Suppose we have two other entities $v^{(1)'}$ and $v^{(2)'}$. Relation similarity $sim(\langle v^{(1)}, v^{(2)}\rangle, \langle v^{(1)'}, v^{(2)'}\rangle)$ aims to measure the similarity between the relation instances $\langle v^{(1)}, v^{(2)}\rangle$ and $\langle v^{(1)'}, v^{(2)'}\rangle$. There is no trivial way to apply entity similarity measures to measure relation similarity. For example, "Barack Obama" is similar to "Bill Clinton" (President of United States), and "John Kerry" is similar to "John F. Kennedy" (Democratic). But $\langle$Barack Obama, John Kerry$\rangle$ is not similar to $\langle$Bill Clinton, John F. Kennedy$\rangle$ according to the LSR "president vs. secretary-of-state." Second, there are some existing methods on directly measuring similarity between two relation instances [2, 22]. However, they cannot be directly applied to the relation similarity search problem because (1) these approaches do not distinguish the diverse LSRs existing in a relation instance, and (2) they focus on measuring similarity between two relation instances represented with patterns mined from external text data, which is hard to be adapted to similarity search in schema-rich HINs.

This paper first defines a novel meta-path-based relation similarity measure, *RelSim*, to measure the similarity between two relation instances based on the LSR: two relation instances are more similar when sharing more important (heavily weighted) meta-paths. Then it provides a systematic solution to finding similar relation instances based on RelSim in schema-rich HINs for a given query.

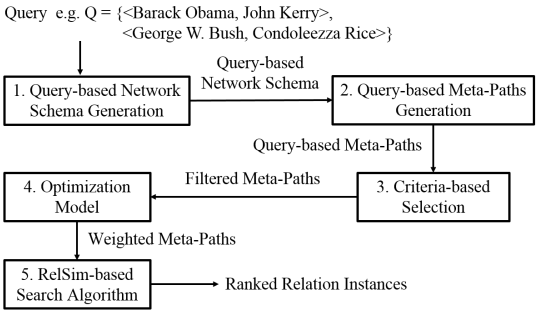


Figure 2: Relation similarity search framework.

Figure 2 illustrates our relation similarity search framework: Given a query Q, a *query-based network schema* is firstly generated (Step 1), *e.g.*, this schema can reduce the number of entity types from 1,500+ in Freebase to five types, which substantially facilitates the subsequent analysis. Then *query-based meta-paths* are generated and filtered (Steps 2 and 3). The most likely LSR is further learned based on the optimization model (Step 4), which can best explain the semantic meaning in the query through linear programming. Finally, an efficient RelSim-based similarity search algorithm is used to generate the search result (Step 5).

After Step 1, we should have a simpler network schema: the query-based network schema. Using the query to generate the query-based network schema can significantly reduce the complexity of whole network schema to the limited number of types related to the query (*e.g.*, in Freebase, the resultant query-based network schema for the query $Q = \{\langle$Barack Obama, John Kerry$\rangle, \langle$George W. Bush, Condoleezza Rice$\rangle\}$ can reduce the number of entity types from 1,500+ to five types), which substantially facilitates the subsequent analysis.

In Steps 2 and 3, the query-based meta-paths are generated based on the query-based network schema including types of entities and relations only relevant to the query, and selected based on several criteria.

In order to distinguish the various LSRs held in the query, we represent an LSR as a weighted combination of query-based meta-paths. The corresponding weights are either explicitly specified by the user or implicitly learned. In Step 4, to find the most likely LSR in the query, we propose an optimization model to learn the

weights of the query-based meta-paths, which can best explain the semantic meaning in the query through linear programming.

After having the most likely LSR, in Step 5, a RelSim-based similarity search algorithm is performed to rank the similar relation instances based on the similarity scores computed using RelSim.

To the best of our knowledge, no systematic solution to relation similarity search has been proposed in schema-rich HINs. Our contributions can be highlighted as follows:

- We study *relation similarity search in schema-rich heterogeneous information networks*, a new but very important problem due to its broad applications.
- We define a novel relation similarity measure, *RelSim*, to compute the similarity between two relation instances.
- We present a framework for relation similarity search in *schema-rich HINs*, mainly including *latent semantic relation* representation and learning, and an efficient search algorithm.
- We construct five real world datasets for relation similarity search research, and experimental results on the datasets demonstrate the effectiveness and efficiency of our approach, in comparison with baseline systems.

The remainder of the paper is organized as follows. Section 2 introduces the preliminary knowledge. Section 3 formalizes the problem. Section 4 presents the relation similarity search framework in schema-rich HINs. Experiments and results are discussed in Section 5. Section 6 discusses the related work, and we conclude this study in Section 7.

2. PRELIMINARIES

In this section, we introduce some preliminary knowledge of our proposed problem. Following [21], we define heterogeneous information network as follows.

DEFINITION 1. *A **heterogeneous information network** (HIN) is a directed graph $G = (V, E)$ with an entity type mapping ϕ: $V \to \mathcal{A}$ and a relation type mapping ψ: $E \to \mathcal{R}$, where V denotes the entity set and E denotes the link set, $\mathcal{A}$ denotes the entity type set and $\mathcal{R}$ denotes the relation type set, and the number of entity types $|\mathcal{A}| > 1$ or the number of relation types $|\mathcal{R}| > 1$.*

The network schema provides a high-level description of a given heterogeneous information network.

DEFINITION 2. *Given a heterogeneous network $G = (V, E)$ with the entity type mapping ϕ: $V \to \mathcal{A}$ and the relation type mapping ψ: $E \to \mathcal{R}$, the **network schema** for network G, denoted as $T_G = (\mathcal{A}, \mathcal{R})$, is a directed graph with nodes as entity types from $\mathcal{A}$ and edges as relation types from $\mathcal{R}$.*

Note that in most real world schema-rich HINs, such as Freebase and DBpedia, the types of entities or relations are often organized in a *conceptually hierarchical* manner. For example, *Employee* is a sub-perspective of *Organization*. *Organization* is a sub-perspective of *Business*. All the types or perspectives share a common root, called *Object*. Figure 3 depicts an example of conceptual hierarchy of entity types.

Another important concept in heterogeneous information network is *meta-path* [21], proposed to systematically define relations between entities that have different semantic meanings.

DEFINITION 3. *A **meta-path** $\mathcal{P}$ is a path defined on the graph of network schema $T_G = (\mathcal{A}, \mathcal{R})$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_L} A_{L+1}$, which defines a composite relation $R = R_1 \bullet R_2 \bullet \cdots \bullet R_L$ between types A_1 and A_{L+1}, where $\bullet$ denotes relation composition operator, and L is the length of $\mathcal{P}$.*

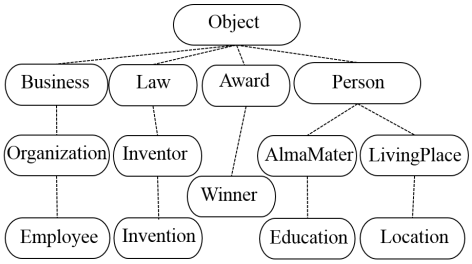


Figure 3: Conceptual hierarchy of entity types.

For simplicity, we also use type names connected by "-" to denote the meta-path when there exist no multiple relations between a pair of types: $\mathcal{P} = (A_1 - A_2 - \cdots - A_{L+1})$. For example, in the Freebase network, the composite relation *two Person co-founded an Organization* can be described as *Person* $\xrightarrow{\text{found}}$ *Organization* $\xrightarrow{\text{found}^{-1}}$ *Person*, or *Person-Organization-Person* for simplicity. We say a path $p = (v_1 - v_2 \cdots v_{L+1})$ between v_1 and v_{L+1} in network G follows the meta-path $\mathcal{P}$, if $\forall l, \phi(v_l) = A_l$ and each edge $e_l = \langle v_l, v_{l+1} \rangle$ belongs to each relation type R_l in $\mathcal{P}$. We call these paths as *path instances* of $\mathcal{P}$, denoted as $p \in \mathcal{P}$. R_l^{-1} represents the reverse order of relation R_l.

Due to the complexity of the schema of a schema-rich network, the number of meta-paths that can be generated between two entity types could be extremely large, even confined to a small meta-path length. For example, there are $49,298$ length-4 meta-paths between entity types *Person* and *Person* in our Freebase subset.

3. THE RELATION SIMILARITY SEARCH PROBLEM

We study the *relation similarity search* problem, that is, finding similar relation instances for a user query in schema-rich HINs. Given a small set of relation instances as an example query (*e.g.*, ⟨Larry Page, Sergey Brin⟩ and ⟨Jerry Yang, David Filo⟩), the system will first discover its *latent semantic relation (LSR)* (*e.g.*, "co-founders") and then output similar relation instances (*e.g.*, ⟨Bill Gates, Paul Allen⟩).

In a simple case, a query may imply a simple LSR that can be represented as a single meta-path, such as *Person* $\xrightarrow{\text{found}}$ *Organization* $\xrightarrow{\text{found}^{-1}}$ *Person*. In general, an LSR can be represented as a weighted combination of multiple meta-paths.

