How to Build Templates for RDF Question/Answering —An Uncertain Graph Similarity Join Approach

Weiguo Zheng
Peking University
Beijing, China
zhengweiguo@pku.edu.cn

Lei Zou
Peking University
Beijing, China
zoulei@pku.edu.cn

Xiang Lian
University of Texas - Pan American
Texas, USA
lianx@utpa.edu

Jeffrey Xu Yu
The Chinese Univ. of Hong Kong
Hong Kong, China
yu@se.cuhk.edu.hk

Shaoxu Song
Tsinghua University
Beijing, China
sxsong@tsinghua.edu.cn

Dongyan Zhao
Peking University
Beijing, China
zhaody@pku.edu.cn

ABSTRACT

A challenging task in the natural language question answering (Q/A for short) over RDF knowledge graph is how to bridge the gap between unstructured natural language questions (NLQ) and graph-structured RDF data (G). One of the effective tools is the "template", which is often used in many existing RDF Q/A systems. However, few of them study how to generate templates automatically. To the best of our knowledge, we are the first to propose a join approach for template generation. Given a workload D of SPARQL queries and a set N of natural language questions, the goal is to find some pairs $\langle q, n \rangle$, for $q \in D \wedge n \in N$, where SPARQL query q is the best match for natural language question n. These pairs provide promising hints for automatic template generation. Due to the ambiguity of the natural languages, we model the problem above as an uncertain graph join task. We propose several structural and probability pruning techniques to speed up joining. Extensive experiments over real RDF Q/A benchmark datasets confirm both the effectiveness and efficiency of our approach.

1. INTRODUCTION

Recently, knowledge graphs have attracted lots of attentions in both academia and industry. As Resource Description Framework (RDF) is de facto the standard of a knowledge graph, we focus on RDF repository in this paper. Given a large RDF knowledge graph G, a key issue is how to access G and quickly obtain the desired information. Although SPARQL is a structural query language over RDF graphs, it is impractical for non-professional users to query RDF graphs using SPARQL, because of the complexity of the SPARQL syntax and RDF schema. We illustrate a SPARQL query example over DBpedia as follows.

```
SELECT ?person WHERE {
 ?person rdf:type Artist.
 ?person graduatedFrom Harvard_University.
}
```

1.1 Motivation

To hide the complexity of SPARQL syntax, RDF question/answering (Q/A) systems provide an easy-to-use interface for users, which has attracted extensive attention in both NLP (natural language processing) [19, 23, 20] and DB (database) communities [24, 33]. A challenging task is how to translate natural language questions (NLQ) into structural queries, such as SPARQL, over a large knowledge graph G. One of the effective tools is the "template", which is used in many existing RDF Q/A systems [19, 20].

A typical system is EVI (http://www.evi.com/, formally TrueKnowledge), which logged over one million users within four months of launch in January 2012. EVI was acquired by Amazon in October 2012, and is now part of the Amazon group of companies. EVI aims to directly answer questions posed in plain English text over knowledge base. It employs a template-based approach to answer natural language questions, where a template describes how to turn a class of natural language questions into correct structural queries. However, EVI has to manually define the translation templates [19].

It is clear that the quality of templates determines the answer quality. The challenge is how to generate a large number of high quality templates automatically. Few previous works study this issue but all have to manually define the templates [19, 20]. Obviously, it is expensive to manually define these templates, especially for open-domain Q/A systems over large-scale RDF knowledge graphs. In this paper, we study how to generate templates automatically.

Figure 1 presents the framework for RDF Q/A using templates. It consists of two tasks: *how to generate templates* and *how to use these templates in RDF Q/A*.

To address the first issue (the focus of this paper), we propose a query workload-based approach for automatic template generation. Specifically, the inputs for the template generation task are two query workloads, as shown in Figure 1. One is a set D of SPARQL queries on the RDF repository (such as DBpedia workload[1]). The other one is a

[1]aksw.org/Projects/DBPSB.html

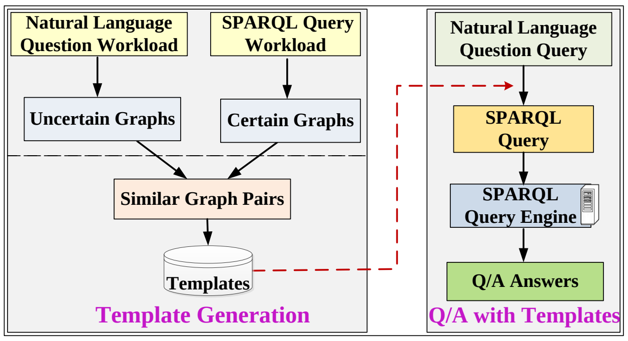


Figure 1: Framework for Template-based RDF Q/A

set N of natural language questions, which can be collected from some community-driven question-and-answer (Q&A) sites (such as Yahoo Answers) or search engine query workload (such as WebQuestions[2]). The whole process is done in three steps as follows.

Step 1. Uncertain Graph Generation. According to the method in [33], natural language questions can be interpreted into semantic query graphs. Due to the ambiguity of the interpretation, we model the semantic query graph as an uncertain graph, i.e, each vertex/edge has multiple possible labels with different probabilities. Here, labels are uncertain due to semantic ambiguity. For example, let us consider the question "which actor from USA is married to Michael Jordan born in a city of NY" in Figure 2. There are three persons named "Michael Jordan" in the DBpedia RDF graph, i.e., an NBA star, a professor, and an actor, each of which is associated with a probability. "NY" also has two possible mappings, i.e., state and city. Therefore, we can obtain an uncertain graph containing nodes with multiple labels, each associated with a probability.

Step 2. Finding Similar Graph Pairs. The goal is to find some pairs $\langle q, g\rangle$, for $q \in D \wedge g \in U$, where D corresponds to the SPARQL queries and U are the uncertain graphs derived from the natural language questions.[3] Here, we utilize graph edit distance, which is widely used to measure the similarity of graphs, to determine the similarity of q and g.

Step 3. Generating Templates with Similar Graph Pairs. Given a returned pair $\langle q, g\rangle$, based on the mapping between q and g (the mapping is found when computing graph edit distance at Step 2), we can build the templates.

In order to generate templates, the key problem is how to find the similar graph pairs efficiently (i.e. Step 2). This is the focus of this paper. We propose a novel approach, i.e., Similarity Join over uncertain graphs, denoted as *SimJ* (formally defined in Def. 7). Specifically, we propose a series of effective bounds to improve the performance.

Given the templates generated, a natural language question is translated into a SPARQL query using the templates generated in the first task. Then, the SPARQL query is issued over the knowledge graphs. The existing systems, such as Jena [2], RDF-3x [15] and Virtuoso[4], can be utilized.

1.2 Challenges and Our Contributions

Challenge 1: *Effectiveness*. Since the templates are built based on the returned graph pairs, we need to guarantee that the returned pairs $\langle q, n\rangle$ are similar to each other. In our method, we represent the natural language question n as an uncertain graph g, due to the ambiguity of natural language phrases. To tackle this challenge, we propose a notion, called the *similarity probability*, $SimP_\tau(q, g)$, to measure the probability that (uncertain) graph g and q are similar under graph edit distance constraints, which will be formally defined in Def. 6. Our experiments on DBpedia SPARQL query workload and WebQuestions natural language question workload generate more than 8,000 templates with precision 86.54%. More experimental results on other real datasets are reported in Section 7.

Challenge 2: *Efficiency*. Another critical issue is related to efficient *SimJ* processing. To our best knowledge, no prior works studied this *SimJ* problem in the context of uncertain graphs. One straightforward method to solve the *SimJ* problem is to employ the *possible world* semantics [18, 3] to uncertain graphs, where each *possible world* is a materialized instance of uncertain graphs. However, since there are an exponential number of possible worlds, it is rather inefficient to materialize all possible worlds and perform graph similarity join on them. Therefore, efficient computations of *SimJ* answers are challenging and non-trivial.

There are several existing lower bounds for graph edit distances in the literature, such as *c-star* in [29], *k-Adjacent Tree* in [21], *paths* in [31], and *Pars* in [30]. However, these bounds are based on the deterministic vertex/edge labels. If we utilize them in uncertain graphs, there are two extreme solutions. One is to enumerate all possible worlds of an uncertain graph and compute lower bounds for each possible world. The other one is to ignore all vertex/edge labels and only consider structures of graphs. Obviously, the former is not efficient due to an exponential number of possible worlds; while the latter's pruning power is low. It is not straightforward to revise the existing lower bounds to provide both efficient and effective pruning in uncertain graphs.

We derive a novel edit distance lower bound (called CSS-based lower bound) that utilizes the relationship between graph edit distance (GED) and *common structural subgraph* (CSS) in the uncertain graph context. An advantage of CSS-based bound is that there is a uniform CSS-based lower bound between q and all possible worlds of uncertain graph g (see Theorem 3). In other words, we do not have to enumerate all possible worlds of g. Our lower bound considers structure and uncertainty together instead of using them separately. Compared with existing lower bounds, our bound is tighter, and can thus achieve higher pruning power.

Applying the relationship between GED and CSS, we also design an upper bound for similarity probability between q and g. Section 5 discusses more details about that.

To summarize, we make the following contributions in this work.

- We propose a graph similarity join (SimJ) approach to generate SPARQL query templates for RDF Q/A automatically. Due to the inherent ambiguity of the natural language phrases, we study SimJ in the context of uncertain graphs.

- None of existing lower bounds of GED can be applied in SimJ over uncertain graphs. We propose a novel lower bound that is independent on the possible worlds of a uncertain graph, which is based on the relationship between GED and common structural subgraphs.

[2]A natural language question log, the website is www-nlp.stanford.edu/software/sempre/
[3]We focus on the basic graph patterns of OPT-free SPARQL queries and do not handle other questions.
[4]http://virtuoso.openlinksw.com/

- We devise an upper bound for similarity probability between q and g to prune unpromising candidates.
- To optimize query performance, we further propose a cost-based technique. That is, dividing uncertain graphs into fine-grained groups so as to attain higher pruning power.
- We conduct extensive experiments over real (including QALD-3 benchmark) datasets to confirm the effectiveness and efficiency of our proposed approach. The precisions on QALD-3 and WebQuestions datasets achieve 97.67% and 86.54%, respectively.

2. OVERVIEW AND RUNNING EXAMPLE

2.1 Template Generation

As mentioned in Section 1, there are three steps to build templates. We illustrate the detailed steps.

Step 1: Uncertain Graph Generation. There are two phases, i.e., generating semantic query graphs and assigning uncertain labels, in order to generate uncertain graphs for natural language questions (NLQ).

Generating semantic query graphs. Consider each question n in the NLQ workload. n is translated into a *semantic query graph* (Def. 1) using the approach in [33]. A semantic query graph is a collection of *semantic relations*. A semantic relation is a triple $\langle rel, arg1, arg2\rangle$, where rel is a relation phrase, $arg1$ and $arg2$ are the two argument phrases.

Definition 1. (**Semantic Query Graph** [33]). A semantic query graph is denoted as Q^S, in which each vertex v_i is associated with an argument and each edge $\overline{v_i v_j}$ is associated with a relation phrase, $1 \leq i, j \leq |V(Q^S)|$.

Let us consider a natural language question "Which actor from USA is married to Michael Jordan born in city of NY ?" in Figure 2. We can extract four semantic relations from the question, i.e., ⟨"from", "which actor", "USA"⟩, ⟨"be married to", "which actor", "Michael Jordan" ⟩, ⟨"born in", "Michael Jordan", "city"⟩, ⟨"located in", "city", "NY"⟩, each of which is represented as an edge in the semantic query graph Q^S. Two edges share one common vertex if the corresponding relations share one common argument in Q^S.

Assigning Uncertain Vertex Labels. The uncertain labels of Q^S is from the ambiguity of natural language phrases. Each vertex in Q^S is an argument arg, which is a natural language phrase, such as "Michael Jordan" and "NY". Applying entity linking techniques [4], an argument arg (i.e., a vertex) in Q^S may be linked to multiple entities associated with different existence confidences. We use the corresponding type of entities to denote the vertex label. For example, "Michael Jordan" in Q^S may correspond to three different persons, i.e., an NBA player, a professor and an actor. Therefore, we replace "Michael Jordan" in Q^S (i.e., vertex v_2) with three possible classes, i.e., ⟨NBA_playrer⟩, ⟨Professor⟩ and ⟨Actor⟩, each of which is associated with an existence confidence. Many entity linking algorithms can provide the confidence probability, such as [4]. Furthermore, all variable vertices are assigned the same label (a wildcard label), i.e., all the labels starting with "?" can match any vertex label.

