Interactive Knowledge Graph Querying through Examples and Facets (Long Version)

Yael Amsterdamer
Bar-Ilan University

Laura Gáspár
Bar-Ilan University

Abstract

Knowledge graphs are a highly useful form of information representation. To assist end users in understanding the contents of a given graph, multiple lines of research have proposed and studied various data exploration tools. Despite major advancements, it remains highly non-trivial to find entities of interest in a large-scale graph where the user requirements may depend on the initially unknown contents and structure of the graph. We provide in this paper a formal approach for the problem, which combines in a novel way ideas from two approaches: query-by-example and faceted search. We first provide a novel model for user interaction that includes different formal semantics for interpreting the answers. The semantics correspond to natural interpretations of feedback in faceted search. We show that for each of these semantics, any sequence of user feedback may be encoded as a SPARQL query under standard closed-world semantics. We then turn to the problem of iteratively choosing which user feedback to prompt in order to optimize the expected length of interaction. We show that depending on the probabilities of user answers, the optimal choice of question may depend on the semantics; in contrast, we show that for a natural way of estimating the probabilities, the optimal choices coincide.

1 Introduction

The widespread adoption of knowledge graphs as means for representing information calls for effective ways to allow users to query and explore them. SPARQL, the predominant query language for RDF graphs, allows specifying complex data selection criteria. Yet, writing formal queries requires the user not only to master the query language, but also to be familiar with the contents and structure of the queried knowledge graph, and to have a crisp notion of a question to be asked over this data. Due to the difficulty of these tasks in light of the the increasing scale of knowledge graphs, assisting the uninformed user in identifying *relevant parts thereof* is a crucial need.

These challenges are well known and have been extensively studied, leading to the development of dedicated *data exploration tools*. There is a wide range of approaches and technologies for this task (see [11] for a survey). A particularly prominent approach is that of *query-by-example* (e.g., [1, 2, 6, 13, 15, 16, 19, 23, 28, 32, 38]), which aims at the "reverse-engineering" of a query from output examples provided or evaluated by the user. The work of [2, 6, 15, 23] has

developed such solutions specifically for SPARQL. Another prominent approach, that of *faceted search*, allows users to browse through criteria that may be added to a gradually forming query, typically by providing a friendly interface for criteria selection, and dynamically updating the results (e.g., [7, 8, 14, 20, 33, 34]).

Despite this great progress, the problem is far from being solved, and specifically, in many cases the user is unable to provide even a single output example (for query-by-example) or to effectively browse through the list of properties to describe it (for faceted search). Consider, for instance, a criminologist examining governmental data published in RDF, with the goal of studying properties of "interesting criminals". Since this user is not familiar with the contents of the repository and her query is not well-formulated, she may struggle in finding relevant information. Presenting the user example entities and properties may help her discover, e.g., that "a criminal" in this repository is identified through `convictedOf` properties, or realize that the data contains many historic convicts while she is interested in relatively recent convictions of living people (and what properties identify this irrelevant data).

In this paper, we develop an approach for querying a given knowledge graph for entities based on their properties without prior knowledge of the graph contents. This approach "marries" ideas from query-by-example and faceted search. Briefly, users are not requested to input example entities, but rather the framework selects example entities to show the user in order to interactively identify the relevant properties. Importantly, we observe that for knowledge graphs, it is difficult for a system to select output examples that would converge to a query in reasonable time: entity properties are usually numerous, sparse, and partly unknown; so hitting entities that have the desired properties is unlikely. To this end, our framework not only presents examples but also asks about (carefully selected) subsets of their properties, in the spirit of faceted search. Following interactive variants of query-by-example, the subsets of properties are chosen to maximize the expected information gain at each step; this may greatly speed up the convergence to the user intention.

This paper lays formal foundations for the proposed approach, making the following contributions.

Formal Model for User Interaction. Our first contribution is a formal model for user interactions. The model is based on fairly standard notions of questions and answers; questions are properties of entities in the knowledge graphs and answers take the form of "yes"/"no"/"Don't care". The novelty lies in the interpretations of answers: our main observation is that since knowledge graphs are inherently incomplete, an entity may in fact match the desiderata of the user even if a requested property is not present with respect to it. We provide four different semantics for interpreting user feedback and defining its effect on qualifying entities. We show that there is a chain of inclusion between the sets of qualifying entities according to the four semantics. Thus, the semantics may be viewed as means of balancing the precision and recall of queries, taking into account the incomplete nature of knowledge graphs.