DEFINITION 4. *A **latent semantic relation** (LSR) is defined as a weighted combination of meta-paths, denoted as $\{w_m, \mathcal{P}_m\}_{m=1}^{M}$, where $\mathcal{P}_m$ is m^{th} meta-path and ω_m is the corresponding weight for $\mathcal{P}_m$.*

An advantage of modeling LSR as a weighted combination of meta-paths is augmenting the capability of representing different semantic meanings. For example, as shown in Figure 1, given a user query $Q = \{\langle$Larry Page, Sergey Brin$\rangle$, *etc.*$\}$, we show two meta-paths with weights between the two entities: $\mathcal{P}_1$ = *Person* $\xrightarrow{\text{found}}$ *Organization* $\xrightarrow{\text{found}^{-1}}$ *Person*, $\mathcal{P}_2$ = *Person* $\xrightarrow{\text{alma mater}}$ *Education* $\xrightarrow{\text{alma mater}^{-1}}$ *Person*. The corresponding weights are ω_1 and ω_2. If $\omega_1 > \omega_2$, there is a higher possibility that the LSR is "co-founders." If $\omega_1 = \omega_2$, the possibility of the LSR be "co-founders" is equal to be "schoolmates." If $\omega_1 < \omega_2$, there is a higher possibility that the LSR is "schoolmates." Different weighted combinations of meta-paths lead to different semantic meanings.

3.1 RelSim: A Novel Relation Similarity Measure

Although there are some existing relation similarity measures [2, 22], but they do not distinguish the diverse subtle semantic meanings in the relation instance (*i.e.*, they assume there is only one general relation held in one relation instance). For example, only with semantic meaning "co-founders," ⟨Larry Page, Sergey Brin⟩ and ⟨Bill Gates, Paul Allen⟩ are similar. When the meaning turns to "schoolmates", they are dissimilar. Here, we define a meta-path-based relation similarity measure, ***RelSim***, to measure similarity between two relation instances based on the LSR with subtle semantic meaning. The intuition behind RelSim is that if two relation instances share more heavily weighted meta-paths, they tend to be more similar.

We formally define RelSim below:

DEFINITION 5. ***RelSim: a meta-path-based relation similarity measure.*** *Given an LSR, denoted as $\{w_m, \mathcal{P}_m\}_{m=1}^{M}$, RelSim between two relation instances $r = \langle v^{(1)}, v^{(2)} \rangle$ and $r' = \langle v^{(1)'}, v^{(2)'} \rangle$ is defined as:*

$$RS(r, r') = \frac{2 \times \sum_m \omega_m \min(x_m, x'_m)}{\sum_m \omega_m x_m + \sum_m \omega_m x'_m} \quad (1)$$

where x_m is the number of path instances between $v^{(1)}$ and $v^{(2)}$ in relation r following meta-path $\mathcal{P}_m$, and x'_m is the number of path instances between $v^{(1)'}$ and $v^{(2)'}$ in relation r' following meta-path $\mathcal{P}_m$. We use a vector $\mathbf{x} = [x_1, \cdots, x_m, \cdots, x_M]$ to characterize a relation instance r, and a vector $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_m, \cdots, \omega_M]$ to denote the corresponding weights. M is the number of meta-paths.

In schema-rich HINs, the number of path instances between two entities following a specific meta-path is often 1 or 0, denoting whether the two entities satisfy the meta-path-based relation. For example, "Larry Page" and "Sergey Brin" have co-founded one organization. By looking at RelSim defined in Def. 5, we can see that $RS(r, r')$ is defined in terms of two parts: (1) the *semantic overlap* in the numerator, which is the weighted number of overlapped meta-path-based relations of r and r'; and (2) the *semantic broadness* in the denominator, which is the weighted number of total meta-path-based relations satisfied by r and r'. Note that, if the number of path instances for a meta-path is larger than 1, *i.e.*, $x_m > 1$, we treat the two entities have satisfied the relation x_m times. We can see that the larger number of overlapped meta-path-based relations shared by the r and r', the more similar the two relation instances are, which is further normalized by the semantic broadness of r and r'.

For example, let us consider two relation instances, r = ⟨Larry Page, Sergey Brin⟩ and r' = ⟨Bill Gates, Paul Allen⟩, and the LSR represented with a weighted combination of three meta-paths, including the two meta-paths mentioned above $\mathcal{P}_1$ ("co-founders") and $\mathcal{P}_2$ ("schoolmates"), and $\mathcal{P}_3 = \textit{Person} \xrightarrow{\text{invent}} \textit{Invention} \xrightarrow{\text{invent}^{-1}} \textit{Person}$ ("co-inventors"), with the weights $\boldsymbol{\omega}$ = [0.8, 0, 0.2]). By looking at the Freebase network, we can find $\mathbf{x}$ = [1, 1, 2] and $\mathbf{x}'$ = [1, 0, 1] for relation instance r and r', respectively. Thus, the number of shared meta-paths between them are 1, 0, and 1, respectively. The RelSim between these two relation instances is:

$$RS(\langle \text{Larry Page}, \text{Sergey Brin} \rangle, \langle \text{Bill Gates}, \text{Paul Allen} \rangle) = \frac{2 \times (0.8 \cdot 1 + 0 \cdot 0 + 0.2 \cdot 1)}{(0.8 \cdot 1 + 0 \cdot 1 + 0.2 \cdot 2) + (0.8 \cdot 1 + 0 \cdot 0 + 0.2 \cdot 1)} = 0.959,$$

which has shown that the two relation instances are rather similar given the LSR we are interested in.

RelSim satisfies several nice properties as indicated from properties (1) to (3).

- (1) Range: $\forall r, r', 0 \leq sim(r, r') \leq 1$.
- (2) Symmetric: $sim(r, r') = sim(r', r)$.
- (3) Self-maximum: $sim(r, r) = 1$.

PROOF. Please see Proof in the Appendix. □

3.2 Problem Definition

Given the above definitions, we now formally define the relation similarity search problem as follows.

Since we are aiming to find other relation instances similar to the ones stated in a query, we first define the RelSim between query and a relation instance as the average similarity between each relation instance in the query and the relation instance:

DEFINITION 6. *Given a user query $Q = \{r_k = \langle v_k^{(1)}, v_k^{(2)} \rangle\}, k = 1, \cdots, K$, and a relation instance r', the RelSim between Q and r' is calculated as $RS(Q, r') = \sum_k RS(r_k, r')/K$.*

Then our relation similarity search problem is to find all the top relation instances that are similar to query Q.

In schema-rich HINs, it is not trivial to (1) identify the LSR given a user query and (2) perform RelSim computation, as the number of meta-paths that can be generated is extremely large. In next section, we propose an efficient search framework to solve the problem.

4. THE RELATION SIMILARITY SEARCH FRAMEWORK

In this section, we introduce our framework for efficient relation similarity search in schema-rich HINs. Our framework, shown in Figure 2, can effectively and efficiently address the two challenges raised in the introduction, *i.e.*, identifying the LSR implied in the query (Section 4.1) and performing the RelSim computation efficiently (Section 4.2). We take the query $Q = \{$⟨Larry Page, Sergey Brin⟩, ⟨Jerry Yang, David Filo⟩$\}$ shown in Figure 1 as an example to illustrate each step. We first introduce our LSR learning method.

4.1 LSR Learning

In order to represent an LSR in a schema-rich HIN, we need to find a small number of representative query-based meta-path set beforehand, because of the following two issues:

1. It is commonsense that the real semantic meaning that user implies in a query is specific, *i.e.*, the meaning should be represented with limited number of meta-paths that focus on relevant types of entities and relations;

2. It is time consuming and impractical to automatically generate meta-paths by enumerating all the possible meta-paths between entities.

We therefore design three steps (Step 1-3 as shown in Figure 2) to address the above issues. To address the first issue, in Step 1, we construct a query-based network schema based on the user query Q, through keeping the types of entities and relations relevant to the query. In Step 2 and Step 3, we address the second issue, by generating meta-paths using an efficient query-based meta-path generation algorithm using a query-based network schema, and filtering the query-based meta-paths generated in Step 2 using three criteria.

After an LSR is represented with a smaller meta-path set, in Step 4, the weights of the most likely LSR are learned based on an optimization model using the user provided examples in Q.

We introduce the query-based network schema next.

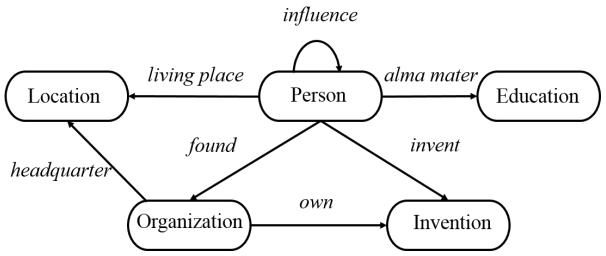


Figure 4: Query-based network schema for query $Q = \{\langle$Larry Page, Sergey Brin$\rangle$, $\langle$Jerry Yang, David Filo$\rangle\}$.

4.1.1 Query-based Network Schema

Due to the sophisticated structure of the rich network schema, for a user query which contains several relation instances (examples), it is impossible and not necessary to use the whole HIN, and thus we only keep a sub-network that is relevant to the query. For example, given the query Q in Figure 1, as illustrated in Figure 4, the entity types, such as *Person, Organization, Education*, are relevant to the query. While types like *Film, Musician, Book* are not relevant, and thus ignored. Formally, we propose the concept of query-based network schema to represent the part of entire network schema that is relevant to the query.