Assigning Uncertain Edge Labels. Similarly, one relation phase rel in Q^S may correspond to multiple predicates [33]. Zou

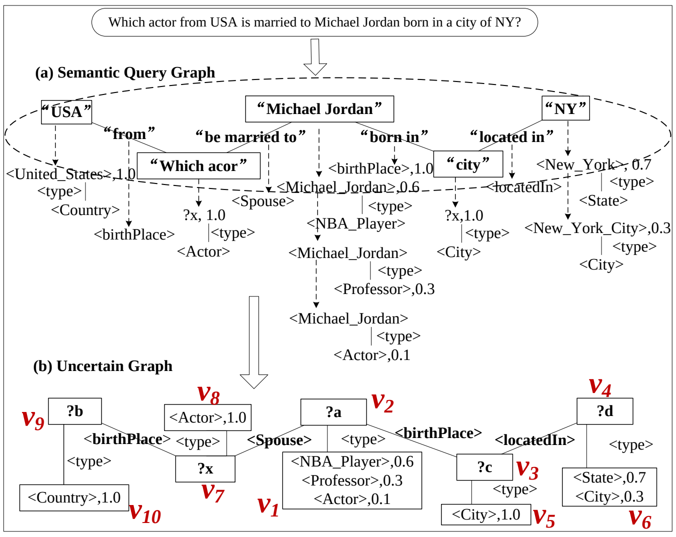


Figure 2: Uncertain graph generation

et al. proposed a graph mining-based relation paraphrasing algorithm [33]. Thus, an edge in Q^S may also have multiple labels, each of which is a predicate associated with a confidence probability.

Step 2: Finding Similar Graph Pairs. Let U denote all uncertain graphs that are generated from natural language question (NLQ) workload in the first step. It is straightforward to represent each SPARQL query (in the SPARQL workload) as a certain graph q, all of which are collected to form a certain graph dataset D. As shown in Figure 3, our goal of this step is to find similar graph pairs $\langle q, g\rangle$, where $q \in D$ and $g \in U$ are similar to each other with high probability in the uncertain model.

We use minimum graph edit distance (GED for short), a widely used error-tolerant measure, to evaluate the graph similarity. Finding similar graph pairs by joining D with U is challenging, since computing GED is an NP-hard problem [29] and an uncertain graph $g \in U$ has an exponential number of possible worlds. The existing lower bounds of GED cannot be used directly unless that we materialize all possible worlds or ignore all vertex/edge labels of g, as these lower bounds are designed for certain graphs. To improve the performance, a careful-designed lower bound for a certain graph q (derived from a SPARQL query) and an uncertain graph g (derived from a natural language question) is desirable.

In this paper, we propose a common structural subgraph (CSS for short)-based lower bound, which avoids enumerating possible worlds of g and also considering the vertex/edge labels. The technical details are given in Sections 3 to 6. In the running example, we find two similar graph pairs $\langle g_1, q_2\rangle$ and $\langle g_2, q_1\rangle$. Note that the similar graph pair does not necessarily mean that the two queries (such as g_1 and q_2) ask for the same question, but they potentially generate a good template (see the next step) based on the matching between g_1 and q_2 under the graph edit distance constraint.

Step 3: Generating Templates with Similar Graph Pairs. Assume that we have obtained the similar graph pairs (such as $\langle g_1, q_2\rangle$ and $\langle g_2, q_1\rangle$), the next step is to build the templates automatically. We illustrate the detailed process using the running example. Consider the graph pair $\langle g_2, q_1\rangle$ in Figure 3, where g_2 corresponds to the second question "Which politician graduated from CIT?".

Figure 4(b) gives the uncertain graph g_2 that is generated based on the semantic query graph as shown in Figure 4(a). In computing graph edit distance, we obtain the mapping between the uncertain graph g_2 and SPARQL query q_1. The dashed lines in Figure 4(b) and (c) illustrate the mappings between g_2 and q_1, such as $\langle politician \leftrightarrow Artist \rangle$, $\langle CIT \leftrightarrow Harvard\_University \rangle$. It also shows the mappings between phrases in natural language questions and the entities/classes in SPARQL queries. We replace these specific entities/classes as slots and also keep the mappings, which lead to some templates, as shown in Figure 4(d).

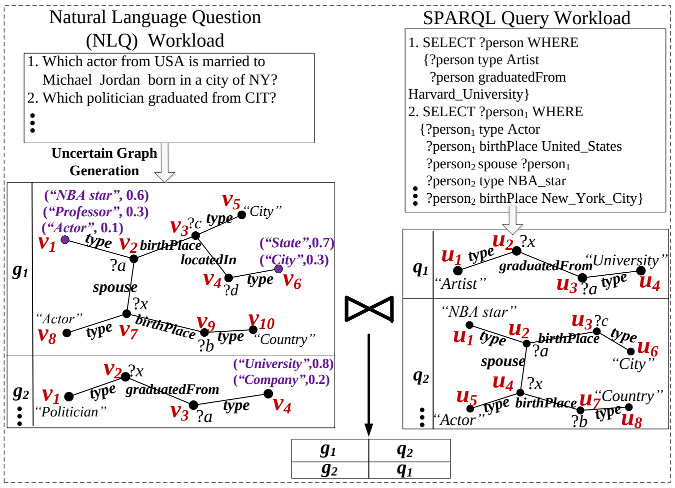
2.2 Q/A with Templates

Given a new natural language question n, if a template t can match the question n, we translate n into a SPARQL query using the template. Then, any SPARQL query engine can be used to answer the SPARQL query. There are two main steps: *finding a template matching the question, slot filling*, and *entity linking.*

Finding a template matching the question and slot filling. Given a natural language question, we first find which natural language template can match the question. Then, based on the match, we can find the mapping between the phrases (in natural language question) and the slots (in templates). Here we propose a syntactic dependency tree based alignment for the slot filling task.

Syntactic dependency tree based alignment. First, by using the NLP (Natural Language Processing) parsers, e.g., Stanford Parser [13], the natural language question and the natural language part of templates are parsed into syntactic dependency trees, respectively. Many tree edit distance (TED)-based approaches have been proposed to find the alignment between two dependency trees in NLP literature, such as [6, 9, 26]. Based on these existing methods, we try to find a template's dependency tree that best aligns with the dependency tree of the natural language question sentence (i.e., the minimum tree edit distance). Once the alignment is found, it is straightforward to fill the slot with the corresponding phrases in the natural language question sentences. We give an example to illustrate the whole process.

Example 1. *Given a natural language question "Which physicist graduated from CMU?" and a natural language part of a query template "Which ⟨__⟩ graduated from ⟨__⟩?", they are parsed into two syntactic dependency trees as shown in Figure 5. By aligning these two syntactic dependency trees,*

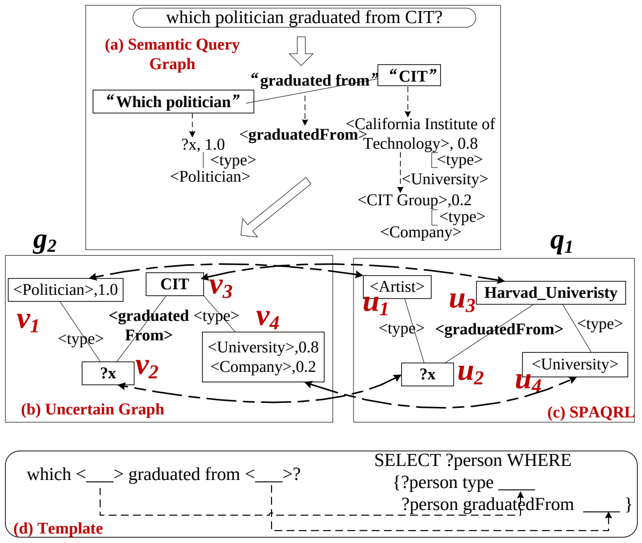


1. Which actor from USA is married to Michael Jordan born in a city of NY?
2. Which politician graduated from CIT?

```
1. SELECT ?person WHERE
   {?person type Artist
    ?person graduatedFrom
Harvard_University}
2. SELECT ?person1 WHERE
   {?person1 type Actor
    ?person1 birthPlace United_States
    ?person2 spouse ?person1
    ?person2 type NBA_star
    ?person2 birthPlace New_York_City}
```

g_1	q_2
g_2	q_1

Figure 3: Finding Similar Graph Pairs



```
which <___> graduated from <___>?
SELECT ?person WHERE
{?person type ____
 ?person graduatedFrom ____ }
```

Figure 4: Template Generation.

we can fill the slots according to the mapping.

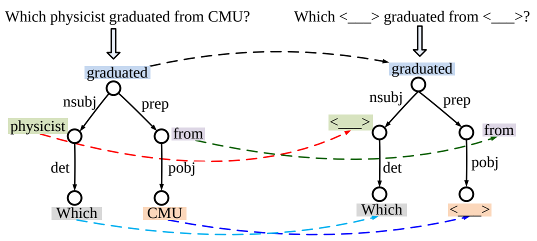

Which physicist graduated from CMU?
Which <___> graduated from <___>?
graduated
nsubj
prep
physicist
from
det
pobj
Which
CMU
<___>

Figure 5: Syntactic dependency graph alignment

Entity Linking. Obtaining the mapping between entities and slots, we need to find the entities (in the underlying knowledge graph) that correspond to phrases filled at the slot. Actually, this is a classical entity linking problem, which has been extensively studied [4, 16, 32]. When it is done, we have obtained a complete SPARQL statement.

3. FINDING SIMILAR GRAPH PAIRS

We have outlined the whole process of our solution. As mentioned earlier, a challenging task is Step 2 of the template generation, i.e., how to find similar graph pairs from an uncertain graph set U (derived from the NLQ workload) and a certain graph set D (derived from the SPARQL query workload). We model this task as a graph similarity join problem. From this section, we focus on finding similar graph problem. In this section, we first give some preliminary knowledge in Section 3.1 and then formalize finding similar graph pair problem (denoted as SimJ) in Section 3.2 and discuss the steps of computing SimJ problem in Section 3.3. Table 1 lists the major notations in this paper.

Table 1: Notations and Descriptions

Notation	Description
D	certain graphs corresponding to SPARQL queries
U	uncertain graphs derived from language questions
q	a (certain) query graph, $q \in D$
g	an uncertain graph, $g \in U$
$pw(g)$ or g^c	possible world of uncertain graph g
$Pr\{pw(g)\}$	the appearance probability of $pw(g)$
τ	user-specified threshold of GED
α	user-specified threshold of similarity probability
$ged(\cdot,\cdot)$	the GED between two (un)certain graphs
$SimP_\tau(q,g)$	the similarity probability between q and g

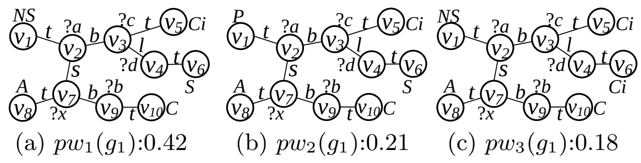


Figure 6: Possible worlds of the uncertain graph g_1

3.1 Preliminary

In order to handle the uncertainty in the semantic query graph Q^S, we resort to the classical uncertain model—possible world [3]—in uncertain graphs (Section 3.1.1). We use a popular graph similarity measure—minimum graph edit distance (GED)—to evaluate the graph similarity (Section 3.1.2).

3.1.1 Uncertain Graph Model

Definition 2. (**Uncertain Graph Model**) An uncertain graph g is defined as a five-tuple $g = \{V(g), E(g), F_L(g), \Sigma_V(g), \Sigma_E(g)\}$, where $V(g)$ is a set of vertices, $E(g)$ is a set of edges, $F_L(g)$ is a mapping function: $V(g)\times V(g) \to E(g)$, that maps from vertices to edges, $\Sigma_V(g)$ is an uncertain vertex label set which contains possible labels, $l(v)$, of vertex $v \in V(g)$ with *existence probability* $l(v).p \in (0,1]$, and $\Sigma_E(g)$ is an edge label set.

Def. 2 provides an uncertain graph model. Each vertex v in uncertain graphs is assigned with one or multiple *mutually exclusive* labels $l(v)$, each with an existence probability $l(v).p$, where $\sum_{\forall l(v)} l(v).p \leq 1$. In this work, we use the existing entity linking technique [4] to assign the probability $l(v).p$. In fact, a certain graph, g^c, is a special case of the uncertain graph g, where each vertex v has only one deterministic label $l(v)$ with existence probability $l(v).p = 1$. Note that for ease of presentation, we do not discuss the edge label uncertainty in finding similar graph pair (i.e., SimJ) problem, although it is straightforward to handle the general case. For example, we can introduce fictitious vertices to represent (uncertain) edges and assigning uncertain labels of edges to these new vertices.