Encoding in Standard Semantics. The end result of the interaction is a specification of the properties that make an entity interesting to the user.

We show that regardless of the semantics of user interaction, the end result may be encoded as a SPARQL query under standard closed-world semantics that SPARQL engines use, and that further the resulting SPARQL query is a member of a simple SPARQL fragment. Retrieving the relevant entities then simply involves invoking a SPARQL query engine.

Choosing the Right Questions. Given a choice of semantics for user interaction, the problem is then to choose questions in a way that minimizes the set of candidate entities. We provide a natural probabilistic formulation of the problem and show that, in general, the optimal choice of question may depend on the semantics assigned to user feedback. In contrast, we show a concrete way of inferring the probabilistic distribution of anticipated future answers based on statistics of past answers, and show that for this particular way of setting the probabilities, the optimal question to ask no longer depends on the semantics of user feedback.

The rest of this paper is organized as follows. Section 2 describes the theoretical model underlying our solution. Section 3 discusses the choice of questions to be asked in the course of interaction in order to optimize it. Section 4 overviews related work and we conclude in Section 5.

2 Model

We next overview a standard model of knowledge graphs, and define our notion of user interaction.

2.1 Knowledge Graphs

We use here a simple knowledge graph model in the spirit of languages such as RDF and OWL, abstracting away details that are not necessary for our setting. Let $\mathcal{E}$ and $\mathcal{P}$ be sets of element ids and property ids respectively, standing for entities/concepts from the knowledge domain, and their properties. We further allow defining literal values from Σ^* assuming $\Sigma^* \cap \mathcal{E} = \Sigma^* \cap \mathcal{P} = \emptyset$.

Definition 2.1 (Facts and knowledge graph)**.** *A* fact *over $\mathcal{E}$, $\mathcal{P}$, Σ^* is a triple of the form* {subject property object} *where* subject $\in \mathcal{E}$, property $\in \mathcal{P}$ *and* object $\in \mathcal{E} \cup \Sigma^*$. *A knowledge graph is a set of facts.*

Example 2.2. *The knowledge graph could include facts* {`Saddam_Hussein type Leader`} *and* {`Saddam_Hussein birthDate "1937-04-28"`}. *In this case,* `Saddam_Hussein`, `Leader`$\in \mathcal{E}$, `type`,`birthDate`$\in \mathcal{P}$ *and* `"1937-04-28"`$\in \Sigma^*$.

A knowledge graph can be also viewed as a labelled directed graph with parallel edges and self-edges, where the vertices are elements in $\mathcal{E}$ or Σ^*, and every directed edge (subject, object) is labelled by some property $\in \mathcal{P}$. We interpret the meaning of a knowledge graph under *an open world assumption*: facts in the knowledge graph are asserted to be true and facts not in it may be true or false. This allows incompleteness, which typically holds in practice.

2.2 Interaction Model

Let u be a user who seeks entities in the knowledge graph $\mathcal{G}$. Denote by E_u the set of all entities that are acceptable by u. We pose questions to this user in order to identify entities in E_u. For that, we ask u questions that can be interpreted as "Should entities in E_u have property p with object o?" e.g., "Should entities in E_u have the property `type` with object `Leader`?" (See Section 5 for a discussion of additional question forms.) Formally,

Definition 2.3 (Question and answer model)**.** *A user question in our framework is denoted by $q_{p,o}$ where $p \in \mathcal{P}$ and $o \in \mathcal{E} \cup \Sigma^* \cup \{\texttt{[]}\}$. In response a user u chooses an answer $a \in \{\Box p(o), \Box\neg p(o), \Diamond p(o)\}$, where for $o \neq$ `[]`, $\Box p(o)$ (resp., $\Box\neg p(o)$) implies that for every $e \in E_u$, the fact $\{e \;\; p \;\; o\}$ is true (resp., is false); $\Diamond p(o)$ implies that this fact may or may not hold for any $e \in E_u$ ("don't care"). If $o =$ `[]`, $\Box p($`[]`$)$ (resp., $\Box\neg p($`[]`$)$) implies that for every $e \in E_u$ there is some (no) value $o' \in \mathcal{E} \cup \Sigma^*$ such that the fact $\{e \;\; p \;\; o'\}$ is true (resp.,is false).*

The user feedback may then be encoded as a SPRAQL query. To this end we next recall the syntax of a simple fragment of SPARQL, which we refer to as `sSPARQL`. It consists of the following basic patterns for facts.