DEFINITION 7. *Query-based network schema. A query-based network schema is a sub-network schema of a schema-rich HIN. Given a schema-rich HIN $G = (V, E)$, a user query Q, and the diameter of schema D, D is the maximum length of hops that an entity $v \in Q$ can arrive on the graph of schema, then query-based network schema contains types of entities in the Q and within D-hop to the entities (denoted as V_u), and types of relations $\langle v^{(1)}, v^{(2)}\rangle$ ($v^{(1)}, v^{(2)} \in V_u$) (denoted as E_u). A query-based network schema is denoted as $QNS_G = (\mathcal{A}_u, \mathcal{R}_u)$, where $\mathcal{A}_u = \{\phi(v_u), v_u \in V_u\}$ and $\mathcal{R}_u = \{\psi(e_u), e_u \in E_u\}$.*

Given a query Q and the diameter of the schema D, we accordingly generate QNS_G based on Q and D as follows. First, for each example $r_k \in Q$ (Def. 6), we enumerate all the neighbor entities within d-hop ($d \leq D/2$) relations for each entity ($v_k^{(1)}$ and $v_k^{(2)}$). Next, we look up the intersection of all entity and relation types to generate the $QNS_G = (\mathcal{A}_u, \mathcal{R}_u)$, where $\mathcal{A}_u$ is the intersection set of entity types, and $\mathcal{R}_u$ is the intersection set of relation types. For example, given the query Q in the previous example, QNS_G generated by the above process is shown in Figure 4, where $\mathcal{A}_u = \{$*Person, Organization, Education, etc.*$\}$, and $\mathcal{R}_u = \{$*found, influence, alma mater, etc.*$\}$. We can see that, when using query-based network schema, the cost for generating meta-paths is reduced by confining the rich network schema.

4.1.2 Query-based Meta-Path Generation Algorithm

Most existing work assumes that meta-paths are provided by users. This assumption may be true for schema-simple HIN (*e.g.*, the DBLP network), it may be infeasible for schema-rich HIN such as the Freebase network. Besides, long meta-paths can be difficult to discover. A simple way can be proposed to automatically generate meta-paths: for a relation instance $\langle v^{(1)}, v^{(2)}\rangle$, one can generate all the possible meta-paths via enumerating all the relations, starting from $v^{(1)}$ and ending with $v^{(2)}$. However, it is time consuming and impractical. As pointed out in [16], the number of possible meta-paths grows sharply with the length of meta-path. We therefore propose an efficient query-based meta-path generation algorithm (*QMPG* in Algorithm 1) to generate meta-paths for a relation instance $\langle v^{(1)}, v^{(2)}\rangle$ based on query-based network schema.

Motivated by binary search, given a relation instance, to generate the meta-paths within length-L for the relation instance, we first generate meta-paths that within $L/2$-hop to each entity of the relation instance, and then composite the meta-paths within length-$L/2$. We build inverted indices on types of entities and relations to speed up the process. The details of the algorithm is described as below.

Algorithm 1: *QMPG*($\langle v^{(1)}, v^{(2)}\rangle$, QNS_G, L).

input : A relation instance $\langle v^{(1)}, v^{(2)}\rangle$, the query-based network schema $QNS_G = (\mathcal{A}_u, \mathcal{R}_u)$, and the maximum query-based meta-path length L.
output : A query-based meta-path set $\mathbf{P}$ for $\langle v^{(1)}, v^{(2)}\rangle$.

1 $\mathbf{P} \leftarrow \{\}$, $\mathbf{V}^{(1)} \leftarrow \{v^{(1)}\}$, $\mathbf{V}^{(2)} \leftarrow \{v^{(2)}\}$;
2 **for** $l \leftarrow 0$ **to** $L/2$ **do**
3 **for** *entity* $v^{(1)\prime}$ *within l-hop to entity* $v^{(1)}$ **do**
4 $\mathbf{V}^{(1)} \leftarrow v^{(1)\prime}$;
5 **for** *entity* $v^{(2)\prime}$ *within l-hop to entity* $v^{(2)}$ **do**
6 $\mathbf{V}^{(2)} \leftarrow v^{(2)\prime}$;
7 **for** *entity* $v \in \mathbf{V}^{(1)} \cup \mathbf{V}^{(2)}$ **do**
8 **if** $\phi(v) \in \mathcal{A}_u$ **then**
9 $\mathbf{A}(v) \leftarrow \phi(v)$; // $\mathbf{A}$ is the inverted index on entity types
10 **for** *entity* v' *within 1-hop to entity* v **do**
11 **if** $\phi(v') \in \mathcal{A}_u \vee \psi(\langle v, v'\rangle) \in \mathcal{R}_u$ **then**
12 $\mathbf{A}(v') \leftarrow \phi(v')$, $\mathbf{R}(\langle v, v'\rangle) \leftarrow \psi(\langle v, v'\rangle)$; // $\mathbf{R}$ is the inverted index on relation types
13 **for** $v^{(1)\prime} \in \mathbf{V}^{(1)}$ **do**
14 $\mathcal{P}^{(1)} \leftarrow \mathbf{A}(v^{(1)}) \circ \mathbf{R}(\langle v^{(1)}, v'\rangle) \cdots \mathbf{A}(v^{(1)\prime})$;
15 **for** $v^{(2)\prime} \in \mathbf{V}^{(2)}$ **do**
16 $\mathcal{P}^{(2)} \leftarrow \mathbf{A}(v^{(2)}) \circ \mathbf{R}(\langle v^{(2)}, v'\rangle) \cdots \mathbf{A}(v^{(2)\prime})$;
17 **if** $(\mathbf{A}(v^{(1)\prime}) == \mathbf{A}(v^{(2)\prime}))$ **then**
18 $\mathcal{P} \leftarrow \mathcal{P}^{(1)} \oplus \mathcal{P}^{(2)}$, $\mathbf{P} \leftarrow \mathbf{P} \cup \{\mathcal{P}\}$;
19 **return** $\mathbf{P}$;

Given a relation instance $\langle v^{(1)}, v^{(2)}\rangle$, the maximum length of query-based meta-path L, and QNS_G, the *QMPG* algorithm returns a set of query-based meta-paths $\mathbf{P}$ for $\langle v^{(1)}, v^{(2)}\rangle$. First, in Lines 1-6, we retrieve all the neighbor entities within $L/2$-hop to $v^{(1)}$ and $v^{(2)}$, denoted as $v^{(1)\prime}$ and $v^{(2)\prime}$, and build indices $\mathbf{V}^{(1)}$ and $\mathbf{V}^{(2)}$ for $v^{(1)}$ and $v^{(1)\prime}$, $v^{(2)}$ and $v^{(2)\prime}$.

Second, in Lines 7-12, for each entity v in $\mathbf{V}^{(1)} \cup \mathbf{V}^{(2)}$, we check its types $\phi(v)$. We build an inverted index $\mathbf{A}$ on the types of entities, when $\mathcal{A}_u$ contains the type. For each 1-hop neighbor entity v', we build an inverted index when the types of entities are included in the $\mathcal{A}_u$. An inverted index $\mathbf{R}$ is also built on relation types between v and v', if the relation type is included in $\mathcal{R}_u$.

Next, in Lines 13-19, we generate query-based meta-paths between the entities and their neighbor entities (*e.g.*, $\mathcal{P}^{(1)}$ denotes a meta-path from $v^{(1)}$ to $v^{(1)\prime}$), where $\circ$ represents the composition operator between types of entities and relations. Then, a meta-path $\mathcal{P}$ is generated by compositing $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$, if $v^{(1)\prime}$ and $v^{(2)\prime}$ are sharing the same entity type, where $\oplus$ is the composition operator on entity types. $\mathcal{P}$ is appended to the query-based meta-path set $\mathbf{P}$. Finally, $\mathbf{P}$ is returned for the given relation instance $\langle v^{(1)}, v^{(2)}\rangle$.

For example, given the query Q in Figure 1 and $L = 2$, we generate meta-paths for the example $\langle$Larry Page, Sergey Brin$\rangle$ based on QNS_G as shown in Figure 4. First, we retrieve the

neighbor entities within 1-hop to "Larry Page" and "Sergey Brin", build an index for "Larry Page" and neighbor entities as $\mathbf{V}^{(1)} = \{$Larry Page, Google, Stanford University, PageRank, Mountain View, Google Hacks, *etc.*$\}$, and an index for "Sergey Brin" and neighbor entities as $\mathbf{V}^{(2)} = \{$Sergey Brin, Google, Stanford University, PageRank, IT, Eric Schmidt, Mountain View, Male, *etc.*$\}$. Second, an inverted index $\mathbf{A}$ is built on entity types in QNS_G (*e.g.*, $\mathbf{A}$(Larry Page) = *Person*). Notice that "Google Hacks" and "Male" are not indexed, since their entity types (*Book* and *Gender*) are irrelevant. An inverted index $\mathbf{R}$ is built on 1-hop relations between entities in $\mathbf{V}^{(1)}$ and $\mathbf{V}^{(2)}$ (*e.g.*, $\mathbf{R}(\langle$Larry Page, Stanford University$\rangle)$ = *alma mater*). Then a query-based meta-path $\mathcal{P}$ = *Person* $\xrightarrow{\text{invest}}$ *Organization* $\xrightarrow{\text{employee in}^{-1}}$ *Person* is generated by connecting $\mathcal{P}^{(1)}$ = *Person* $\xrightarrow{\text{invest}}$ *Organization* and $\mathcal{P}^{(2)}$ = *Person* $\xrightarrow{\text{employee in}}$ *Organization*, since they share the same type *Organization*, where a path instance *Larry Page-Google-Sergey Brin* follows the meta-path. These multi-hop meta-paths (with length larger than one) are quite useful for discovering missing links between two entities. There are no direct connections between them. Finally, the query-based meta-path set $\mathbf{P}$ is generated. We show six query-based meta-paths in $\mathbf{P}$ in Table 1.