In order to model the uncertain graph, we consider the well-defined *possible world* semantics [3] as follows.

Definition 3. (**Possible Worlds of an Uncertain Graph**) Let $PW(g)$ denote a set of all possible worlds in uncertain graph g. A possible world, $pw(g) \in PW(g)$, of an uncertain graph g is a deterministic graph instance, materialized from g, in which each vertex v is assigned with a certain label $l(v)$. Each possible world $pw(g)$ has an appearance probability $Pr\{pw(g)\} = \prod_{\forall v \in pw(g)} l(v).p$.

In Def. 3, each possible world is a materialized instance of uncertain graph g, which corresponds to one label assignment in vertices of g. Thus, the probability that a possible world appears in the real world is given by multiplying existence probabilities of vertex labels.

Example 2. *Due to the space limit, Figure 6 only shows an example of 3 possible worlds for uncertain graph g_1 in Figure 3. For simplicity, we use the acronym of each original label to denote this label, e.g., "NBA star"(NS), "State"(S), "type"(t). The existence probability of possible world $pw_1(g_1)$, $Pr\{pw_1(g_1)\}$, can be given by multiplying existence probabilities of vertex labels in $pw_1(g_1)$, that is, $Pr\{pw_1(g_1)\} = \prod_{i=1}^{10} l(v_i).p = 0.6 \times 1 \times 1 \times 1 \times 1 \times 0.7 \times 1 \times 1 \times 1 \times 1 = 0.42$.*

To keep our notations simple, we will use $pw(g)$ and g^c interchangeably to denote either a possible world of uncertain graph g or a certain graph, when there is no ambiguity.

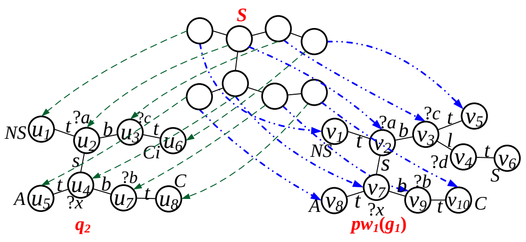


Figure 7: The CSS of q_2 and $pw_1(g_1)$

3.1.2 Graph Edit Distance

We use *minimum graph edit distance* [31], a widely used error-tolerant measure, to compute graph similarity.

Graph Edit Distance Between Certain Graphs. There are six *primitive edit operations* [29] on a certain graph g^c:

- insert/delete an isolated vertex with label;
- insert/delete an edge between two vertices; and
- substitute a (or an) vertex/edge label.

Given two certain graphs g_1^c and g_2^c, there exist many sequences of primitive edit operations to transform g_1^c to g_2^c. However, among them, the sequence with the shortest length is called the *minimum graph edit distance* (GED for short) [8] (i.e., dissimilarity) between two graphs. Computing GED between two certain graphs is NP-hard [29]. In the literature, previous works usually focus on its efficient computation [17, 31]. The most widely used approach is based on the A^* algorithm incorporating some heuristics [17].

3.1.3 Graph Edit Distance VS. Common Structural Subgraph

In this section, we presentation the relationship between graph edit distance and common structural subgraph (CSS) [1], which is used to design our bounds.

Definition 4. (Structurally Subgraph Isomorphism [1]). Given two certain graphs g_1^c and g_2^c, g_1^c is structurally subgraph isomorphic to g_2^c if and only if there exists a bijection $f : V(g_1^c) \to V(g_2^c)$, *s.t.*, 1) $\forall\ v \in V(g_1^c), f(v) \in V(g_2^c)$; 2) $\forall v_1, v_2 \in V(g_1^c)$, $\overrightarrow{v_1 v_2} \in E(g_1^c) \Leftrightarrow \overrightarrow{f(v_1)f(v_2)} \in E(g_2^c)$.

Note that, "structurally subgraph isomorphism" in Def. 4 does not consider the constraint of vertex/edge labels.

Definition 5. (Common Structural Subgraph [1], CSS) Given two certain graphs q and g^c, a graph s is a *common structural subgraph* (CSS) of q and g^c, if s is *structurally subgraph isomorphic* to both q and g^c, respectively.

For example, the graph s is a CSS of graph q_2 and the possible world $pw_1(g_1)$ in Figure 7. In this paper, we assume that the cost of each primitive edit operation is 1.

Lemma 1. *(Graph Edit Distance VS. CSS [1]) Given two certain graphs, q and g^c, s is a CSS of q and g^c. The following equation about $ged(q, g^c)$ holds.*

$$
\begin{aligned}
& ged(q, g^c) \\
&= \min_{\forall s}\{(|V(q)| + |E(q)|) - (|V(s)| + |E(s)|) && \text{(Part 1)} \\
&\quad +(|V(g^c)| + |E(g^c)|) - (|V(s)| + |E(s)|) && \text{(Part 2)} \\
&\quad +(|V_s| + |E_s|)\} && \text{(Part 3)} \\
&= |V(q)| + |E(q)| + |V(g^c)| + |E(g^c)| - F(q, g^c) && (1)
\end{aligned}
$$

where V_s and E_s are subsets of vertices and edges in CSS s, respectively, whose labels need to be substituted, and we have

$$F(q, g^c) = \max_{\forall s}\{2 \cdot (|V(s)| + |E(s)|) - (|V_s| + |E_s|)\}. \quad (2)$$

Lemma 1 provides a formula of calculating GED through CSS. The intuition is as follows. Part 1 in Eq. (1) is the cost of deleting those vertices/edges in q that do not belong to CSS s. Part 2 in Eq. (1) is the cost of inserting those vertices/edges in g^c that do not belong to CSS s. Part 3 in Eq. (1) is the cost of substituting some vertex/edge labels in CSS s. Since we may have multiple CSSs between q and g^c, the graph edit distance $ged(q, g^c)$ corresponds to a CSS that minimizes Eq. (1). According to Lemma 1, in order to compute the lower bound of $ged(q, g^c)$, we need to find the upper bound of $F(q, g^c)$ in Eq. (2).

Remarks. The work in [1] only presents the relationship between GED and CSS. As one of our technical contributions, we are the first to utilize the relationship to derive a novel lower bound so as to reduce the search space. Furthermore, we also extend the bound to handle uncertain graphs.

3.2 Finding Similar Graph Pairs: SimJ

As mentioned earlier, we obtain two graph datasets before the template generation. One is an uncertain graph dataset U derived from natural language questions, while the other one is a certain graph dataset D that collects all SPARQL query graphs in the workload. To provide promising hints for generating templates, we aim to find similar graph pairs $\langle q, g\rangle$, where $q \in D$ and $g \in U$. Since each graph $g \in U$ is uncertain, we need to consider the graph similarity measure in the uncertain graph model. Therefore, we propose a GED-based *similarity probability* as follows.

Definition 6. (Similarity Probability) Given a deterministic graph q, an uncertain graph g, and a graph edit distance threshold τ, we define the similarity probability, $SimP_\tau(q, g)$, between q and g as:

$$SimP_\tau(q, g) = \sum_{pw(g) \in PW(g)} Pr\{pw(g) | ged(q, pw(g)) \le \tau\} \quad (3)$$

Formally, finding similar graph pairs by joining a deterministic graphs D and a set of uncertain graphs U (SimJ for short) is formulated as follows:

Definition 7. (Finding Similar Graph Pairs) Given a set of deterministic graphs D (derived from the SPARQL query workload), a set of uncertain graphs U (derived from natural language question workload), a graph edit distance threshold τ, and a similarity probability threshold $\alpha \in (0, 1]$, a *SimJ* query returns the graph pairs $\langle q, g\rangle$ ($q \in D$ and $g \in U$) with $SimP_\tau(q, g) \ge \alpha$, where the similarity probability $SimP_\tau(q, g)$ is given by Eq. (3).

Intuitively, in Def. 7, the *SimJ* query retrieves those graph pairs $\langle q, g\rangle$ such that g is within τ graph edit distance from q with high confidence.

Example 3. *We consider two graphs q_2 and g_1 in Figure 3. Let the GED threshold be $\tau = 4$. We add up the appearance probabilities of the possible worlds ($pw_1(g_1)$ and $pw_2(g_3)$) whose graph edit distances to q_1 are not larger than τ, that is, $SimP_\tau(q_2, g_1) = 0.42 + 0.18 = 0.6$.*

A straightforward solution to the *SimJ* problem is: performing a nested loop join algorithm over graph sets U and D. The graph pairs whose similarity probabilities are no less than the threshold are returned as final results.

Clearly, it is an inefficient solution since uncertain graphs may have an exponential number of possible worlds. Furthermore, it involves the time-consuming computation of GED. Therefore, we need to design effective and efficient approaches, which is challenging and non-trivial.

3.3 Framework for Finding Similar Graph Pairs

We propose a filtering-and-refinement framework for processing the *SimJ* problem efficiently.

Pruning Phase. We first prune the search space to avoid the expensive computation of $SimP_\tau(q, g)$ by two techniques.

1) *Structural Pruning*: Utilizing the CSS, we design a tighter lower bound for GED between graphs q and g (Section 4). If the lower bound is larger than the GED threshold τ, the graph pair $\langle q, g\rangle$ can be filtered out.

2) *Probabilistic Pruning*: We also deduce an upper bound for the similarity probability $SimP_\tau(q, g)$ (Section 5). We can prune the candidate graph pair whose upper bound is less than the similarity probability threshold α.

Refinement Phase. For those remaining uncertain graph candidates that cannot be pruned, we compute their exact similarity probabilities, and return actual *SimJ* answers.

4. COMMON STRUCTURAL SUBGRAPH-BASED PRUNING

We present a structural pruning strategy, namely CSS (Common Structural Subgraph)-based pruning. It prunes graph pairs that are definitely not similar to each other.

Pruning Heuristics. The function $SimP_\tau(q, g)$ (as given in Eq. (3)) sums up probabilities of all possible worlds, $pw(g)$, that satisfy $ged(q, pw(g)) \le \tau$. Therefore, based on this definition, if the inequality above does not hold for any possible world $pw(g)$, then we have $SimP_\tau(q, g) = 0$, and the graph pair $\langle q, g\rangle$ can be safely pruned (since $0 \not\ge \alpha$).

Since there are an exponential number of possible worlds for uncertain graphs, it is not efficient to enumerate all possible worlds. Moreover, for each possible world $pw(g)$, the calculation of $ged(q, pw(g))$ in Eq. (3) is also very costly. Therefore, the basic idea of our solution is as follows. Given a deterministic graph q and an uncertain graph g, we propose a uniform lower bound for GED between q and each possible world of g. If the lower bound is larger than τ, it means that the GED between q and each possible world of g is larger than τ. Hence, $\langle q, g\rangle$ can be pruned safely.

Since our lower bound is based on the relationship between GED and common structural subgraph (see Section 3.1.3), the lower bound is called CSS-based lower bound. In the sequel, we first introduce CSS-based lower bound for certain graph and then extend it to the uncertain model.

4.1 Lower Bound for Certain Graph

Recall that Lemma 1 illustrates the relationship between GED and CSS. To obtain a lower bound of $ged(q, g^c)$, we can, instead, compute an upper bound, $ub\_F(q, g^c)$, of function $F(q, g^c)$ (as given in Eq. (2) of Lemma 1). Specifically, we use $\lambda_V(q, g^c)$ and $\lambda_E(q, g^c)$ to denote the common vertex/edge labels between q and g^c, i.e., $\lambda_V(q, g^c) = |\Sigma_V(q) \cap \Sigma_V(g^c)|$ and $\lambda_E(q, g^c) = |\Sigma_E(q) \cap \Sigma_E(g^c)|$, where $|\Sigma_V(\cdot)|$ and $|\Sigma_E(\cdot)|$ are the numbers of vertex or edge labels, respectively, in graphs. Due to the property of $\lambda_V(q, g^c)$ and $\lambda_E(q, g^c)$, we have the following inequalities about vertex/edge labels:

$$2 \cdot |V(s)| - |V_s| \le |V(s)| + \lambda_V(q, g^c) \quad (4)$$

$$2 \cdot |E(s)| - |E_s| \le |E(s)| + \lambda_E(q, g^c) \quad (5)$$

Intuitively, during the transformation from graph q to g^c, the number of vertex/edge labels that are not substituted is upper bounded by that of common vertex/edge labels between graphs q and g^c.