Definition 2.4. *A* fact pattern *is a triple* `$e p o` *where* `$e` *is a variable (fixed for a given query) and* `p` $\in \mathcal{P}$ *and* `o` $\in \mathcal{E} \cup \Sigma^* \cup \{$`[]`$\}$. *A fact* $\{s$ pred obj$\}$ matches *the pattern if* `p` = pred, *if either* `o` = obj *or* `o` = `[]` *(the latter stands for an undistinguished variable, allowing any value to be assigned to* `[]`*). s is assigned to* `$e`*. We say that an entity s matches a pattern P with respect to a knowledge graph $\mathcal{G}$ if there exists a fact $\{s$ pred obj$\} \in \mathcal{G}$ that matches P.*

An `sSPARQL` query is composed of a `SELECT` and `WHERE` clauses, as well as an optional `MINUS` operator.

`WHERE`**.** The `WHERE` clause consists of groups of fact patterns (in curly brackets), which may be nested in `MINUS` operators.

`SELECT`**.** Queries in our fragment always contain the clause `SELECT DISTINCT $e`, which means the result of a query Q over a knowledge graph $\mathcal{G}$, denoted $Q(\mathcal{G})$, includes every distinct assignment to `$e` that is consistent with the query contents (whose semantics are defined below).

Example 2.5. *Figure 1a illustrates a query selecting all human convicts of who are still alive and have not been convicted of war crimes. This is done by selecting entities (assignments to* `$e`*) that appear in a fact with a* `convictedOf` *predicate (with any object), have a type* `Person`*, do not appear in a fact with a* `deathPlace` *predicate (with any object), and do not appear in a fact with* `convictedOf` *predicate and* `WarCrimes` *object.*

Given a set of questions and respective answers $\{\langle q_{p_1,o_1}, a_1\rangle,$ $\ldots, \langle q_{p_n,o_n}, a_n\rangle\}$ after n interaction steps, we encode them in a query Q_n, as follows.

- $a_i = \Box p_i(o_i)$ is encoded by `$e` p_i o_i.
- $a_i = \Box\neg p_i(o_i)$ is encoded by `MINUS{$e` p_i o_i`}`.

```
SELECT DISTINCT $e
WHERE {$e type Person.
       $e convictedOf []
       MINUS {$e deathPlace []}
       MINUS {$e convictedOf WarCrimes}.}
```

(a) Q_{ex} under "standard" Closed-world semantics.

```
SELECT DISTINCT $e
WHERE {$e [] []
      MINUS {$e deathPlace []}
      MINUS {$e convictedOf WarCrimes}.}
```

(b) Equivalent of Q_{ex} under Open-world Semantics.

```
SELECT DISTINCT $e
WHERE {{$e type Person} UNION {$e convictedOf []}
      MINUS {$e deathPlace []}
      MINUS {$e convictedOf WarCrimes}.}
```

(c) Equivalent of Q_{ex} under Step-wise Semantics.

```
SELECT DISTINCT $e
WHERE $e convictedOf []
      MINUS {$e deathPlace []}
      MINUS {$e convictedOf WarCrimes}.}
```

(d) Equivalent of Q_{ex} under Weighted Semantics.

Figure 1: Variations of example selection query Q_{ex}.

Note that Q_n does not encode $\Diamond p(o)$, as it has no effect on the selected entities (but we record the answer, to avoid repeating a question more than once).

The query can be evaluated according to the aforementioned four semantics, which corresponds to different levels of strictness (Closed-world is most strict) or flexibility (Open-world is the most inclusive in adhering to the user input.

We illustrate the translation from a user interaction to an `sSPARQL` query via an example.

Example 2.6. *The query in Figure 1a is an encoding of the answer sequence* $\Diamond$`gender(male)`, $\Box\neg$`deathPlace(Any)`, $\Box$`convictedOf(Any)`,$\Box$`type(Person)`, $\Box\neg$`convictedOf(WarCrimes)`. *Note that the order of answers in not important, and that the first, "Don't care" answer is not encoded.*

2.3 Multiple Semantics for User Interactions

As explained in the Introduction, there are multiple reasonable semantics that could be assigned to the user feedback, and thereby to the `sSPARQL` query corresponding to it. We can also view the different semantics as different ways of balancing the precision and recall of queries over an incomplete knowledge graph with respect to a (complete) ground truth. We next overview these semantics.