Meta-Path No.	Meta-Paths
$\mathcal{P}_1$	*Person* $\xrightarrow{\text{found}}$ *Organization* $\xrightarrow{\text{found}^{-1}}$ *Person*
$\mathcal{P}_2$	*Person* $\xrightarrow{\text{alma mater}}$ *Education* $\xrightarrow{\text{alma mater}^{-1}}$ *Person*
$\mathcal{P}_3$	*Person* $\xrightarrow{\text{invent}}$ *Invention* $\xrightarrow{\text{invent}^{-1}}$ *Person*
$\mathcal{P}_4$	*Person* $\xrightarrow{\text{invest}}$ *Organization* $\xrightarrow{\text{employee in}^{-1}}$ *Person*
$\mathcal{P}_5$	*Person* $\xrightarrow{\text{win}}$ *Award* $\xrightarrow{\text{win}^{-1}}$ *Person*
$\mathcal{P}_6$	*Person* $\xrightarrow{\text{birthdate year}}$ *Year* $\xrightarrow{\text{birthdate year}^{-1}}$ *Person*

Table 1: Example meta-paths generated for the query Q shown in Figure 1.

Notice that, we manually translate the hierarchy of relations in a meta-path into natural language for better understanding. For example, we translate the relation *Business/Organization/Employee* to *employee in* in $\mathcal{P}_4$. We use "/" to represent the hierarchy of relations. As illustrated in Figure 3, *Business* represents the most general concept (at the top) of the hierarchy. While *Employee* represents the most concrete concept (at the bottom) of the hierarchy. These meta-paths are of different hierarchical semantic levels, and used to explain the LSRs held in the query. We use all the meta-paths with different hierarchical semantics without distinguishing them, and will explore better ways in the future.

Moreover, with larger L, more query-based meta-paths will be generated. More meta-paths may provide us more semantics about the LSRs held in the relation instance. But in the real world, long meta-paths are not that meaningful, because they often reflect weaker semantic connections between entities.

4.1.3 Criteria for Query-based Meta-Path Selection

We assume an LSR could be expressed with a small number of (weighted) representative query-based meta-paths. In practice, the size of $\mathbf{P}$ ($|\mathbf{P}|$) is large. It is necessary to select such a set out of $\mathbf{P}$ to keep the useful meta-paths but remove the useless or noisy ones. We use $\mathbf{P}'$ to denote the query-based meta-path set selected from $\mathbf{P}$.

Given a user query Q, and $\mathbf{P} = \cup_{k=1}^{K}\mathbf{P}_k$ where $\mathbf{P}_k$ is the meta-path set for k^{th} example $r_k \in Q$ generated by *QMPG*, we include query-based meta-paths in $\mathbf{P}'$ satisfying at least one of the following criteria:

Criterion 1. *Select* $\mathcal{P}_m \in \mathbf{P}$, *s.t.* $\sum_k \mathbf{x}_{km} > \theta_1$

where $\mathbf{x}_{km}$ is the number of path instances between $v_k^{(1)}$ and $v_k^{(2)}$ of r_k following query-based meta-path $\mathcal{P}_m$. This criterion suggests that if the total number of path instances in Q is beyond the threshold θ_1, the corresponding meta-path $\mathcal{P}_m$ should be included in $\mathbf{P}'$. Similar to *term frequency* used in information retrieval, this criterion aims to find meta-paths that are important for the query. For example, since $\mathcal{P}_6$ co-occurs only once with the examples in Q, it should not be in $\mathbf{P}'$.

Criterion 2. *Select* $\mathcal{P}_m \in \mathbf{P}$, *s.t.* $information\_gain\_ratio(X_m) > \theta_2$

where information gain ratio [1] is a widely used technique for feature selection [17]. We model the m^{th} feature ($\mathcal{P}_m$) as a random variable X_m. The intuition is that if a meta-path $\mathcal{P}_m$ has a higher information gain ratio, it will better split the positive and negative examples (positive and negative examples will be described in Section 4.1.4). The criterion prefers to select meta-paths with higher information gain ratio (beyond the threshold θ_2) into $\mathbf{P}'$.

Criterion 3. *Select* $\mathcal{P}_m \in \mathbf{P}$, *s.t.* $\exists \mathcal{P}_n \in \mathbf{P}'$, $KL(Pr_m||Pr_n) < \theta_3$

where two distributions $Pr_m \triangleq P(X_m)$ and $Pr_n \triangleq P(X_n)$. X_n is a random variable for n^{th} feature ($\mathcal{P}_n$). Distributional hypothesis, which states that words that occurred in the same contexts tend to have similar meanings [10], has been used for identifying similar words. We make an assumption that it is an extension to the distributional hypothesis: *if the instances of two meta-paths tend to similarly distribute over examples in a query, then the two meta-paths tend to be similar.* The degree of similarity is measured using the Kullback-Leibler (KL) divergence between two distributions Pr_m and Pr_n. This criterion infers $\mathcal{P}_m$ in $\mathbf{P}'$ when $\mathcal{P}_n$ has already been included in $\mathbf{P}'$, because they are similarly distributed over examples in a query (*i.e.*, KL under the threshold θ_3).

We will discuss the settings of three parameters, θ_1, θ_2, and θ_3, in the experiment section.

4.1.4 Optimization Model

Our system finds the most likely LSR implied in the query by learning the best weights of the selected query-based meta-paths in $\mathbf{P}'$. To express the user's need, it is easier for her to provide a query of several examples, and let system learn the weight of each meta-path automatically, rather than specify the weights of them.

Given a query Q, and filtered query-based meta-path set $\mathbf{P}'$, we propose an optimization model to learn the weight of each meta-path $\mathcal{P} \in \mathbf{P}'$. We assume one or several meta-paths in $\mathbf{P}'$ can capture the most likely LSR held in the query. For example, given a user's query Q as shown in Figure 1, it's probable that the most likely LSR is a combination of two meta-paths, *e.g.*, $\mathcal{P}_1$ and $\mathcal{P}_2$, indicating that *two Person co-founded an Organization, and both of them are from the same Education Institute*. Our task is to discover such important query-based meta-paths by optimizing the weights.

The difficulty of understanding the LSR is that there is a lot of background noise. For example, in Table 1, $\langle$Larry Page, Sergey Brin$\rangle$ and $\langle$Jerry Yang, David Filo$\rangle$ both have the meta-path $\mathcal{P}_1$. But at the same time, they also share meta-paths like $\mathcal{P}_4$, which is a less important meta-path. $\mathcal{P}_4$ can be considered as background noise, since randomly choosing a relation between *Person* and *Person* may have a higher possibility to satisfy $\mathcal{P}_4$. For example, in Figure 1, "Larry Page" and "Paul Allen" do not share the important meta-paths, such as $\mathcal{P}_1$, with the examples in Q. We call such artificial pairs (*e.g.*, $\langle$Larry Page, Paul Allen$\rangle$) as "negative examples."[2]

[2]Sometimes, negative examples may accidentally share meta-paths

Formally, the negative examples are generated by randomly replacing the subject ($v_k^{(1)}$) (object ($v_k^{(2)}$)) entity of one relation instance by the subject (object) entity of another. A relation instance may have multiple negative examples. We hope to maximize the weights of query-based meta-paths that are mainly shared by positive examples (*i.e.*, examples in Q), but never or rarely appear in negative examples.

Denote $K = |Q|$ as the number of examples in the user query, and $M = |\mathbf{P}'|$ as the number of selected query-based meta-paths. Then, each relation instance would have a feature vector of length M, which is denoted as $\mathbf{x}_k$ ($k = 1, \cdots, K$). The m^{th} element of $\mathbf{x}_k$ is the number of path instances between $v_k^{(1)}$ and $v_k^{(2)}$ of $r_k \in Q$. We also denote $\tilde{\mathbf{x}}_k$ as a negative (or corrupted) example. We use L_2 norm to normalize all feature vectors.

We assign each meta-path a weight ω_m ($m = 1, \cdots, M$), $\omega_m \geq 0$ and regularize $\sum_{m=1}^{M} \omega_m = 1$. Then given the relation instances and the negative examples, we try to find a set of weights in which "important" meta-paths have higher weights, while "unimportant" ones near 0. Inspired by the *ranking loss* proposed as Eq. (17) in [4], we propose the following optimization model:

$$\min_{\boldsymbol{\omega}} \quad \sum_{k=1}^{K} \max\{0, c - \boldsymbol{\omega}^T \mathbf{x}_k + \boldsymbol{\omega}^T \tilde{\mathbf{x}}_k\} \tag{2}$$

$$\text{s.t.} \quad \omega_m \geq 0 \qquad \forall m = 1, \cdots, M$$

$$\sum_{m=1}^{M} \omega_m = 1 \tag{3}$$

where $c \in (0, 1]$ is a tuning parameter. If $c = 1$, then we have $\max\{0, c - \boldsymbol{\omega}^T \mathbf{x}_k + \boldsymbol{\omega}^T \tilde{\mathbf{x}}_k\} = 1 - \boldsymbol{\omega}^T \mathbf{x}_k + \boldsymbol{\omega}^T \tilde{\mathbf{x}}_k$ (since $\mathbf{x}_k - \tilde{\mathbf{x}}_k$ is going to be a vector with each entry smaller than or equal to 1 after normalization, with the constraint $\sum_{m=1}^{M} \omega_m = 1$, we then have $\boldsymbol{\omega}^T(\mathbf{x}_k - \tilde{\mathbf{x}}_k) \leq 1$). As a result, this model will essentially maximize the weights of meta-paths that have the biggest difference between positive and negative examples. If $c < 1$, then the model will consider the accident that positive and negative examples share the important meta-paths, and that some of the important meta-paths are missing in some positive examples. Following our example, $\mathbf{P}' = \{\mathcal{P}_m, m \in (1, \cdots, 5)\}$ as shown in Table 1, and the learned weights $\boldsymbol{\omega} = [0.45, 0.25, 0.15, 0.05, 0.1]$, it demonstrates that $\mathcal{P}_1$ is more important than $\mathcal{P}_4$ to represent the most appropriate LSR implied by Q, because negative examples (*e.g.*, ⟨Larry Page, Paul Allen⟩) may have a higher possibility to satisfy $\mathcal{P}_4$ compared to $\mathcal{P}_1$.