By substituting Eqs. (4) and (5) into Eq. (2), we can obtain the following inequality about function $F(\cdot,\cdot)$:

$$F(q,g^c) \leq |V(s)| + \lambda_V(q,g^c) + |E(s)| + \lambda_E(q,g^c). \quad (6)$$

Thus, given two graphs q and g^c, in order to obtain an upper bound of $F(q,g^c)$, we need to derive the upper bound of $(|V(s)|+|E(s)|)$ for any CSS s, since $\lambda_V(q,g^c)$ and $\lambda_E(q,g^c)$ are constants (when q and g^c are given).

Lemma 2. *Given two certain graphs q and g^c, the maximum value of $(|V(s)|+|E(s)|)$ in Inequality (6) is achieved if and only if $|V(s)| = \min\{|V(q)|, |V(g^c)|\}$, where s is a CSS between q and g^c.*

Proof. (proof by contradiction) According to the definition of CSS, we know $|V(s)| \leq \min\{|V(q)|, |V(g^c)|\}$. Assume that $(|V(s)|+|E(s)|)$ is largest when $|V(s)| < \min\{|V(q)|, |V(g^c)|\}$. It means that $(V(q) - V(s)) \neq \phi$ and $(V(g^c) - V(s)) \neq \phi$. Therefore, we can arbitrarily introduce two vertices in $(V(q) - V(s))$ and $(V(g^c) - V(s))$, respectively, into the current CSS s to form another CSS s'. It means that we can obtain another CSS s' between q and g^c, where $|V(s')| + |E(s')| > |V(s)| + |E(s)|$. Obviously, it is contradicted to the assumption that $|V(s)| + |E(s)|$ is maximum. Therefore, Lemma 2 holds. □

Without loss of generality, we assume that $|V(q)| \leq |V(g^c)|$. The above Lemma tells us that $(|V(s)| + |E(s)|)$ is maximum if and only if $|V(s)| = |V(q)|$, i.e, the CSS s covers all vertices in graph q. For simplicity of notations, let $|V(s)| = |V(q)| = m$ and $|V(g^c)| = n$ and $m \leq n$. In order to estimate the upper bound of $(|V(s)|+|E(s)|)$, we should estimate the upper bound of $|E(s)|$ where $|V(s)| = |V(q)| = m$. We propose the following method. Readers can skip to Lemma 4 and Theorem 1 directly if they are not interested in the detailed derivation process.

Since s is a CSS between q and g^c, there must exist a subgraph q' in q, where q' is structurally graph isomorphic to s. Analogously, we can also find a subgraph $g^{c\prime}$ in g^c, where $g^{c\prime}$ is also structurally graph isomorphic to s. Since $|V(s)| = m$, thus both q' and $g^{c\prime}$ have m vertices, respectively. Assume that the m vertices in q' are u_1, ..., u_m and the m vertices in $g^{c\prime}$ are v_1, ..., v_m. It is easy to know q' is structurally graph isomorphic to $g^{c\prime}$. We assume that the graph isomorphism is defined under an injective function f from $\{u_1,...,u_m\}$ to $\{v_1,...,v_m\}$. A naive upper bound for $|E(s)|$ is $|E(s)| \leq |E(q)|$. Obviously, this naive bound is not tight. Thus, we need to tighten the bound.

We need to delete some edges in q to form CSS s. To estimate the upper bound for $E(s)$, we need to delete as few edges as possible from q to form s. We utilize the degree differences between q and s to count how many edges that should be deleted. Let $d(u_1,q)$ denote the u_1's degree in graph q. $d(u_1,s)$ is the u_1's degree in CSS s. Deleting one edge in q affects two vertex degrees. Therefore, we need to delete $[\frac{d(u_1,q)-d(u_1,s)}{2}]$ edges to degrade $d(u_1,q)$ to $d(u_1,s)$. In a word, we have the following equation.

$$DelEdge = |E(q)| - |E(s)| = \sum_{i=1}^{i=m} \frac{d(u_i,q) - d(u_i,s)}{2}$$

where $DelEdge$ denotes the number of edges that should be deleted from q.

Now, in order to estimate the lower bound for $DelEdge$, we define an operator $\ominus$ as follows.

Definition 8. Given two integers a and b, a binary operator, denoted by $\ominus$, is defined as

$$a \ominus b = \begin{cases} a-b, & \text{if} \quad a > b, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Let $f(u_i)$ denote the mapping vertex in graph g^c under the injective function f. It is straightforward to know $d(u_i,s) \leq d(f(u_i),g^c)$. Therefore, we have the following inequality.

$$\begin{aligned} DelEdge &= |E(q)| - |E(s)| = \textstyle\sum_{i=1}^{i=m} \frac{d(u_i,q)-d(u_i,s)}{2} \\ &\geq \textstyle\sum_{i=1}^{i=m} \frac{d(u_i,q)\ominus d(f(u_i),g^c)}{2} \end{aligned} \quad (8)$$

According to Eq. (8), we need to estimate the lower bound of $\sum_{i=1}^{i=m} \frac{d(u_i,q)\ominus d(f(u_i),g^c)}{2}$. Fortunately, Lemma 3 holds.

Without loss of generality, assuming $m \leq n$, we define the *degree distance*, $dif(q,g^c)$, of q and g^c as follows.

Definition 9. (Degree Distance) Given two certain graphs q and g^c with sorted degree sets in non-increasing order, i.e., $d(u_1,q) \geq ... \geq d(u_m,q)$ and $d(v_1,g^c) \geq ... \geq d(v_n,g^c)$. The degree distance between q and g^c, denoted by $dif(q,g^c)$, is:

$$dif(q,g^c) = \sum_{i=1}^{m} (d(u_i,q) \ominus d(v_i,g^c)) \quad (9)$$

Lemma 3. *Let us consider two certain graphs q and g^c with m and n vertices, $m \leq n$. For any injective function f from m vertices $\{u_1,...,u_m\}$ in q to n vertices $\{v_1,...,v_n\}$ in g^c, the following inequality holds.*

$$\sum_{i=1}^{i=m} \frac{d(u_i,q)\ominus d(f(u_i),g^c)}{2} \geq \frac{dif(q,g^c)}{2} \quad (10)$$

Proof. The proof can be achieved by induction. For details, please refer to Appendix A. □

Lemma 4. *Let us consider two certain graphs q and g^c with m and n vertices, $m \leq n$. s is a CSS between q and g^c and $|V(s)| = |V(q)|$. The upper bound of $|E(s)|$ is as follows.*

$$|E(s)| \leq |E(q)| - \frac{dif(q,g^c)}{2} \quad (11)$$

where $dif(q,g^c)$ is defined in Def. 9.

Proof. According to Eqs. (8), (9) and (10), we have:

$$DelEdge = |E(q)| - |E(s)| \geq \frac{dif(q,g^c)}{2}$$

Finally, we derive the upper bound for $E(s)$. □

Theorem 1. (CSS-Based Lower Bound of Graph Edit Distance for Certain Graphs) *Let us consider two certain graphs q and g^c. Without loss of generality, assume that $|V(q)| \leq |V(g^c)|$. The lower bound for graph edit distance between q and g^c is:*

$$\begin{aligned} &ged(q,g^c) \geq lb\_ged_{CSS}(q,g^c) \\ &= |V(g^c)|+|E(g^c)|-\lambda_E(q,g^c)+\frac{dif(q,g^c)}{2}-\lambda_V(q,g^c) \end{aligned} \quad (12)$$

where $dif(q,g^c)$ is defined in Eq. (9).

Proof. According to Lemma 2 and Eqs. (6) and (11),

$$\begin{aligned} &F(q,g^c) \leq |V(s)| + \lambda_V(q,g^c) + |E(s)| + \lambda_E(q,g^c) \\ &\leq |V(q)| + \lambda_V(q,g^c) + |E(q)| - \frac{dif(q,g^c)}{2} + \lambda_E(q,g^c), \end{aligned}$$

Thus, based on Eq. (1) we obtain the following inequality $ged(q,g^c) \geq |V(g^c)| + |E(g^c)| - \lambda_E(q,g^c) + \frac{dif(q,g^c)}{2} - \lambda_V(q,g^c)$. □

Pruning Effect Analysis of CSS-Based Lower Bound. Our CSS-based bound falls into the first category, namely, "global filter". Please refer to our discussion about global filters in the related work section (Section 8). More importantly, our CSS-based lower bound is proven to be tighter than the two existing ones in [29] and [31]. Note that, we only need to compare our lower bound with the label-multiset-based lower bound in [31], denoted as $lb\_ged_{LM}(q, g^c)$, since [31] has proven that $lb\_ged_{LM}(q, g^c)$ is tighter than the vertex/edge number-based lower bound in [29].

THEOREM 2. *Given two certain graphs q and g^c, it holds that*

$$lb\_ged_{CSS}(q, g^c) \geq lb\_ged_{LM}(q, g^c),$$

where $lb\_ged_{LM}(q, g^c) = \max\{|V(q)|, |V(g^c)|\} - \lambda_V(q, g^c) + \max\{|E(q)|, |E(g^c)|\} - \lambda_E(q, g^c)$.

PROOF. For details, please refer to Appendix B. □

4.2 Lower Bound for Uncertain Graphs

In this section, we discuss how to compute a lower bound for GED between graph q and uncertain graph g. Let us recall $lb\_ged_{CSS}(q, g^c)$ in Theorem 1. If we utilize Eq. (12) to compute the bound for uncertain graph g, all elements in Eq. (12) are constants except for $\lambda_V(q, g^c)$. Therefore, the only remaining issue is to compute an upper bound of $\lambda_V(q, g^c)$ for all possible worlds g^c of uncertain graph g.

Definition 10. (Vertex Label Bipartite Graph) Given a certain graph q and an uncertain graph g, we can construct a vertex label bipartite graph $G_b = (V(g), V(q))$. There is an edge between $v_i(\in V(g))$ and $u_j(\in V(q))$ iff $l(u_j) \in l(v_i)$.

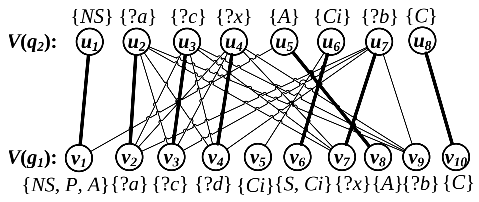

{NS} {?a} {?c} {?x} {A} {Ci} {?b} {C}
V(q2):
u1 u2 u3 u4 u5 u6 u7 u8
V(g1):
v1 v2 v3 v4 v5 v6 v7 v8 v9 v10
{NS, P, A} {?a} {?c} {?d} {Ci} {S, Ci} {?x} {A} {?b} {C}

Figure 8: Vertex label bipartite graph of g_1 and q_2

Figure 8 shows an example of a vertex label bipartite graph of q_2 and g_1. Since $\lambda_V(q, g) \leq \max_{g^c \in PW(g)} \{\lambda_V(q, g^c)\}$, we need to estimate the maximum value for $\lambda_V(q, g^c)$ in all possible worlds g^c of uncertain graph g. We can reduce the problem of estimating the maximum value for $\lambda_V(q, g^c)$ to the maximum matching problem. A maximum matching is exactly to identify the largest set of non-adjacent matching edges, which can be solved by a classic algorithm, e.g., *Hungarian algorithm* [10].

The upper bound for $\lambda_V(q, g)$ equals to the number of matching edges in the maximum matching. According to the above analysis, we have the following theorem.

THEOREM 3. *(CSS-Based Lower Bound of Graph Edit Distance for Uncertain Graphs) Let us consider a certain graph q and an uncertain graph g. The CSS-based lower bound, $lb\_ged_{CSS}(q, g)$, of graph edit distance $ged(q, g)$ is given by:*

$$\begin{aligned} &ged(q, g) \geq lb\_ged_{CSS}(q, g) \\ &= |V| + |E| - \lambda_E(q, g) + \frac{dif(q, g)}{2} - \lambda_V(q, g), \end{aligned} \tag{13}$$

where $|V| = \max\{|V(g)|, |V(q)|\}$, $|E|$ is the number of edges in graph q or g with more vertices, $\lambda_E(q, g) = |\sum_E(q) \cap \sum_E(g)|$, $dif(q, g)$ is defined in Eq. (9) and $\lambda_V(q, g)$ is the size of the maximal matching in vertex label bipartite graph.

5. PROBABILISTIC PRUNING

The basic idea of probabilistic pruning is: utilizing the similarity probability threshold to filter out false alarms. Specifically, the intuition is to prune those graph pairs $\langle q, g \rangle$ with low similarity probabilities (i.e., $Sim_\tau(q, g) < \alpha$).