Closed-world semantics. Given a query over a knowledge graph $\mathcal{G}$, typical SPARQL engines interpret it as follows: an assignment s to `$e` is consistent with a pattern group if for each pattern p in the group s matches p with respect to $\mathcal{G}$/ (This also holds if a matching fact could be *inferred* from $\mathcal{G}$'s contents, using logical inference rules as in RDF Schema and OWL.) `MINUS` has the usual

meaning of difference. This semantics is closed-world in the sense that we regard missing facts as false.

Open-world Semantics. Closed-world semantics is conservative in the sense it only returns entities that certainly match the query. However, this limits the ability of a user to explore entities with missing information or to err in the interaction with the system. An alternative semantics which complies with the open-world assumption ignores fact patterns in a "positive" context, i.e., not nested in negation. We cannot prune entities for which a matching fact does not occur in the graph, but may still be true. `MINUS` has the same semantics as in Closed-world.

Step-wise Semantics. Closed world semantics may miss valuable results due to missing facts, while Open-world semantics misses important characterization of entities via positive fact patterns. As an intermediate solution, step-wise semantics has the same semantics as closed-world, as long as query result set is not empty. If it is empty, the query results contain all the entities that match a *maximal subset* of the fact patterns in positive context.

Weighted Semantics. Finally, we consider a semantics that accounts for *which* positive fact patterns are matched by each selected entity, and the importance of each constraint, which is not reflected in Open-world and Step-wise semantics. For that, we associate with each fact pattern group $G = \{P_1, \ldots, P_n\}$ an importance weight $\mathrm{w}(G)$. Let $G(\mathcal{G})$ denote the set of entities that match the pattern group under Closed-world Semantics, and let $\{G_1, \ldots, G_m\}$ denote the maximal subsets of positive constraints use in Step-wise semantics. We will then return the entities that match one of $\arg\max_{G_i} \mathrm{w}(G_i)$ as well as the `MINUS` constraints.

An example weight function, in the spirit of information content is $\mathrm{w}(G) = -\log\left(\frac{|G(\mathcal{G})|}{|\mathcal{G}|}\right)$, i.e., inverse to the number of entities matching the group in the knowledge graph $\mathcal{G}$.

We next exemplify the different semantics.

Example 2.7. *Reconsider the query Q_{ex} in Figure 1a, and now assume a small knowledge graph $\mathcal{G}_{\mathrm{ex}}$ that contains the following facts:* `{Saddam_Hussein type Person. Saddam_Hussein deathPlace Baghdad. Silvio_Berlusconi convictedOf Fraud. Angela_Merkel type Person.}` `Saddam_Hussein` *matches the third pattern and hence does not match the first* `MINUS` *constraint, and will not appear in the query result under any of our semantics.*

Under Closed-world semantics, $Q_{\mathrm{ex}}(\mathcal{G}_{\mathrm{ex}}) = \emptyset$, as each entity does not match some constraint: `Silvio_Berlusconi` *and* `Angela_Merkel` *do not match the first pattern and the second patterns respectively.*

Under Open-world semantics, we ignore the positive constraints and will return both of these entities.

Under Step-wise semantics, we have two maximal subsets of the patterns in positive context that match some entity: `{$e type Person}` *and* `{$e convicted of []}`*, matched by* `Silvio_Berlusconi` *and* `Angela_Merkel` *respectively, Hence in this case, Step-wise semantics yields the same output as Open-world.*

Finally, assuming a weight function inverse to number of matching entities, the second pattern will have a higher weight (having only one matching person, as opposed to two matching the first pattern). This means we will return the results adhering to the maximal subset of patterns containing only the second pattern, and adhering to the `MINUS` *constraints - only* `Silvio_Berlusconi`*. This makes sense if, intuitively, the omission of* `Silvio_Berlusconi type Person` *is more likely than the omission of* `Angela_Merkel convictedOf o` *for any* `o`*.* □

We may show:

Proposition 2.8. *For any Q and $\mathcal{G}$, it holds that*

$$Q_{\text{Closed-world}}(\mathcal{G}) \subseteq Q_{\text{Weighted}}(\mathcal{G}) \subseteq Q_{\text{Step-wise}}(\mathcal{G}) \subseteq Q_{\text{Open-world}}(\mathcal{G})$$

Assume we start with the entire domain $\mathcal{E}$ as the set of candidate entities. We may also show that under Closed-world and Open-world semantics, a sequence of questions and answers is yields queries whose set of answers monotonically decreases. This matches the intuition that query answers are equivalent to constraints that serve to narrow down the set of candidate entities.