By introducing slack variables $\alpha_k = \max\{0, c - \boldsymbol{\omega}^T \mathbf{x}_k + \boldsymbol{\omega}^T \tilde{\mathbf{x}}_k\}$, the above optimization problem can be turned into linear programming with $(M + K)$ variables and $(M + 1 + 2K)$ constraints:

$$\min_{\boldsymbol{\omega}, \alpha} \quad \sum_{k=1}^{K} \alpha_k \tag{4}$$

$$\text{s.t.} \quad \omega_m \geq 0 \qquad \forall m = 1, \cdots, M \qquad \sum_{m=1}^{M} \omega_m = 1$$

$$\alpha_k \geq 0 \qquad \alpha_k \geq c - \boldsymbol{\omega}^T \mathbf{x}_k + \boldsymbol{\omega}^T \tilde{\mathbf{x}}_k \qquad \forall k = 1, \cdots, K$$

We use the interior point method (Chapter 11 in [3]) to solve the above linear programming problem.

Now, we have a weighted query-based meta-path set $\mathbf{P}'$, each $\mathcal{P}_m \in \mathbf{P}'$ is associated with corresponding weight ω_m. We consider this weighted combination of query-based meta-paths as the LSR held in the query.

with positive examples. But we have demonstrated the effectiveness by comparing that with the human provided negative examples in the experiment.

4.2 Efficient RelSim-based Similarity Search

In schema-rich HINs, it is time consuming and impractical to perform relation similarity search for online queries in the scope of the whole network[3]. Notice that, most of relation instances do not share any common meta-paths with the query, thus there is no need to search all the possible relation instances in the HIN, which is both time and space expensive.

After having the learned LSR, now we revisit our framework shown in Figure 2, in Step 5, a fast RelSim-based relation similarity search algorithm (*FRS* in Algorithm 2) is used to efficiently generate the search result ranked by the similarity scores computed using RelSim. The intuition of the algorithm is that the search space can be extremely pruned if we only search for the candidates that have at least one common meta-path with the LSR. We build an inverted index on meta-paths to speed up the searching process.

Algorithm 2: *FRS*(G, Q, QNS_G, L, $\mathbf{P}'$, $\boldsymbol{\omega}$).

input : A schema-rich HIN $G = (V, E)$, a user query Q, the query-based network schema $QNS_G = (\mathcal{A}_u, \mathcal{R}_u)$, the maximum query-based meta-path length L, the selected meta-path set $\mathbf{P}'$, and the weights $\boldsymbol{\omega}$ of query-based meta-paths in $\mathbf{P}'$.
output : A ranked list $\mathbf{RL}$ of similar relation instances for Q.

1 $\mathbf{CR} \leftarrow \{\}$, $\mathbf{RL} \leftarrow \{\}$; // $\mathbf{CR}$ is the candidate relation instance set, $\mathbf{RL}$ is the final ranked list of similar relation instances
2 **for** $v \in V$ **do**
3 **if** $\phi(v) \in \mathcal{A}_u$ **then**
4 $\mathbf{A}'(\phi(v)) \leftarrow v$; // $\mathbf{A}'$ is the inverted index on entities
5 **for** *entity* $v \in \mathbf{A}'(\phi(v_k^{(1)}))$ **do**
6 **for** *entity* $v' \in \mathbf{A}'(\phi(v_k^{(2)}))$ **do**
7 $\mathbf{P}^{\langle v, v' \rangle} \leftarrow QMPG(\langle v, v' \rangle, QNS_G, L)$;
8 **for** $\mathcal{P} \in \mathbf{P}'$ **do**
9 **if** $\mathcal{P} \in \mathbf{P}^{\langle v, v' \rangle}$ **then**
10 $\mathbf{CR} \leftarrow \mathbf{CR} \cup \{\langle v, v' \rangle\}$;
11 **for** *candidate relation instance* $\langle v, v' \rangle \in CR$ **do**
12 $sim_{score} \leftarrow RS(Q, \langle v, v' \rangle)$;
13 $\mathbf{RL} \leftarrow \mathbf{RL} \cup \{(\langle v, v' \rangle, sim_{score})\}$;
14 SORT($\mathbf{RL}$, sim_{score});
15 **return** $\mathbf{RL}$;

Given an HIN $G = (V, E)$, a user query Q, the query-based network schema $QNS_G = (\mathcal{A}_u, \mathcal{R}_u)$, and the maximum length of query-based meta-path L, *FRS* illustrates how we perform relation similarity search for online queries, in order to return a ranked list $\mathbf{RL}$ of similar relation instances.

First, we build an inverted index $\mathbf{A}'$ on entities (Line 1-4). Second, the candidate relation instances $\mathbf{CR}$ are generated (Line 5-10), after enumerating all the entity pairs $\langle v, v' \rangle$ that share the same entity types with the entities in query (*i.e.*, $\phi(v_k^{(1)})$ and $\phi(v_k^{(2)})$). $\langle v, v' \rangle$ is inserted into $\mathbf{CR}$ as a candidate, if the meta-path set $\mathbf{P}^{\langle v, v' \rangle}$ (containing meta-paths for $\langle v, v' \rangle$) includes at least one meta-path $\mathcal{P} \in \mathbf{P}'$. Next, we compute the relation similarity by using RelSim $RS(Q, \langle v, v' \rangle)$ defined in Def. 6, and sort the candidates according to the similarity scores (Line 11-14). Finally, the sorted list of candidates is returned (Line 15).

For example, two examples in Q in Figure 1 are characterized as $\mathbf{x}_1 = [1, 1, 2, 1, 3]$ and $\mathbf{x}_2 = [1, 1, 0, 1, 1]$. $r' = \langle$Bill Gates, Paul Allen$\rangle$, $r'' = \langle$Steve Ballmer, Mark Zuckerberg$\rangle$ and $r''' = \langle$Steve Jobs, Steve Wozniak$\rangle$ are retrieved as three candidates, where

[3]See https://groups.google.com/forum/#forum/freebase-discuss.

Relation Categories	#Entities	#Relations	Examples
⟨*Organization, Founder*⟩	9,836,649	560,688,893	⟨Google, Larry Page⟩, ⟨Microsoft, Bill Gates⟩, ⟨Facebook, Mark Zuckerberg⟩
⟨*Book, Author*⟩	16,640,478	981,788,232	⟨Gone with the Wind, Margaret Mitchell⟩, ⟨The Kite Runner, Khaled Hosseini⟩
⟨*Actor, Film*⟩	4,340,986	182,121,412	⟨Leonardo DiCaprio, Inception⟩, ⟨Daniel Radcliffe, Harry Potter⟩, ⟨Jack Nicholson, Head⟩
⟨*Location, Contains*⟩	1,037,791	62,229,669	⟨United States of America, New York⟩, ⟨Victoria, Chillingollah⟩, ⟨New Mexico, Davis House⟩
⟨*Music, Track*⟩	1,653,931	86,658,343	⟨My Worlds, Baby⟩, ⟨21, Someone Like You⟩, ⟨Thriller, Beat It⟩
Total	26,841,657	1,483,834,223	⟨Google, Larry Page⟩, ⟨Leonardo DiCaprio, Inception⟩, ⟨Thriller, Beat It⟩

Table 2: **Rel-Full** dataset statistics. #Entities means the number of entities; #Relations means the number of relations.

$\mathbf{x}' = [1, 0, 1, 2, 3]$, $\mathbf{x}'' = [0, 1, 0, 0, 0]$, $\mathbf{x}''' = [1, 0, 3, 1, 2]$ characterize them respectively. With the LSR ($\{w_m, \mathcal{P}_m\}_{m=1}^{5}$) obtained in previous section, the similarity scores between Q and r', r'', r''', are calculated based on RelSim (0.903, 0.483, and 0.827, respectively). Thus, the ranking of each candidate in the result is: ⟨Bill Gates, Paul Allen⟩ (Rank-1), ⟨Steve Jobs, Steve Wozniak⟩ (Rank-2), and ⟨Steve Ballmer, Mark Zuckerberg⟩ (Rank-3).

In summary, our framework improves the efficiency of relation similarity search in schema-rich HINs from four aspects: (1) Query-based network schema is used to preserve a small part of the whole network schema that is relevant to the query (Section 4.1.1); (2) *QMPG* is proposed to speed up the procedure of query-based meta-path generation (Section 4.1.2); (3) We use three criteria to select important meta-paths, and reduce the search space (Section 4.1.3); (4) An efficient relation similarity search algorithm uses inverted index to further reduce the time cost (Section 4.2).

5. EXPERIMENTS

In this section, we evaluate the effectiveness and efficiency of our proposed approach.

5.1 Datasets

We use five subsets of Freebase data as our evaluation datasets. We first construct a dataset called **Rel-Full** as follows: We select five popular relation categories in Freebase, ⟨*Organization, Founder*⟩, ⟨*Book, Author*⟩, ⟨*Actor, Film*⟩, ⟨*Location, Contains*⟩, and ⟨*Music, Track*⟩. For each relation category, we randomly sample 5,000 entity pairs, then enumerate all the neighbor entities and relations within 2-hop of each entity. In Table 2, we show statistics of the five relation categories in **Rel-Full**, including the number of entities, relations, and some corresponding examples. We employ Freebase API[4] to crawl the Freebase network. The entities and relations can be obtained by parsing the Freebase data in Json format.