By replacing graph edit distance $ged(q, pw(g))$ (given in Eq. (3)) with the CSS-based lower bound $lb\_ged_{CSS}(q, pw(g))$ (given in Theorem 3), we obtain an upper bound of $SimP_\tau(q, g)$ under possible world semantics. That is, we have:

$$SimP_\tau(q, g) \leq \sum_{pw(g) \in PW(g)} Pr\{pw(g) | lb\_ged_{CSS}(q, pw(g)) \leq \tau\}. \tag{14}$$

Let $C(q, g)$ be $|V| + |E| - \lambda_E(q, g) + \frac{dif(q, g)}{2}$, which is a constant (given q and g). By substituting $C(q, g)$ into Eq. (14), we can rewrite Eq. (14) as:

$$\begin{aligned} SimP_\tau(q, g) &\leq \sum_{pw(g) \in PW(g)} Pr\{pw(g) | \lambda_V(q, pw(g)) \geq C(q, g) - \tau\} \\ &= Pr\{\lambda_V(q, g) \geq C(q, g) - \tau\}. \end{aligned} \tag{15}$$

The Rewriting of Probability Upper Bound. Next, the only remaining issue is how to compute the upper bound of $SimP_\tau(q, g)$ in Inequality (15). Intuitively, $\lambda_V(q, g)$ in Inequality (15) is the number of common vertex label pairs between graphs q and g. In other words, we can define a mapping from $V(g)$ to $V(q)$, and count the number of matching vertex labels.

Without loss of generality, we assume that graph g has m vertices, which correspond to m variables $x_1, x_2, \ldots$, and x_m, respectively. Each variable x_i has the value domain $\{0, 1\}$. If a vertex $v_i \in V(g)$ has a matching with $u_j \in V(q)$ such that $l(v_i) = l(u_j)$, then variable x_i (w.r.t. vertex v_i) is set to 1, otherwise x_i is set to 0. As a result, we can prove that the RHS of Inequality (15) is equivalent to: $Pr\{\sum_{i=1}^m x_i \geq C(q, g) - \tau\}$. Specifically, we have the following lemma.

Lemma 5. *Given a deterministic graph q and an uncertain graph g, it holds that: $SimP_\tau(q, g) \leq Pr\{\sum_{i=1}^m x_i \geq C(q, g) - \tau\}$.*

Note that, in a matching each vertex in q can be only matched with one vertex in g at most. It indicates that variables x_i take different values in different possible worlds $pw(g)$, and sometimes they might be correlated with each other. Thus, we will further relax this probability upper bound in Lemma 5, by using independent variables directly.

To reduce the computation cost w.r.t. correlated variable x_i, we introduce new variable y_i for each vertex $v_i \in V(g)$ to further relax the probability upper bound in Lemma 5. Specifically, similar to variables x_i, we also assume that $y_i = 1$, if $l(v_i) = l(u_j)$. The difference is that each vertex u_j can be used more than one time (i.e., matching with more than one vertices in $V(g)$). In this case, we know that it always holds that $y_i \geq x_i$. Thus, we obtain the following lemma.

Lemma 6. *Given a graph q and an uncertain graph g, it holds that: $Pr\{\sum_{i=1}^m x_i \geq C(q, g) - \tau\} \leq Pr\{\sum_{i=1}^m y_i \geq C(q, g) - \tau\}$.*

Furthermore, our newly defined m variables $y_1, y_2, \ldots$, and y_m are independent. Therefore, we denote $(y_1 + y_2 + \ldots + y_m)$ as a single random variable Y. By applying the Markov's inequality [5], we derive an upper bound as follows:

THEOREM 4. *(Probabilistic Upper Bound for Similarity Probability) Let us consider a deterministic graph q and an uncertain graph g. The upper bound for $SimP_\tau(q,g)$ is given as follows.*

$$SimP_\tau(q,g) \le ub\_SimP_\tau(q,g) = \frac{E(Y)}{C(q,g)-\tau} \qquad (16)$$

where $E(Y) = \sum_{i=1}^{m} E(y_i)$ and $E(y_i) = \Sigma_{l_j \in l(v_i)} Pr(l_j | l_j \cap \Sigma_V(q) \neq \emptyset)$ and $C(q,g) = |V| + |E| - \lambda_E(q,g) + \frac{dif(q,g)}{2}$, and $|V|$, $|E|$, $\lambda_E(q,g)$ and $dif(q,g)$ are defined in Theorem 3.

PROOF. According to Lemmas 5 and 6, we have $SimP_\tau(q,g) \le Pr\{\sum_{i=1}^{m} y_i \ge C(q,g)-\tau\}$. Let us denote $(y_1+y_2+\ldots+y_m)$ as a single random variable Y. Thus, $SimP_\tau(q,g) \le Pr\{Y \ge C(q,g)-\tau\}$. Based on the Markov's inequality [5], we obtain $SimP_\tau(q,g) \le \frac{E(Y)}{C(q,g)-\tau}$. $\square$

Example 4. *Assume that $\tau = 4$ and $\alpha = 0.6$. Considering the deterministic graph q_1 and uncertain graph g_1 in Figure 3, $E(Y) = \sum_{i=1}^{10} E(y_i) = 5$. According to Eq. (16), we have $ub\_SimP_\tau(q_1,g_1) = 5/13 \approx 0.38 < 0.6$. Thus, we prune the graph pair $\langle q_1, g_1\rangle$ safely.*

We also consider correlations among variables x_i directly and derive tight upper bounds by the law of total probability [5]. Due to space limit, we have to put it in our future work.

6. SIMJ QUERY PROCESSING

We first present how to conduct *SimJ* processing. Then, we propose a cost-based query optimization technique, which divides possible worlds of uncertain graphs into fine-grained groups and thus attains higher pruning power.

6.1 SimJ Procedure

Algorithm 1 (please refer to Appendix D) presents the procedure of our *SimJ* processing approach. For each pair of graphs, we will first compute the lower bound $lb\_ged_{CSS}(q,g)$ of graph edit distance via Theorem 3 (line 3). If $lb\_ged_{CSS}(q,g)$ is larger than τ, the graph pair $\langle q,g\rangle$ can be pruned safely; otherwise, we compute the upper bound of the similarity probability by Theorem 4 (line 5). If the probability upper bound is smaller than α, then we can safely filter out the graph pair $\langle q,g\rangle$; otherwise, we need to conduct the verification (lines 6-15). Regarding the computation of GED between two certain graphs for the refinement (as given in Eq. (3)), any existing method, such as [17], can be adopted.

We give the time complexity of Algorithm 1. For more details please refer to Appendix D.

6.2 Cost-Based Query Optimization

In order to improve the pruning power, we propose to divide all possible worlds of an uncertain graph g into disjoint possible world groups (denoted as PWG_i). Each PWG_i contains fewer possible worlds. Thus, we can obtain tighter lower/upper bounds of the graph edit distance and similarity probability. Algorithm 2 (in Appendix E) gives the details.

Specifically, the uncertain graph g is divided into k possible world groups $PWG_1, \ldots, PWG_k$. Each lower bound $lb\_ged_{CSS}(q, PWG_i)$ is computed according to Theorem 3. The groups satisfying the inequality $lb\_ged_{CSS}(q, PWG_i) > \tau$ can be pruned. Then, we can sum up upper bounds, $ub\_SimP_\tau(q, PWG_i)$, of similarity probabilities for the remaining groups, which can be used for probabilistic pruning.

Partitioning Strategy. Given an uncertain graph g, we can partition g into different possible world groups, which may lead to different bounds for the pruning. Hence, a challenging problem arises, that is, how to partition possible worlds of g so that we can gain a tighter lower bound.

Process of Partitioning Possible Worlds. Our partitioning strategy starts with one group, and then recursively selects one group to split into two groups, until the number of groups reaches a predefined threshold. Here, we will select a group with the smallest lower bounds of graph edit distance and/or the one with the largest upper bound of similarity probability. Intuitively, such a group incurs low pruning power, and thus needs to be further partitioned to improve the pruning power.

Dividing a Possible World Group into Two Groups. Next, we discuss how to partition a possible world group, PWG_i, into two groups by splitting uncertain labels of a selected vertex, which is based on the two observations below. First, according to Theorem 3, it is intuitive that the less uncertainties g contains, the tighter the lower bound $lb\_ged_{CSS}(q,g)$ is. Second, as mentioned in Section 5, if the number of possible labels that a vertex contains is small and $P(x_i = 1 | L(v_i) = l)$ is small, the probability $P(x_i = 1)$ is small.

From observations above, we have the following two principles to select a vertex (in PWG_i) to split uncertain labels, and divide PWG_i into two groups.

- Select a vertex with high (total) existence probabilities of uncertain labels in PWG_i to split;
- Select a vertex with more possible labels to split.

We only need to sum up $ub\_SimP_\tau(q, PWG_i)$ for groups whose GED lower bounds are no larger than τ, and minimize the summation to obtain tighter similarity probability upper bounds.

Cost Model for Evaluating the Partitioning. We present a cost model to evaluate our grouping results:
$\arg\min\{ub\_SimP_\tau(q, PWG_i) | lb\_ged_{CSS}(q, PWG_i) \le \tau\}$.

With the heuristics/principles above, we can obtain different partitioning strategies, which are evaluated by our cost model. We select one partitioning strategy that minimizes the proposed metric (provided by our cost model).

7. EXPERIMENTAL RESULTS

7.1 Experiment Setting

7.1.1 Datasets

Real Dataset 1. QALD-3[5]: A benchmark delivered in the third evaluation campaigns on question answering over linked data. It contains 200 natural language questions, each of which has the corresponding SPARQL query.

Real Dataset 2. WebQuestions+DBpedia SPARQL: WebQuestions[6] is an open dataset created by Berant et al. It contains 5,810 natural language questions, denoted by *WebQ*. DBpedia SPARQL[7] is a set of SPARQL query logs which are actually posed queries against the DBpedia knowledge base. We randomly select 73,000 SPARQL queries from the whole dataset and create the deterministic graphs.

For natural language questions, we generate its corresponding uncertain graphs, denoted by U, utilizing the method

[5]http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=home&q=3
[6]http://www-nlp.stanford.edu/software/sempre/
[7]http://aksw.org/Projects/DBPSB.html

Table 2: Statistics of Data sets

Dataset	$\lvert U\rvert$	avg.$\lvert V\rvert$	avg.$\lvert E\rvert$	avg.$\lvert L_V\rvert$	$\lvert D\rvert$
QALD3	200	5.73	4.51	4.50	200
WebQ	5,810	6.15	5.14	4.39	73,057
ER	100,000	64.86	157.07	9.39	100,000
SF	100,000	63.35	88.61	13.52	100,000
MM	23,250	5.35	4.92	4.21	2,500

introduced in Section 2.1. Regarding the SPARQL queries, we build their deterministic graphs, denoted by D.

Real Dataset 3. MM Workloads[8]**:** It contains both NLQ workloads and SPARQL query workloads, which are based on a closed domain, i.e., music and movies.

Furthermore, in order to evaluate efficiency of our SimJ algorithm, we construct two synthetic datasets as follows.

Synthetic Datasets. ER and **SF:** For the ER model, we randomly connect M pairs of vertices as edges in the graph; for the SF model, we use a graph generator *gengraph win*[9] to generate graphs whose vertex degrees, d, satisfy the power law distribution. We generate two sets of deterministic graphs for ER and SF, respectively. Table 2 lists statistics of all datasets used in the experiments.

7.1.2 Evaluation Measures

Metrics. We adopt several metrics for the evaluation: (1) Correct Answers ($|C|$), the number of the returned correct graph pairs[10]; (2) *Precision* (p), as $p = \frac{|C|}{|R|}$, where R is the set of returned answers for all questions; (3) *Response Time*, the time cost of filtering out false alarms and refining; (4) *Candidate Ratio*, the number of candidate graph pairs (after pruning) divided by $|U| \times |D|$. (5) Matching Proportion ϕ, which is defined as follows:

$$\phi = \frac{|\text{words covered by template P}|}{|\text{words in the whole question NLQ}|}$$

Correct answers: Consider a returned pair $\langle q, n\rangle$, where q is a SPARQL query and n is a natural language question. For natural language question n, we manually issue the corresponding SPARQL query q'. If q matches q' except for entity phrases, we regard that the returned pair is correct; otherwise, it is wrong. 10 students helped evaluate the results returned by our algorithm.

Setup. All experiments are conducted on a Windows 7 PC with a 2.9GHz Pentium IV CPU and 4GB main memory. All programs were implemented in C++.