Proposition 2.9. *Let $\langle q_{p_1,o_1}, a_1\rangle, \ldots, \langle q_{p_n,o_n}, a_n\rangle$ be a sequence of questions and answers. For every i, $1 \leq i < n$, and any input graph $\mathcal{G}$, $Q^{i+1}_{\text{sem}}(\mathcal{G}) \subseteq Q^{i}_{\text{sem}}(\mathcal{G})$, where Q^i is the query encoding of $\{\langle q_{p_1,o_1}, a_1\rangle, \ldots, \langle q_{p_i,o_i}, a_i\rangle\}$, and $Q^i_{\text{sem}}(\mathcal{G})$ is its evaluation on $\mathcal{G}$ under semantics* sem *which is one of Closed-world, Open-world.*

This monotonicity property does not hold for Step-wise and Weighted semantics, since as more constraints are added, new fact pattern subsets that select additional entities may be formed.

Encoding in Closed-world Semantics. Since standard SPARQL engines evaluate queries under Closed-world semantics, it may be useful to encode queries under other semantics by their Closed-world equivalents. Indeed, we may show:

Proposition 2.10. *For every* `sSPARQL` *query Q and a Knowledge Graph G, there exist* `sSPARQL` *queries Q^1, Q^2, Q^3 such that*

- $Q_{\text{Step-wise}}(\mathcal{G}) = Q^1_{\text{Closed-World}}(\mathcal{G})$
- $Q_{\text{Weighted}}(\mathcal{G}) = Q^2_{\text{Closed-World}}(\mathcal{G})$
- $Q_{\text{Open-World}}(\mathcal{G}) = Q^3_{\text{Closed-World}}(\mathcal{G})$.

Proof. We next provide the encoding for each semantics.

- *Open-world to Closed-world.* Given a query Q under Open-world semantics, a query Q' such that $Q_{\text{Open-world}} \equiv Q'_{\text{Closed-world}}$ can be obtained by omitting the positive constraints from Q.
- *Step-wise to Closed-world.* Given a query Q under Step-wise semantics, a query Q' such that $Q_{\text{Step-wise}}(\mathcal{G}) = Q'_{\text{Closed-world}}(\mathcal{G})$ for a given knowledge graph $\mathcal{G}$ can be obtained by using a `UNION` operation, which operates over

two fact pattern groups and is supported by standard SPARQL engines under the standard (Closed-world) semantics: define a group for each maximal subset of the fact patterns, and replace the positive part of the query by this union. As the semantics depends on the knowledge graph contents, the encoding depends on the input graph.

- Given a query Q under Weighted Semantics, a query Q' s.t. $Q_{\text{Step}-\text{wise}}(\mathcal{G}) = Q'_{\text{Closed}-\text{world}}(\mathcal{G})$ for a given knowledge graph $\mathcal{G}$ and weight function w can be obtained similarly to Step-wise semantics, except that we take the union of the groups with the maximal weight only.

□

Note that for Step-wise and Weighted semantics, the encoding depends on the input graph, since these semantics depend on the (non-)emptiness of queries. The encoding of the latter further depends on the weight function. To bind `$e` when the positive part of the query is empty, we have added the fact pattern `$e [] []`. When a union consists of only one group, we replace it by that group.

Example 2.11. *Figures 1b, 1c and 1d demonstrate, respectively, the encoding of* Q_{ex} *from Figure 1a under Open-world, Step-wise and Weighted semantics.*

3 Optimizing the Interaction

We have defined a model of interaction with users, including formal semantics for interpreting their answers. We next provide a framework for choosing the *questions*, namely combinations of property and object that should/not hold for the sought entities. We next discuss the iterative choice of questions, conditional on the semantics of previous answers.

Problem definition. We focus on the following scenario.

(a) Questions of the form $q_{p,o}$ are the only means of interacting with users, as opposed to e.g., allowing to users to type input such as search strings.