Besides, for efficiency study, we generate four smaller datasets based on **Rel-Full**: (1) **Rel-100**: contains 243,134 entities and 19,858,802 relations, generated based on 100 entity pairs (each relation category has 20 entity pairs); (2) **Rel-200**: contains 561,389 entities and 32,519,062 relations, generated based on 200 entity pairs (each relation category has 40 entity pairs); (3) **Rel-500**: contains 991,498 entities and 103,681,534 relations, generated based on 500 entity pairs (each relation category has 100 entity pairs); (4) **Rel-1000**: contains 1,621,711 entities and 171,871,099 relations, generated based on 1,000 entity pairs (each relation category has 200 entity pairs);

We randomly generate 10 user queries from each relation category in **Rel-Full** by sampling 5 relation instances for each query. As a result, there are 50 queries in total.

5.2 Effectiveness Study

We first study the effectiveness of our framework and query-based meta-path generation algorithm.

[4]https://developers.google.com/freebase/v1/

5.2.1 Analysis of Similarity Search Performance

We herein test the performance of our framework for relation similarity search. P@$\mathcal{K}$ (precision at $\mathcal{K}$), NDCG@$\mathcal{K}$ are used as the evaluation measures. P@$\mathcal{K}$ is the percentage of relevant results at the given value of $\mathcal{K}$ in the search result. NDCG@$\mathcal{K}$ is the normalized discounted cumulative gain at the given value of $\mathcal{K}$ in the search result. Both measures assume value between 0 and 1, and a higher value indicates a better search result. We use three baseline systems as below.

(1) Vector-Space-Model-based Similarity Search (*VSM-S*): We use the relation similarity function defined by vector space model (VSM) [23] in our framework (Line 15 in *FRS*). Each relation instance is represented using a vector of predefined lexical pattern frequency. The relation similarity between two relation instances is computed based on cosine similarity between the two vectors representing the two relation instances.

(2) Latent-Relational-Analysis-based Similarity Search (*LRA-S*): We use the relation similarity function defined by latent relational analysis (LRA) [22] in our framework. First, a matrix is generated with rows representing relation instances and columns representing patterns. The value for the cell in i^{th} row and j^{th} column is the frequency of the j^{th} lexical pattern for i^{th} relation instance collected from the result of search engine. Then, singular value decomposition (SVD) is applied to the matrix, which reduces the number of columns. Finally, the relation similarity between two relation instances is calculated using the cosine angle between two corresponding rows in the matrix.

(3) ImplicitWeb-based Similarity Search (*IW-S*): The relation similarity measure proposed in [2] is used in our framework. A supervised approach is proposed to learn a Mahalanobis distance metric between relation instances. Each relation instance is represented with a vector of lexical patterns. Next, a sequential pattern clustering algorithm is used to cluster similar patterns. Finally, an information-theoretic metric learning algorithm [5] is used to compute the similarity.

We re-implement all of the above baseline systems, by replacing the lexical patterns with query-based meta-paths. Notice that, we apply the whole meta-path set $\mathbf{P}$ to *VSM-S*, *LRA-S* and *IW-S*. In *LRA-S*, we reduce the size of $\mathbf{P}$ to 100 following [22]. While in *IW-S*, we cluster the meta-paths with the same parameter setting in [2].

We denote *RelSim-WS* the framework with RelSim as the similarity measure, and the weight of each query-based meta-path in $\mathbf{P}'$ is learned by the optimization model. Further, *RelSim-S* is *RelSim-WS* without weight learning by setting all each meta-path with equal weight.

First, we manually label the top-20 results for the 50 queries, to test the quality of ranking lists given by the five systems. We label each candidate relation instance with three relevant levels: 0 (non-relevant), 1 (some-relevant), and 2 (very relevant). We report the average P@$\mathcal{K}$ and NDCG@$\mathcal{K}$ for the 50 queries. Notice that P@$\mathcal{K}$ is evaluated based on whether the candidate is relevant (1 or 2) or not. Table 3 shows the quality of top-$\mathcal{K}$ ($\mathcal{K} = 5, 10, 20$)

Query: {⟨Google, Larry Page⟩, ⟨Microsoft, Bill Gates⟩, ⟨Facebook, Mark Zuckerberg⟩, ⟨Yahoo!, Jerry Yang⟩, ⟨DreamWorks Animation, David Geffen⟩}					
Rank	*VSM-S*	*LRA-S*	*IW-S*	*RelSim-S*	*RelSim-WS*
1	⟨Forbes, Forbes⟩	⟨Yelp, Inc., Simmons⟩	⟨Image Comics, Silvestri⟩	⟨DoubleClick, Merriman⟩	⟨Apple, Jobs⟩
2	⟨U-Haul, Shoen⟩	⟨Image Comics, Silvestri⟩	⟨Walt Disney, Disney⟩	⟨YouTube, Chen⟩	⟨IBM, Watson⟩
3	⟨HealthGrades, Hicks⟩	⟨U-Haul, Shoen⟩	⟨Forbes, Forbes⟩	⟨Apple, Wozniak⟩	⟨YouTube, Chen⟩
4	⟨Perot Systems, Perot⟩	⟨Forbes, Forbes⟩	⟨HealthGrades, Hicks⟩	⟨McDonald, McDonald⟩	⟨Linkedin, Hoffman⟩
5	⟨Image Comics, Silvestri⟩	⟨Perot Systems, Perot⟩	⟨New York Library, Dewey⟩	⟨Ford Motor, Ford⟩	⟨DoubleClick, Merriman⟩

Table 4: Case study on top-5 relation similarity search results on **Rel-Full**.

	P@5	P@10	P@20
VSM-S	0.4132	0.5657	0.7471
LRA-S	0.4414	0.6458	0.8292
IW-S	0.3627	0.4597	0.6047
RelSim-S	0.5778	0.7922	0.8416
RelSim-WS	**0.6642**	**0.8383**	**0.8715**

(a) P@$\mathcal{K}$.

	NDCG@5	NDCG@10	NDCG@20
VSM-S	0.5389	0.6296	0.7225
LRA-S	0.5880	0.6848	0.7814
IW-S	0.5210	0.6095	0.7010
RelSim-S	0.6395	0.7427	0.8432
RelSim-WS	**0.6651**	**0.7716**	**0.9559**

(b) NDCG@$\mathcal{K}$.

Table 3: Performance of relation similarity search on **Rel-Full**.

search result. From the result, we can see that RelSim-based systems (*RelSim-WS* and *RelSim-S*) outperform the baseline systems. The reasons are as follows: (1) *RelSim-WS* can better use the semantics in schema-rich HINs because it automatically learns the weights of different meta-paths; (2) Both *RelSim-WS* and *RelSim-S* consider more subtle semantics by incorporating the number of shared meta-paths of two relation instances, rather than just normalizing the total number of meta-paths like most cosine based relation similarity measures do (*e.g.*, *VSM-S*); (3) Both *RelSim-WS* and *RelSim-S* make use of three criteria (shown in Section 4.1.3) to remove the noise in meta-paths to ensure most representative ones are used to find similar relation instances. Significance is measured using the t-test with p-value < 0.001.

Then, a case study on top-5 search result is shown in Table 4, under the query $Q = \{\langle$ Google, Larry Page$\rangle$, ⟨Microsoft, Bill Gates⟩, ⟨Facebook, Mark Zuckerberg⟩, ⟨Yahoo!, Jerry Yang⟩, ⟨DreamWorks Animation, David Geffen⟩$\}$. Due to the space limitation, we just show the last name of entities. The most likely LSR held in Q is *(1) the Founder of Organization, (2) who also wins award in the same industry that the Organization runs business in*. The two most important query-based meta-paths below are used to represent the LSR, *Organization* $\xrightarrow{\text{is founded by}}$ *Founder* ($\omega = 0.384$), *Organization* $\xrightarrow{\text{run business in}}$ *Industry* $\xrightarrow{\text{win award in}^{-1}}$ *Founder* ($\omega = 0.274$). From the results, we can see that both *RelSim-WS* and *RelSim-S* get more reasonable results than the baseline systems. Although the results of the baseline systems contain the semantics (1) in Q, most of them do not imply the semantics (2). For example, in the search result generated by *IW-S*, "Walt Disney" is not an IT company, but at the second ranking position. The result shows that *RelSim-WS* gives the best ranking quality in terms of the human intuition, which is consistent with the previous result.

We further test our approach in a more realistic scenario. We use human generated negative examples (five negative examples for each query) to replace the randomly generated ones for learning the weights using our optimization model. The results on the five relation categories are NDCG@5 = 0.6968, NDCG@10 = 0.7976, NDCG@20 = 0.9712. The results indicate the random negative examples could hurt the search performance by accidentally introducing some errors (positive examples or useless ones), but it still works in a relatively good manner.

5.2.2 Case Study of Query-based Meta-Paths

One of our major contributions is that by representing the LSRs with a set of weighted query-based meta-paths, we are able to distinguish the diverse semantics of LSRs held in a user query.

Table 5a shows the top four (heavily weighted) meta-paths with the corresponding weights, for six queries sampled from the five relation categories[5]. We can see that all the important meta-paths make sense. Besides length-1 meta-paths, we can derive multi-hop meta-paths that are unexpected yet quite important semantics held in the query. For example, given the query {⟨Google, Larry Page⟩, *etc.*}, we can derive meta-paths like *Organization* $\xrightarrow{\text{run business in}}$ *Industry* $\xrightarrow{\text{win award in}^{-1}}$ *Founder* with length larger than one, and it is possible to find related relation instances w.r.t. the multi-hop meta-paths. Interestingly, for queries that have complex semantics, which can not be expressed with length-1 meta-paths, we could express the LSRs between them using multi-hop meta-paths, where originally there are no connections between the entities. For example, given the query {⟨Lord Voldemort, J. K. Rowling⟩, *etc.*}, there is no length-1 meta-paths connecting them, but we are able to use *Character* $\xrightarrow{\text{appear in}}$ *Book* $\xrightarrow{\text{write}^{-1}}$ *Author* to explain the LSR *Character is in a Book, which is written by Author*.