7.2 Effectiveness Evaluation

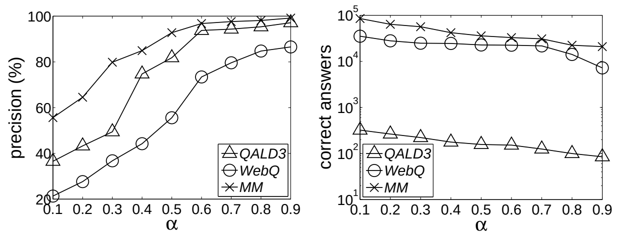
Effect of GED threshold τ. To guarantee the quality of answers, the GED theshold τ is varied from 0 to 2. The similarity probability threshold α is fixed to be 0.9. When $\tau = 0$, the precisions over QALD-3 query log and WebQ are both 100%. It means that all the graph pairs delivered are correct, i.e., the corresponding natural language question and the SPARQL query has the same query intention. However, the number of answers is limited, i.e., 3 and 55, respectively. When $\tau = 1$, the correct answers are more enough with only a small sacrifice of the precisions.

In summary, as shown in Table 3, the precision becomes low with the growth of τ. But the number of correct answers grows with the increasing of τ. That is because smaller τ restricts the difference between two graphs. If τ becomes larger, some noises may be introduced, that is, some unrevelent graph pairs will be returned.

[8]a dataset acquired from a search engine company

[9]http://fabien.viger.free.fr/liafa/generation/

[10]Since it is hard to compute the recall, we utilize $|C|$ to measure the coverage of our method.



(a) Precision *vs.* α (b) Correct answers *vs.* α

Figure 9: Effect of similarity probability threshold α

Table 3: Effect of GED threshold τ

τ	QALD-3 Query Log			WebQ		
	$\lvert R\rvert$	precision	time(s)	$\lvert R\rvert$	precision	time(s)
0	3	100%	1.453	55	100%	76.851
1	86	97.67%	1.862	8,351	86.54%	100.316
2	2,421	52.33%	2.113	179,227	37.69%	652.947

Effect of Similarity Probability Threshold α. We fix the GED threshold τ to be 1, and vary the similarity probability threshold α from 0.1 to 0.9. Figure 9 shows the effect of α on $|C|$ and precision over real datasets. The precision grows with the increasing of α. It is reasonable that high probability threshold filters out the low-quality graph pairs. For example, when α is 0.7, the precision achieves 80%. Meanwhile, the number of correct answers decreases when α is larger as depicted in Figure 9(b).

The precision on MM dataset is better than that on QALD-3 and WeQ datasets. It is because that MM dataset is based on a knowledge graph of specific type (music and movies), which indicates that both natural language questions and SPARQL queries focus on similar topics.

Case Study. We provide a case study over the experiment "QALD-3 + DBpedia" (using QALD-3 questions and DBpedia queries) in Figure 10. *SimJ* identifies meaningful results of high quality. As an example, for the question "Give me all movies directed by Francis Ford Coppola", we find the SPARQL "SELECT ?film where $\{\langle ?film\ type\ Film\rangle\ \langle ?film\ director\ Joel\_Schumache\}$". Note that, the returned graph pairs do not necessarily ask the same question. The key is that we can build templates based on these graph pairs. Using the method discussed in step 3 of Section 2.1, we can obtain the templates. The corresponding templates are shown in Figure 16 (please refer to Appendix F). We also provide the analysis of the results in Appendix F.1.

To evaluate the quality of the generated templates, we use the questions of QALD to test the effectiveness. We translate the questions into SPARQL queries utilizing these generated templates, and then search these SPARQL queries over DBpedia. The precision and recall of our method are both 0.65 and outperform the competitors significantly. We include the details in Appendix F.2.

7.3 Efficiency Evaluation

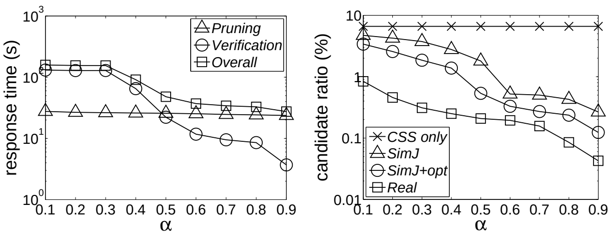
In order to evaluate the efficiency of our method, we test three *SimJ* answering techniques: the "CSS only" approach applies the CSS-based pruning method (without using the probabilistic pruning); the "*SimJ*" approach uses CSS-based and probabilistic pruning methods; the "*SimJ*+opt" approach employs the cost-based optimization by dividing possible worlds into groups to enhance the pruning power.

What is the ruling party in Lisbon?	SELECT ?uri WHERE
	{ Germany type Country.
	Germany leaderParty ?uri. }

Give me all movies directed by Francis Ford Coppola	SELECT ?film WHERE
	{ ?film type Film.
	?film director Joel_Schumache. }

Which software has been developed by organizations founded in California?	SELECT ?product WHERE { ?company type Organisation. ?company foundationPlace Bangalore. ?product developer ?company. ?product type Software. }

Figure 10: Case study over DBpedia+QALD-3



(a) Response time *vs.* α (b) Candidate ratio *vs.* α

Figure 11: Effect of threshold α (WebQ)

Effect of Similarity Probability Threshold α. We study the effect of α over the performance by varying similarity probability thresholds α from 0.1 to 0.9. As presented in Figure 11(a), the threshold has little impact on the pruning time. When α is larger, the candidates will become less. Therefore the overall time consumed decreases. As depicted in Figure 11(b), employing the optimization technique, the pruning power of the "*SimJ+opt*" is higher than "*SimJ*". Note that, the similarity threshold has no impact on the pruning ability of "CSS only".

Effect of Graph Edit Distance Threshold τ. Next, we vary GED thresholds τ from 0 to 5. Figure 12(a) shows the response time including pruning time and verification time. The overall response time grows with the increasing of τ, since large τ will result in more returned answers, as demonstrated in Figure 12(b). Notice that "*SimJ*" adopts the probability pruning in addition to the CSS-based pruning. Therefore, "*SimJ*" incurs fewer candidates, which can greatly reduce the refinement cost. Furthermore, the "*SimJ*+opt" approach considers the pruning with multiple possible world groups (rather than 1 group in "*SimJ*"), so that we can obtain tighter probability upper bound and thus enhance the pruning power. Thus, "*SimJ*+opt" has fewer candidates than "*SimJ*". When τ becomes larger, the improvement degrades slightly since larger τ decreases the upper bound of similarity probability as shown in Eq. 16.

7.4 Scalability Evaluation

Effect of Group Number GN. Here, we study the effect of group number, GN, which is varied from 1 to 40 stepped by 5. As shown in Figure 13(a), the time consumed by "*SimJ+opt*" grows with the increasing of GN. Clearly, the more groups we generate, the more filtering time we may need in the join processing. Meanwhile, the uncertain information in each group will be less, which is beneficial to filtering out more false alarms as confirmed in Figure 13(b). As expected, the group number has no impact on "*SimJ*" and "CSS only" approaches.

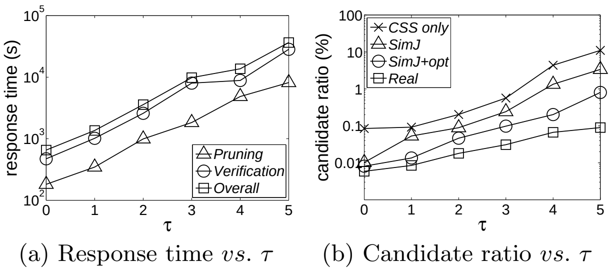
Effect of the Number of Possible Labels $|L(v)|$. For a specific data set, we average the number of possible labels, and vary this number from 2 to 6. Figs. 14(a) and 14(b) report the response time and pruning power, respectively. The response time increases as $|L(v)|$ increases, because "*SimJ*" invokes computing the upper bound of the maximum matching in a bipartite graph. Moreover, the more uncertain labels g contains, the larger the bipartite graph may be. Thus, more time will be consumed. The pruning ability decreases when $|L(v)|$ becomes larger as shown in Figure 14(b). Note that, the pruning powers of "*SimJ*" and "*SimJ+opt*" become higher when $|L(v)|$ is larger than 5. The reason is that although there are more uncertain labels incident to each vertex, the existence probability of each uncertain label is smaller, which favors the computation of the similarity probability upper bound.

Comparisons With Existing Works. To confirm the performance of our CSS-based method, we compare it with existing algorithms that are devised for deterministic graphs. As discussed in Theorem 2, our CSS-based lower bound has been proven to be tighter than the two existing global filters. Regarding these outstanding algorithms proposed recently, i.e., path [31], SEGOS [22], and Pars [30], in order to handle uncertain graphs, they must enumerate all possible worlds or ignore vertex labels. It is impractical for the first case due to the exponential number of possible worlds. Thus, we implement these methods by only considering the graph structure to handle uncertain graphs. Figs. 15(a) and 15(b) show the filtering time and candidate ratio *vs.* GED threshold τ, respectively, for the comparison.

Clearly, the computation of the CSS approach is the most efficient, and it outperforms its competitors greatly in terms of filtering time. More importantly, as shown in Figure 15(b) our proposed GED lower bound, $lb\_ged_{CSS}(q, g)$, has the highest pruning power compared to other competitors.

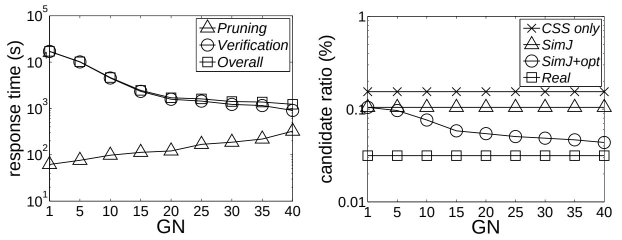
8. RELATED WORK

8.1 Question/Answering (Q/A) over RDF



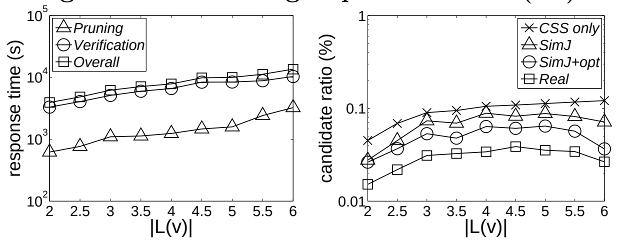
(a) Response time *vs.* τ (b) Candidate ratio *vs.* τ

Figure 12: Effect of GED Threshold τ (ER)



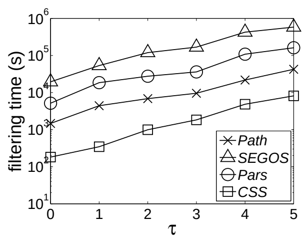
(a) Response time *vs.* GN (b) Candidate ratio *vs.* GN

Figure 13: Effect of group number GN (SF)

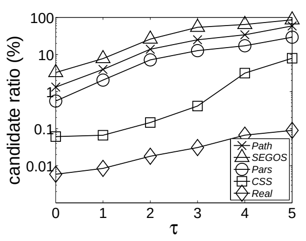


(a) Response time *vs.* $|L(v)|$ (b)Candidate ratio *vs.* $|L(v)|$

Figure 14: Effect of $|L(v)|$ (ER)



(a) Filtering time *vs.* τ



(b) Candidate ratio *vs.* τ

Figure 15: Comparison with Existing Works (AIDS)

RDF Q/A systems have attracted extensive attentions in both NLP (natural language processing) [19, 23, 20] and DB (database) communities [24, 33].

To solve the problem of RDF Q/A, templates are widely used [19, 20]. In the well-known True Knowledge Q/A system, a natural language question is turned into a structural query using some manually defined templates [19]. The templates used in this system are manually defined in the offline phase. It is expensive to pre-define all the templates especially for the open-domain RDF knowledge graphs. The authors in [20] propose a template-based method to answer the aggregate-like questions. They parese the questions to SPARQL templates, and instantiate the templates by entity/predicate mapping [20]. To the best of our knowledge, none of existing work study how to generate templates automatically for RDF Q/A. We are the first to propose a join-based approach for building templates automatically.

There are also some work that do not utilize templates, such as DEANNA [23] and gAnswer [33]. DEANNA [23] builds a disambiguation graph and reduces disambiguation as an integer linear programming problem. gAnswer translates a natural language question into a semantic query graph Q^S and tries to find the top-k matches over RDF graph with the highest confidence probabilities. Fader et al. [7] introduce a machine learning approach to answer factual questions. However, they only focus on single-relation queries. Yao et al. [25] associates question features with answer patterns described by Freebase to answer natural language questions.