Moreover, we show a running example of the optimization model by providing different queries containing some common examples, as shown in Table 5b. In the query, ⟨Google, Larry Page⟩ implies different LSRs, such as "is founded by" and "runs by CEO", which are represented with different weighted combination of meta-paths. By providing different examples, such as ⟨Microsoft, Bill Gates⟩ (only satisfies "is founded by") and ⟨Yahoo!, Marissa Mayer⟩ (only satisfies "runs by CEO"), one can see that the meta-paths as well as the weight of the same meta-path change accordingly, which indicate the LSR changes from "is founded by" to "runs by CEO". The optimization model is able to distinguish the diverse LSRs.

5.3 Efficiency Study

Here, we investigate the efficiency of our framework. First, we check the impacts of query-based network schema on the execution time of *FRS*. We fix the maximum length of query-based meta-paths to $L = 4$, and conduct experiments on datasets **Rel-100**, **Rel-200**, **Rel-500** and **Rel-1000**. We compare average execution time over 50 queries with different diameter D ($D = 1, 2, 3, 4$). Each query is executed 5 times, and the result is summarized in Figure 5. *RelSim-WS* D and *RelSim-WS-QNS* represent the framework with query-based network schema of diameter D and without query-based network schema, respectively. From the result, one can see

[5]For category ⟨*Organization, Founder*⟩, we sample two queries shown in Table 5b.

Query: {⟨Harry Potter and the Philosopher's Stone, J. K. Rowling⟩, ⟨The Lord of the Rings: The Return of the King, J. R. R. Tolkien⟩, *etc.*}	ω
$Book \xrightarrow{\text{is part of}} Series \xrightarrow{\text{write}^{-1}} Author$	0.569
$Book \xrightarrow{\text{win award}} Award \xrightarrow{\text{award winner}} Author$	0.213
$Book \xrightarrow{'\text{s genre}} Genre \xrightarrow{\text{written genre}^{-1}} Author$	0.134
$Book \xrightarrow{\text{is written in}} Location \xrightarrow{\text{is birthplace of}} Author$	0.056
Query: {⟨United States of America, New York⟩, ⟨United Kingdom, London⟩, *etc.*}	ω
$Location_1 \xrightarrow{\text{in state}} State \xrightarrow{'\text{s capital}} Location_2$	0.613
$Location_1 \xrightarrow{'\text{s capital}} Location_2$	0.226
$Location_1 \xrightarrow{'\text{s government}} Government \xrightarrow{'\text{s jurisdiction}} Location_2$	0.083
$Location_1 \xrightarrow{'\text{s nationality}^{-1}} Organization \xrightarrow{\text{is contained by}} Location_2$	0.049
Query: {⟨Impulse, Oberdeck⟩, ⟨21, Someone Like You⟩, *etc.*}	ω
$Music \xrightarrow{\text{release}} Track$	0.477
$Music \xrightarrow{\text{make}^{-1}} Person \xrightarrow{\text{same height}} Person \xrightarrow{\text{perform in}} Video \xrightarrow{\text{play in TV}^{-1}} Track$	0.234
$Music \xrightarrow{\text{release}} Album \xrightarrow{\text{release}} Track$	0.09
$Music \xrightarrow{\text{release}} Track\ \ List \xrightarrow{\text{record}} Track$	0.084
Query: {⟨Jack Nicholson, Head⟩, ⟨Leonardo DiCaprio, Inception⟩, *etc.*}	ω
$Actor \xrightarrow{\text{act in}} Film$	0.294
$Actor \xrightarrow{\text{act in}} Film \xrightarrow{\text{peformance type}} Type \xrightarrow{'\text{s special}} Film$	0.282
$Actor \xrightarrow{\text{award winner}^{-1}} Award \xrightarrow{\text{win award}^{-1}} Film$	0.179
$Actor \xrightarrow{\text{perform in}} Film$	0.109

(a) Most important four query-based meta-paths of different queries.

Query: {⟨Google, Larry Page⟩, ⟨Microsoft, Bill Gates⟩, *etc.*}	ω
$Organization \xrightarrow{\text{is founded by}} Founder$	0.384
$Organization \xrightarrow{\text{run business in}} Industry \xrightarrow{\text{win award in}^{-1}} Founder$	0.274
$Organization \xrightarrow{\text{is founded by}} Person \xrightarrow{\text{is influence peer}^{-1}} Founder$	0.174
$Organization \xrightarrow{'\text{s leadership}} Person \xrightarrow{\text{mailing address}} Location \xrightarrow{\text{mailing address}^{-1}} Founder$	0.115
Query: {⟨Google, Larry Page⟩, ⟨Yahoo!, Marissa Mayer⟩, *etc.*}	ω
$Organization \xrightarrow{\text{run by}} CEO \xrightarrow{\text{job title}} Founder$	0.32
$Organization \xrightarrow{\text{founded date}} Date \xrightarrow{\text{graduation date}^{-1}} Founder$	0.229
$Organization \xrightarrow{\text{headquarter}} Location \xrightarrow{\text{education institute}} Founder$	0.207
$Organization \xrightarrow{\text{run business in}} Industry \xrightarrow{\text{win award in}^{-1}} Founder$	0.113

(b) Query-based meta-paths generated based on different queries.

Table 5: Examples for query-based meta-paths on **Rel-Full**. We manually translate the relations from Freebase into natural language for better understanding.

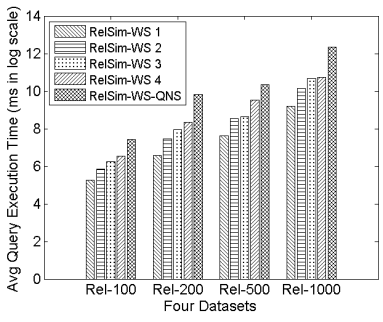


Figure 5: Average query execution time of *FRS* on four datasets with/without query-based network schema.

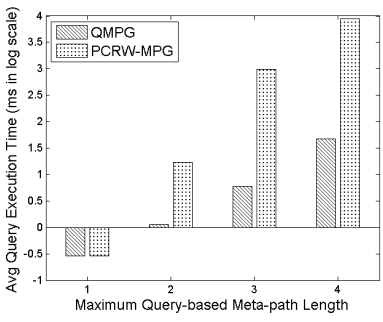


Figure 6: *QMPG* vs. *PCRW-MPG* with different maximum length on **Rel-Full**.

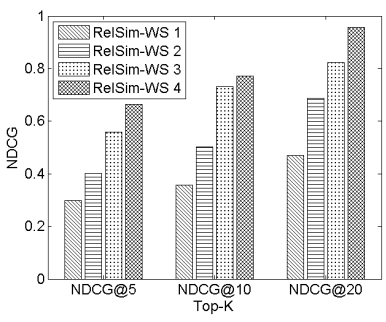


Figure 7: Parameter study of query-based network schema with different diameters.

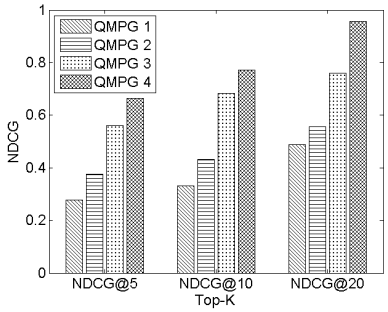


Figure 8: Parameter study of *QMPG* with different maximum lengths.

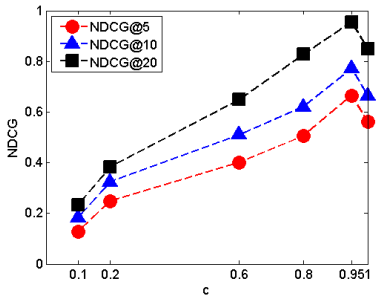


Figure 9: Parameter study of c in optimization model.

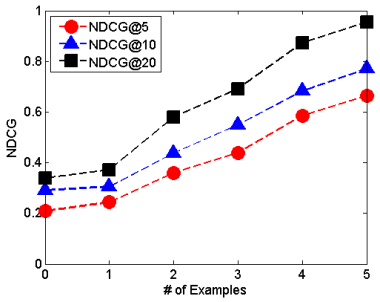
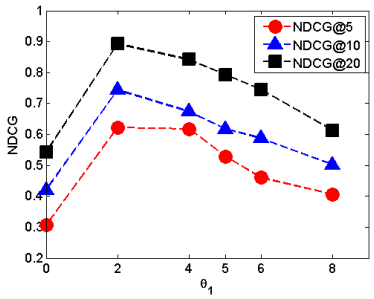
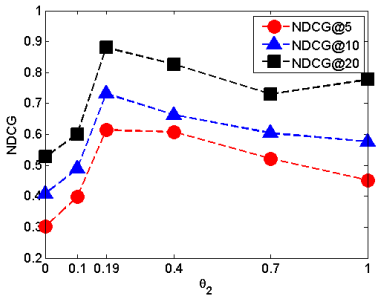


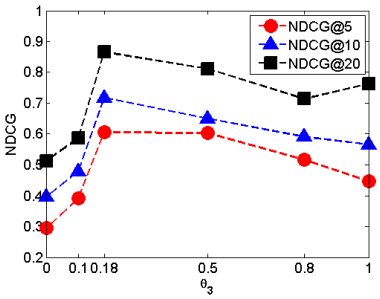
Figure 10: Parameter study of different # of examples (K) in query.