Note that, although we adopt the method in gAnswer to generate the semantic query graph Q^S for each natural language question, we address the different issue. In this paper, we study how to build high-quality templates for RDF Q/A. These templates can be used in the template-based RDF Q/A systems.

8.2 Graph Edit Distance

Recently, edit distance-based graph similarity search has been well studied in certain graphs. Since it is an NP-hard problem to compute GED, lots of lower bounds have been proposed to speed up computation. They are categorized into *global filters*, *n-gram filters*, and *partition-based filters.*

1. global filter. There are two existing global filters. The first one is to compute the differences of vertex number and edge number as the lower bound [29]. The second one is the difference of vertex labels and edge labels [31].

2. n-gram filter. The basic idea is to represent a graph as a set of small-size substructures (called n-grams), such as *c-star* in [29], *k-Adjacent Tree* in [21] and *paths* in [31]. The intuition lies in that the two graphs must share lots of common n-grams if they are similar to each other (i.e., their graph edit distances are less than a threshold).

Although experiment results in existing work [29, 21, 31] show that n-gram filters are better than the two global filters in some real graph datasets, their superiorities over global filters cannot be proven theoretically. Furthermore, it is easy to find some cases that the global filters can beat these n-gram methods. However, we can prove that our filter can beat the two existing global filters from a theoretical perspective (See Theorem 2).

3. Partition-based filter. Pars [30] proposes a partition-based method. It decomposes data graphs into non-overlapping partitions, and computes the lower bound according to the number of mismatchings of these partitions.

Note that, all of these filters are proposed for certain graphs. It is not easy to extend them to uncertain scenarios. One of the technical contributions in this work is to propose a tight lower bound for GED in uncertain graphs.

The most widely used method computes exact graph edit distance based on A^* algorithm incorporating heuristics [17]. It explores the space of all possible vertex mappings between two graphs to find the optimal mapping. Zhao et al. [31] improve A^* with two heuristics. Specifically, exploiting minimum edit filtering and local label filtering to give a heuristic estimate of the distance from the current state to the goal. Note that our work focuses on the filtering phase, which are independent of the verification algorithms.

8.3 Uncertain Graph Management

Managing and mining uncertain graphs have attracted extensive attentions due to noises and inaccuracies. For example, Zou et al. study the problem of frequent subgraph mining on uncertain graphs under probabilistic semantics [34]. Kollios et al. use expected edit distance , which is extended from edit distance based graph clustering, to probabilistic graphs [14]. There are two existing work [27, 28] investigate subgraph search over uncertain graphs. However, their uncertain graph models do not measure the uncertainty of labels as does in our uncertain graph model (Def. 2). Beyond that, the work [27] proposes the similarity model based on maximum common subgraph, not the edit distance that is studied in this paper. There are some previous works [11, 12] that study probabilistic XML data. In contrast, we focus on general uncertain graph data in this paper (instead of tree structure).

9. CONCLUSIONS

To build templates for RDF Q/A, we propose a novel approach, i.e., performing graph similarity join over large uncertain graphs under the constraint of graph edit distance. Following the filtering-and-verification paradigm, we propose Common Structural Subgraph (CSS)-based pruning and probabilistic pruning to filter out as many false alarms as possible. In order to improve the pruning ability, we divide each uncertain graph to a number of groups based on a cost model. Extensive experiments over real and synthetic datasets have confirmed the effectiveness and efficiency of our solution.

Acknowledgments. This work was supported by 863 project (2015AA015402), NSFC funding (61370055,61202008,61272344) and the RGC of Hong Kong SAR (No. CUHK 418512). Authors are also grateful to anonymous reviewers for their constructive comments. This work was partially done (i.e., the uncertain technique part) when the first author was as a visiting student in UTPA. Lei Zou is the corresponding author of this work.

APPENDIX

A. REFERENCES

[1] L. Brun, B. Gaüzère, and S. Fourey. Relationships between graph edit distance and maximal common structural subgraph. In *SSPR/SPR*, pages 42–50, 2012.

[2] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *WWW*, pages 74–83, 2004.

[3] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4), 2007.

[4] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *COLING*, pages 277–285, 2010.

[5] R. Durrett. *Probability: Theory and Examples.* Cambridge University Press, 2010.

[6] J. Eisner. Learning non-isomorphic tree mappings for machine translation. In *ACL*, pages 205–208, 2003.

[7] A. Fader, L. S. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answering. In *ACL*, pages 1608–1618, 2013.

[8] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1):113–129, 2010.

[9] M. Heilman and N. A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*, pages 1011–1019, 2010.

[10] H.W.Kuhn. The hungarian method for the assignment problem. In *Naval Research Logistics*, pages 83–97, 1955.

[11] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query efficiency in probabilistic xml models. In *SIGMOD*, pages 701–714, 2008.

[12] B. Kimelfeld and P. Senellart. Probabilistic xml: Models and complexity. In *Advances in Probabilistic Databases for Uncertain Information Management*, pages 39–66. 2013.

[13] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *ACL*, pages 423–430, 2003.

[14] G. Kollios, M. Potamias, and E. Terzi. Clustering large probabilistic graphs. *IEEE Trans. Knowl. Data Eng.*, 25(2):325–336, 2013.

[15] T. Neumann and G. Weikum. RDF-3X: a risc-style engine for RDF. *PVLDB*, 1(1):647–659, 2008.

[16] D. Rao, P. McNamee, and M. Dredze. Entity linking: Finding extracted entities in a knowledge base. In *A book chapter in Multi-source, Multi-lingual Information Extraction and Summarization*, 2011.

[17] K. Riesen, S. Fankhauser, and H. Bunke. Speeding up graph edit distance computation with a bipartite heuristic. In *MLG*, 2007.

[18] D. Suciu and N. N. Dalvi. Foundations of probabilistic answers to queries. In *SIGMOD Conference*, 2005.

[19] W. Tunstall-Pedoe. True knowledge: Open-domain question answering using structured knowledge and inference. *AI Magazine*, 31(3):80–92, 2010.

[20] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over rdf data. In *WWW*, pages 639–648, 2012.

[21] G. Wang, B. Wang, X. Yang, and G. Yu. Efficiently indexing large sparse graphs for similarity search. *TKDE*, 24(3):440–451, 2012.

[22] X. Wang, X. Ding, A. K. H. Tung, S. Ying, and H. Jin. An efficient graph indexing method. In *ICDE*, pages 210–221, 2012.

[23] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the web of data. In *EMNLP-CoNLL*, pages 379–390, 2012.

[24] M. Yahya, K. Berberich, S. Elbassuoni, and G. Weikum. Robust question answering over the web of linked data. In *CIKM*, pages 1107–1116, 2013.

[25] X. Yao and B. V. Durme. Information extraction over structured data: Question answering with freebase. In *ACL*, pages 956–966, 2014.

[26] X. Yao, B. V. Durme, C. Callison-Burch, and P. Clark. Answer extraction as sequence tagging with tree edit distance. In *NAACL*, pages 858–867, 2013.

[27] Y. Yuan, G. Wang, L. Chen, and H. Wang. Efficient subgraph similarity search on large probabilistic graph databases. *PVLDB*, 5(9):800–811, 2012.

[28] Y. Yuan, G. Wang, H. Wang, and L. Chen. Efficient subgraph search over large uncertain graphs. *PVLDB*, 4(11):876–886, 2011.

[29] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *PVLDB*, 2(1):25–36, 2009.

[30] X. Zhao, C. Xiao, X. Lin, Q. Liu, and W. Zhang. A partition-based approach to structure similarity search. *PVLDB*, 7(3):169–180, 2013.

[31] X. Zhao, C. Xiao, X. Lin, and W. Wang. Efficient graph similarity joins with edit distance constraints. In *ICDE*, pages 834–845, 2012.

[32] Z. Zheng, F. Li, M. Huang, and X. Zhu. Learning to link entities with knowledge base. In *NAACL*, pages 483–491, 2010.

[33] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao. Natural language question answering over RDF: a graph data driven approach. In *SIGMOD Conference*, pages 313–324, 2014.

[34] Z. Zou, H. Gao, and J. Li. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *KDD*, pages 633–642, 2010.

B. PROOF OF LEMMA 3

PROOF. (proof by induction)

Let $dif'(q, g^c)$ denote $\sum_{i=1}^{i=m} d(u_i, q) \ominus d(f(u_i), g^c)$. We just need to prove $dif'(q, g^c) \geq dif(q, g^c)$.

1) <u>(Base Case.)</u>

When $m = 1$, it is straightforward that $dif'(q, g^c) \geq dif(q, g^c)$.

When $m = 2$, assume the two vertex degree sets are $A = \{a_1, a_2\}$ $(a_1 \geq a_2)$, $B = \{b_1, b_2\}$ $(b_1 \geq b_2)$, we need to prove $dif(q, g^c)=(a_1 \ominus b_1) + (a_2 \ominus b_2) \leq dif'(q, g^c)=(a_1 \ominus b_2) + (a_2 \ominus b_1)$. There are four cases to be discussed as follows:

Case 1: When $a_1 \leq b_1$ and $a_2 \leq b_2$, $dif(q, g^c)= 0$, which is minimum.

Case 2: When $a_1 \leq b_1$ and $a_2 > b_2$, $dif(q, g^c) = a_2 - b_2$.

$$\begin{aligned}&dif'(q, g^c) = (a_1 \ominus b_2) + (a_2 \ominus b_1)\\&= (a_1 \ominus b_2) \geq (a_2 \ominus b_2) = dif(q, g^c)\end{aligned}$$

Case 3: When $a_1 > b_1$ and $a_2 \leq b_2$, $dif(q, g^c)= a_1 - b_1$.

$$\begin{aligned}&dif'(q, g^c) = (a_1 \ominus b_2) + (a_2 \ominus b_1)\\&= (a_1 \ominus b_2) = (a_1 - b_2)\\&\geq (a_1 - b_1) = (a_1 \ominus b_1) = dif(q, g^c)\end{aligned}$$

Case 4: When $a_1 > b_1$ and $a_2 > b_2$, $dif(q, g^c) = (a_1 - b_1) + (a_2 - b_2)$.

$$\begin{aligned}&dif'(q, g^c) = (a_1 \ominus b_2) + (a_2 \ominus b_1)\\&= a_1 - b_2 + a_2 \ominus b_1 \geq a_1 - b_2 + a_2 \ominus b_1\\&= dif(q, g^c)\end{aligned}$$

To summarize, we prove that $dif'(q, g^c) \geq dif(q, g^c)$, where there are only two vertices in q and g^c, respectively.

2) (Hypothesis.) Provided that when $m \geq k$, $dif'_m(q, g^c) \geq dif_m(q, g^c)$.

(Induction.) Let us consider the case $m = (k+1)$. $dif_{k+1}(q, g^c) = dif_k(q, g^c) + a_{k+1} \ominus b_{k+1}$. Assume that $f(a_{k+1}) = b_i$ ($i < k + 1$) and $f^{-1}(b_{k+1}) = a_j$ ($j < k + 1$), thus, we have

$$dif'_{k+1}(q, g^c) = dif'_{k-1}(q, g^c) + a_{k+1} \ominus b_i + a_j \ominus b_{k+1}.$$

Let us consider $a_{k+1} \ominus b_i + a_j \ominus b_{k+1}$. According to the non-increasing order, it is straightforward to know $a_j > a_{k+1}$ and $b_i > b_{k+1}$. Hence, according to the proof about $m = 2$ in the basis, we know

$$a_{k+1} \ominus b_i + a_j \ominus b_{k+1} \geq a_j \ominus b_i + a_{k+1} \ominus b_{k+1}$$

Therefore,

$$\begin{aligned}dif'_{k+1}(q, g^c) &= dif'_{k-1}(q, g^c) + (a_{k+1} \ominus b_i) + (a_j \ominus b_{k+1})\\&\geq dif'_{k-1}(q, g^c) + (a_j \ominus b_i) + (a_{k+1} \ominus b_{k+1})\\&= dif'_k(q, g^c) + (a_{k+1} \ominus b_{k+1})\end{aligned}$$

According to hypothesis, we know $dif'_k(q, g^c) \geq dif_k(q, g^c)$.

$$\begin{aligned}dif'_{k+1}(q, g^c) &\geq dif'_k(q, g^c) + (a_{k+1} \ominus b_{k+1})\\&\geq dif_k(q, g^c) + (a_{k+1} \ominus b_{k+1})\\&= dif_{k+1}(q, g^c)\end{aligned}$$

3) According to the above analysis, we can prove that $dif_{k+1}(q, g^c) \geq dif'_{k+1}(q, g^c)$ if $dif_k(q, g^c) \geq dif'_k(q, g^c)$. Since we have proved the basis, according to induction method, we know Lemma 3 holds. □

C. PROOF OF THEOREM 2

PROOF. Without loss of generality, we assume that $|V(q)| = m < |V(g^c)| = n$.

$$\begin{aligned}&lb\_ged_{CSS}(q, g^c) - lb\_ged_{LM}(q, g^c)\\&= |V(g^c)| + |E(g^c)| - \lambda_E(q, g^c) + \frac{dif(q, g^c)}{2} - \lambda_V(q, g^c)\\&-(|V(g^c)| - \lambda_V(q, g^c) + \max\{|E(q)|, |E(g^c)|\} - \lambda_E(q, g^c))\\&= |E(g^c)| - \max\{|E(q)|, |E(g^c)|\} + \frac{dif(q, g^c)}{2}\end{aligned}$$

1) When $|E(g)| \geq |E(q)|$,

$$\begin{aligned}&lb\_ged_{CSS}(q, g^c) - lb\_ged_{LM}(q, g^c)\\&= |E(g^c)| - \max\{|E(q)|, |E(g^c)|\} + \frac{dif(q, g^c)}{2}\\&= |E(g^c)| - |E(g^c)| + \frac{dif(q, g^c)}{2}\\&= \frac{dif(q, g^c)}{2} \geq 0.\end{aligned}$$

2) When $|E(g^c)| < |E(q)|$,

$$\begin{aligned}&lb\_ged_{CSS}(q, g^c) - lb\_ged_{LM}(q, g^c)\\&= |E(g^c)| - |E(q)| + \frac{dif(q, g^c)}{2}\end{aligned}$$

Given two certain graphs q and g^c, it is straightforward to know $|E(q)|-|E(g^c)| = \frac{\sum_{u \in V(q)} d(u, q) - \sum_{v \in V(g^c)} d(v, g^c)}{2}$.

Since we know $|V(q)| = m < |V(g^c)| = n$, according to the definition of $dif(q, g^c)$ in Eq. 9, we have the following equation.

$$\begin{aligned}|E(q)| - |E(g^c)| &= \frac{\sum_{i=1}^{i=m} d(u_i, q) - \sum_{j=1}^{j=n} d(v_j, g^c)}{2}\\&\leq \frac{\sum_{i=1}^{i=m} (d(u_i, q) - d(v_i, g^c))}{2}\\&= \frac{dif(q, g^c)}{2}\end{aligned}$$

Hence, we know

$$\begin{aligned}&lb\_ged_{CSS}(q, g^c) - lb\_ged_{LM}(q, g^c)\\&\geq |E(g^c)| - |E(q)| + \frac{dif(q, g^c)}{2}\\&\geq |E(g^c)| - |E(q)| + (|E(q)| - |E(g^c)|) = 0\end{aligned}$$

To summarize, $lb\_ged_{CSS}(q, g^c) \geq lb\_ged_{LM}(q, g^c)$ always holds, i.e., CSS-based is tighter than the label-multiset-based lower bound in [31]. □

D. ALGORITHM OF SIMJ PROCEDURE

D.1 SimJ Algorithm

Algorithm 1 SimJ_Procedure(D, U, τ, α)

```
Require: deterministic graphs $D$, uncertain graphs $U$, graph edit distance threshold $\tau$, and similarity probability threshold $\alpha$
Ensure: graph pairs $\{\langle q, g\rangle \in D \times U | SimP_\tau(q, g) \geq \alpha\}$
 1: $AS \leftarrow \emptyset$
 2: for $\langle q, g\rangle \in D \times U$ do
 3:   compute $lb\_ged_{CSS}(q, g)$ according to Theorem 3
 4:   if $lb\_ged_{CSS}(q, g) \leq \tau$ then
 5:     compute $ub\_SimP_\tau(q, g)$ according to Theorem 4
 6:     if $ub\_SimP_\tau(q, g) \geq \alpha$ then
 7:       $SimP_\tau(q, g) \leftarrow 0$
 8:       for possible world $pw(g) \in PW(g)$ do
 9:         compute $lb\_ged_{CSS}(q, pw(g))$ by Theorem 2
10:         if $lb\_ged_{CSS}(q, pw(g)) \leq \tau$ then
11:           compute $ged(q, pw(g))$
12:           if $ged(q, pw(g)) \leq \tau$ then
13:             $SimP_\tau(q, g) \leftarrow SimP_\tau(q, g) + Pr(pw(g))$
14:       if $SimP_\tau(q, g) \geq \alpha$ then
15:         put $\langle q, g\rangle$ into $AS$
16: return $AS$
```

D.2 Complexity Analysis

Complexity of computing CSS-based lower bound of GED for uncertain graphs (line 3 in Algorithm 1). Let us recall the lower bound $lb\_ged_{CSS}(q,g) = |V| + |E| - \lambda_E(q,g) + \frac{dif(q,g)}{2} - \lambda_V(q,g)$. Its dominating computation is caused by $\lambda_V(q,g)$, i.e., the size of the maximal matching in the vertex label bipartite graph. The maximal matching problem can be sovled by a classic Hungarian algorithm [10], with time complexity $O(|V|^3)$. Hence, the complexity of computing CSS-based lower of GED for uncertain graphs is $O(|V|^3)$, where $|V| = \max\{|V(g)|, |V(q)|\}$.

Complexity of computing upper bound of $SimP_\tau(q,g)$ for uncertain graphs (line 5 in Algorithm 1). Let us recall the upper bound $ub\_SimP_\tau(q,g) = \frac{E(Y)}{C(q,g)-\tau}$, where $E(Y) = \sum_{i=1}^{m} E(y_i)$ can be computed with the time complexity $O(|V| \cdot |l(v)|)$. The time complexity of computing $C(q,g)$ is $O(|V(q)| \cdot |V(g)|)$. Hence, the time complexity $O(min\{|V| \cdot |l(v)|, |V(q)| \cdot |V(g)|\})$.

Complexity of computing CSS-based lower bound of GED for certain graphs (line 9 in Algorithm 1). Let us recall the lower bound $lb_g ed_{CSS}(q,g^c) = |V(g^c)| + |E(g^c)| - \lambda_E(q,g^c) + \frac{dif(q,g^c)}{2} - \lambda_V(q,g^c)$, where $|V(g^c)|$ and $|E(g^c)|$ are constants. Since the computation complexity of $\lambda_V(q,g^c)$ and $\lambda_E(q,g^c)$ is $O(|V(q)| \cdot |V(g^c)| + |E(q)| \cdot |E(g^c)|)$ and the computation complexity of $\frac{dif(q,g^c)}{2}$ is $O(|V(q)|)$, the complexity of computing CSS-based lower of GED for certain graphs is $O(|E(q)| \cdot |E(g^c)|)$.

Complexity of verification (lines 8-15 in Algorithm 1). In the verification phase, we need to compute $SimP_\tau(q\ ,g)$ by enumerating all possible worlds in the worst case. Moreover, it invokes the GED computation, which is a well-known NP-hard problem. Nevertheless, it is feasible to compute the graph edit distance since the semantic query graphs and SPARQL graphs are small in real applications.

Note that our proposed upper/lower bounds are efficient with the time complexity $O(max\{|E(q)| \cdot |E(g^c)|, |V|^3\})$ even if the verification procedure is NP-hard.

E. ALGORITHM OF GROUP-BASED OPTIMIZATION

Algorithm 2 SimJ_OPT(q, g, τ, α)

```
Require: uncertain graph $q$, deterministic graph $g$, graph edit
    distance threshold $\tau$, and similarity probability threshold $\alpha$
Ensure: $ub\_SimP_\tau(q,g)$
1: $SimP_\tau(q,g) \longleftarrow 0$
2: divide $g$ into $k$ possible world groups $PWG_1, \ldots, PWG_k$
3: for $i = 1$ to $k$ do
4:    compute $lb\_ged_{CSS}(q, PWG_i)$ according to Theorem 3
5:    if $lb\_ged_{CSS}(q, PWG_i) \leq \tau$ then
6:       compute $ub\_SimP_\tau(q, PWG_i)$ according to Theorem 4
7:       $t \longleftarrow ub\_SimP_\tau(q, PWG_i)$
8:       $ub\_SimP_\tau(q,g) \longleftarrow ub\_SimP_\tau(q,g) + t$
9: return $ub\_SimP_\tau(q,g)$
```

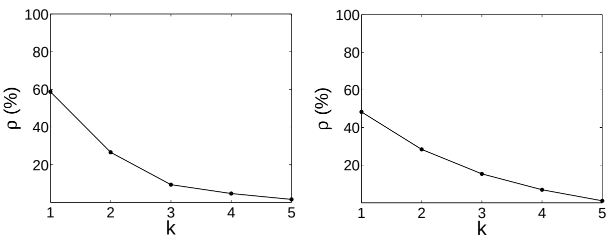
F. EXPERIMENTAL RESULTS

F.1 Result Analysis

Effect of the number of relations k. We have analyzed the experimental results carefully, and find that the simple query patterns are easily recognized. Figure 17 presents the effect of the number of relations over the datasets of QALD-3 and WebQ, where the horizontal x-axis represents the number of relations (denoted by k), and vertical y-axis represents the proportion of correct patterns (i.e., the number of correct patterns with k relations divided by all correct patterns), denoted by ρ. It shows that the patterns with few relations are better recognized. The reason is that if a natural language question is complex, the generated sematic query graph may be incorrect probably.

What is the ruling party in <___>?	SELECT ?uri WHERE { <___> type <___>. <___> leaderParty ?uri. }
Give me all <___> directed by <___>	SELECT ?film WHERE { ?film type <___>. ?film director <___>. }
Which <___> has been developed by organizations founded in <___>?	SELECT ?product WHERE { ?company type Organization. ?company foundationPlace <___>. ?product developer ?company. ?product type <___>. }

Figure 16: Templates for the case study



(a) Results on QALD-3 (b) Results on WebQ

Figure 17: Effect of k (the number of relations)

Failure analysis. We also provide the failure analysis for the templates generated by our method. There are two key reasons for the errors. The first reason is that it fails to generate correct semantic query graphs for some natural language questions. For example, in the question "Who composed the music for Harold and Maude?", it fails in linking "Harold and Maude" to the corresponding entity "Harold_and_Maude". The second one is that some graph pairs have small graph edit distance, but they do not share the same query intension. We give the ratio of each reason in Figure 18.

Reason	Incorrect Semantic Query Graph	Graph Edit Distance	Others
#(Ratio)	73%	21%	6%

Figure 18: Failure Analysis

F.2 Q/A Using Templates

Comparisons With Other Systems. In order to evaluate the quality of the templates, we use the questions of QALD-3 to test the effectiveness. We translate the questions into SPARQL queries utilizing these generated templates, and then search these SPARQL queries over DBpedia.

We utilize the evaluation measures that are also used in QALD task. The precision, recall and F-measure for each question n are computed as follows. Then the overall precision and recall are computed by taking the average mean of all single precision and recall values, as well as the overall

F-measure[11].

$$Recall(n) = \frac{\text{number of correct system answers}}{\text{number of gold standard answers}}$$

$$Precision(n) = \frac{\text{number of correct system answers}}{\text{number of system answers}}$$

$$\text{F-Measure(n)} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

To confirm the effectiveness of our method, we compare it with gAnswer [33] and DEANNA [23]. As shown in Table 4, our template-based method is more effective, which indicates that the generated templates have high quality.

Table 4: Q/A results compared with other systems

Methods	**Precision**	**Recall**	**F-1**
Our method	0.65	0.65	0.65
gAnswer	0.41	0.41	0.41
DEANNA	0.21	0.21	0.21

Effect of matching proportion ϕ. If we cannot find a match between a template with the entire sentence of a new natural language question (NLQ for short), we choose the template that maximizes the matching proportion ϕ.

We can also generate SPARQL queries based on this partial match. To evaluate the effect of ϕ, we conduct experiments over real datasets by varying ϕ from 0.5 to 1. Table 5 shows that allowing partial template match improves both the recall and precision of RDF Q/A task. The main reason is by allowing partial match, we can answer more questions without affecting those can be answered correctly by setting $\phi = 1$ (i.e., take no account of the partial match).

Table 5: Effect of ϕ

ϕ	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
Precision	0.69	0.69	0.68	0.66	0.66	0.65
Recall	0.73	0.72	0.70	0.69	0.67	0.65
F-1	0.71	0.70	0.69	0.67	0.66	0.65

[11] http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/4/qald-4.pdf