(a) Criterion 1 parameter θ_1.



(b) Criterion 2 parameter θ_2.



(c) Criterion 3 parameter θ_3.

Figure 11: Parameter study of criteria.

(1) Query-based network schema improves the efficiency in time nearly two orders of magnitudes. (2) The improvement rate is related to the diameter D. The larger D is, the improvement is more significant. (3) The improvement depends on the number of candidate relation instances. The more candidates are, the improvement is more significant.

Next, we compare the power of *QMPG* with the query-based meta-path generation method proposed in [16] (*PCRW-MPG*) (Figure 6). We fix the diameter of query-based network schema $D = 4$, varying maximum length of query-based meta-path L ($L = 1, 2, 3, 4$) for both methods, and test on **Rel-Full**. Each query is executed 5 times and the output time is the total average time of the 50 queries. The results show that *QMPG* can significantly improve the efficiency of query-based meta-path generation, by at most $\mathbf{89.6}\%$ compared to *PCRW-MPG*.

5.4 Parameter Study

In this sub-section, we study the impacts of parameters on the framework.

We first test the impact of various maximum query-based meta-path length L ($L = 1, 2, 3, 4$) on query-based network schema and *QMPG*. Figure 7 and Figure 8 show the relation between the end-to-end search performance of our framework, with the various diameters of query-based network schema and maximum query-based meta-path lengths for *QMPG*, respectively. *RelSim-WS* D ($D = 1, 2, 3, 4$) represents the *RelSim-WS* with different diameters for query-based network schema generation. *QMPG* L ($L = 1, 2, 3, 4$) represents the *QMPG* with different maximum length of meta-path. From the results, we can see that (1) with the larger diameter D, the higher improvement query-based network schema achieves. (2) The longer the maximum meta-path length L is, the more improvement. In practice, we set $D = 4$ and $L = 4$, because if D and L get larger, the number of meta-paths will be prohibitive.

We next evaluate the three parameters of the criteria, θ_1, θ_2, θ_3, for query-based meta-path filtering. From the result in Figure 11, we can see that in general the larger value each parameter has, the more improvement it can achieve. But it hurts the performance when the value beyond certain threshold. The reason is that a larger criterion threshold may accidentally filter out important meta-paths. We set $\theta_1 = 2$, $\theta_2 = 0.19$, $\theta_3 = 0.18$ in our experiments.

We then study the influence of the parameter in the optimization model, c, on our framework. As illustrated in Figure 9, one can see that with the larger c, the more it improves in general cases. We experimentally set $c = 0.95$ because there is a slight slope after the point. The reason is that when $c < 1$, the model will consider the accident that positive and negative examples share the important meta-paths, and that one of the important meta-paths is missing in some positive examples.

Finally, we investigate the impact of the number of examples (K) in query on the search results. Figure 10 shows that when providing more similar examples in a query, the general end-to-end performance will be improved more. In our experiment, we set $K = 5$, because it is difficult to ask a user to provide too many examples in real world.

6. RELATED WORK

Subgraph querying. There have been many works on subgraph querying [11, 27, 29] based on traditional subgraph isomorphism using identical label matching. However, we focus on the semantic similarity of graph structure, which does not require identical

match. The subgraph querying is enriched with entity similarity and ontology in [8, 24, 25]. A similarity matrix between query entities and related ontology (or data entities) are assumed as input, and the data nodes dissimilar with the query entities or related ontology are filtered according to a threshold and ranked accordingly. Our study provides a new perspective by using relation similarity instead of entity or ontology-based similarity. The network data for subgraph querying is often stored in relational databases, graph databases or triplestores [15]. To retrieve data from these databases, the standard is often to use structured query languages such as SQL, SPARQL. However, writing structured queries requires extensive experience in query language and data model, and good understanding of particular datasets [12]. We do not assume users have such domain knowledge. Instead, we only require users to provide examples of relation instances.

Query by example. Query by example is well studied in relational databases. Querying paradigms for graphs can be categorized into keyword-based [19], interactive and visual interfaces [6]. These works require structured queries, for example, query graphs or patterns [28], meta-paths [21] or structured query languages, explicitly based on the known underlying schema. Recently, unstructured queries have been studied [13] without schema. In contrast, our system allows unstructured queries as examples to query the network by incorporating network schema. As a result, we relieve the burden of users for providing structured queries, as well as improving the quality of results by utilizing the knowledge about the queries embedded in the network schema.

Similarity measures. Similarity measures have been a hot research topic for years. They can be categorized broadly into two types: entity similarity measures and relation similarity measures. Similarity measures, such as SimRank [14], P-Rank [30], PathSim [21], and PCRW [16] capture entity similarity. There exist works on measuring relation similarity [18, 22]. They usually generate a matrix with rows representing entity pairs and columns representing patterns, extracted from text data. Then certain similarity function, like cosine similarity, is applied to calculate the relation similarity by using the two corresponding rows in the matrix. We improve these studies from two aspects: First, our approach distinguishes the diverse latent semantic relations existing in a relation instance; Second, we are able to utilize the rich link information in HINs.

7. CONCLUSION AND DISCUSSION

We study relation similarity search in schema-rich heterogeneous information networks. In order to solve the problem, we need to (1) correctly identify the most likely LSR implied by the input query, and (2) provide an efficient search algorithm that can answer the query in a real-time mode. We propose a framework to address the two requirements. In the framework, we first represent an LSR as a weighted combination of query-based meta-paths that are generated based on query-based network schema. Second, a novel meta-path-based relation similarity measure RelSim is introduced and used in an efficient similarity search algorithm. The experimental results on the five real world datasets demonstrate the power of our approach. Our approach can be used in many applications, such as relation based clustering, classification and recommendation, *etc.*. For example, we can use RelSim in the clustering algorithms to canonicalize similar relations.

8. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *TKDE*, 5(6):914–925, 1993.

[2] D. T. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring the similarity between implicit semantic relations from the web. In *WWW*, pages 651–660, 2009.

[3] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[5] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216. ACM, 2007.

[6] E. Demidova, X. Zhou, and W. Nejdl. Freeq: an interactive query interface for freebase. In *WWW*, pages 325–328, 2012.

[7] X. L. Dong, K. Murphy, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610, 2014.

[8] W. Fan, X. Wang, and Y. Wu. Incremental graph pattern matching. *TODS*, 38(3):18, 2013.

[9] S. Gu, J. Yan, L. Ji, S. Yan, J. Huang, N. Liu, Y. Chen, and Z. Chen. Cross domain random walk for query intent pattern mining from search engine log. In *ICDM*, pages 221–230. IEEE, 2011.

[10] Z. S. Harris. Distributional structure. *Word*, 1954.

[11] H. He and A. K. Singh. Closure-tree: An index structure for graph queries. In *ICDE*, pages 38–38. IEEE, 2006.

[12] H. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *SIGMOD*, pages 13–24. ACM, 2007.

[13] N. Jayaram, M. Gupta, A. Khan, C. Li, X. Yan, and R. Elmasri. Gqbe: Querying knowledge graphs by example entity tuples. In *ICDE*, pages 1250–1253, 2014.

[14] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[15] A. Khan, Y. Wu, and X. Yan. Emerging graph queries in linked data. In *ICDE*, pages 1218–1221, 2012.

[16] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[17] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1997.

[18] P. Nakov and M. A. Hearst. Solving relational similarity problems using the web as a corpus. In *ACL*, pages 452–460, 2008.

[19] J. Pound, I. F. Ilyas, and G. Weddell. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *SIGMOD*, pages 423–434, 2010.

[20] Y. Sun and J. Han. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012.

[21] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*, pages 992–1003, 2011.

[22] P. Turney. Measuring semantic similarity by latent relational analysis. In *IJCAI*, pages 1136–1141, 2005.

[23] P. Turney, M. L. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *RANLP*, pages 482–486, 2003.

[24] V. Vassilevska and R. Williams. Finding, minimizing, and counting weighted subgraphs. In *STOC*, pages 455–464. ACM, 2009.

[25] Y. Wu, S. Yang, and X. Yan. Ontology-based subgraph querying. In *ICDE*, pages 697–708. IEEE, 2013.

[26] C. Xiao, W. Wang, X. Lin, and H. Shang. Top-k set similarity joins. In *ICDE*, pages 916–927. IEEE, 2009.

[27] X. Yan, P. S. Yu, and J. Han. Graph indexing: a frequent structure-based approach. In *SIGMOD*, pages 335–346. ACM, 2004.

[28] X. Yu, Y. Sun, P. Zhao, and J. Han. Query-driven discovery of semantically similar substructures in heterogeneous networks. In *KDD*, pages 1500–1503. ACM, 2012.

[29] S. Zhang, M. Hu, and J. Yang. Treepi: A novel graph indexing method. In *ICDE*, pages 966–975, 2007.

[30] P. Zhao, J. Han, and Y. Sun. P-rank: a comprehensive structural similarity measure over information networks. In *CIKM*, pages 553–562, 2009.

APPENDIX

ReSim satisfies properties (1) to (3).

- (1) Range: $\forall r, r', 0 \leq sim(r, r') \leq 1$. This is because $\omega_m, x_m, x'_m \geq 0$, and $2 \times \min(x_m, x'_m) \leq (x_m + x'_m)\ \forall m$.
- (2) Symmetric: $sim(r, r') = sim(r', r)$. This is because $\min(x_m, x'_m)$ and $x_m + x'_m$ are symmetric.
- (3) Self-maximum: $sim(r, r) = 1$. This is because $2 \times \min(x_m, x'_m) \leq (x_m + x'_m)$.