(b) For a given query semantics sem, there exists a query Q in our fragment such that $Q_{\text{sem}}(\mathcal{G}) = E_u$.

(c) Users always give answers that are accurate to their needs by definition 2.3, as opposed to users who may make mistakes.

(d) The process can halt as soon as an entity $e \in E_u$ is presented to the user, as opposed to only when the intended query Q is discovered.

These assumptions follow a "clean" model of exploration focusing on the type of questions that we consider (assumptions (a)-(c)), and allowing for fast convergence (assumption (d)).

Under these assumptions, we consider the choice of questions that would eliminate as many candidate entities as possible, thereby increasing the likelihood of discovering an entity in E_u. The effect of a question in this respect clearly depends on its answer, but the answer is unknown; we therefore adopt a probabilistic model for user answers, and seek to maximize the *expected* number of eliminated entities.

Definition 3.1 (Problem definition)**.** *Denote by* $\Pr_{\text{p,o}}(\Box)$ *and* $\Pr_{\text{p,o}}(\Box\neg)$ *the probability of "must" and "must not", respectively, for a specific question* $q_{p,o}$ *at a certain point of the interaction. The probability of "don't care" is then* $\Pr_{\text{p,o}}(\Diamond) = 1 - \Pr_{\text{p,o}}(\Box) - \Pr_{\text{p,o}}(\Box\neg)$*. Let* E_φ *and* $E_{\varphi'}$ *denote, respectively, the output of the current query* φ*, encoding the user answers thus far, and the output of the next query* φ'*. Let* $E_{p,o} \subseteq E_\varphi$ *be the set of currently relevant entities* e *(i.e., in the output of the current* φ*) with the property* $\{e\ p\ o\}$*. The expected number of eliminated entities for a question* $q_{p,o}$*, under Closed-world semantics is then*

$$\mathrm{E}[|E_\varphi - E_{\varphi'}|] = \Pr_{\text{p,o}}(\Box)\,|E_\varphi - E_{p,o}| + \Pr_{\text{p,o}}(\Box\neg)\,|E_{p,o}|$$

For Open-world semantics we can replace $|E_\varphi - E_{p,o}|$ *by 0. We seek the question* $q_{p,o}$ *that maximizes this quantity.*

We next study the problem assuming that the probabilities used in the above formula are known. We then propose an estimation for these probabilities based on property frequencies and analyze its effect on selected questions.

3.1 Assuming Known Probabilities

Problem Definition 3.1 gives rise to a simple greedy algorithm that computes the expected probability for each combination of property and object, and then poses the question that maximizes this value. We next describe two optimizations to this algorithm.

First, we observe that only questions in $\{q_{p,o} \mid \{s,p,o\} \in \mathcal{G} \wedge s \in E_\varphi\} \cup \{q_{p,[]} \mid \exists o\ \{s,p,o\} \in \mathcal{G} \wedge s \in E_\varphi\}$, i.e., questions on facts about candidate entities, may lead to entity elimination greater than 0. We may thus only consider questions in this set.

Second, to compute the formula we need to compute $E_{p,o}$ for each question in the above mentioned set. To do so in a comparatively efficient way, we can do a single pass over all the facts $\{s,p,o\}$ such that in $s \in E_\varphi$, and with each such fact increment the counter for $E_{p,o}$ and $E_{p,[]}$, ignoring questions already asked. The set of such facts may be retrieved by adding a pattern `$e $p $o` to φ and executing a SPARQL query that returns the values of these three variables.

The number of eliminated entities may differ between different semantics. For example, if the probability to a "must" answer is high, Closed-world semantics will prefer a question corresponding to a pattern that matches few entities. In contrast, Open-world semantics only eliminates entities given a negative answer, and will thus prioritize patterns that match many entities.

Example 3.2. *Assume* $|E_\varphi| = 10$ *and that, among these 10 entities, 6 entities match the pattern* p_1 *and 4 match* p_2*. Let the probabilities for* $\Box, \Box\neg$ *and* $\Diamond$ *be 0.75, 0.25 and 0 respectively for the questions corresponding to the two patterns. In this case, as illustrated by Table 1, for Closed-world semantics we will prefer* p_2 *and for Open-world we will prefer* p_1*.*

3.2 Estimating the probabilities

We next consider the estimation of $\Pr_{\text{p,o}}(\Box)$ and $\Pr_{\text{p,o}}(\Box\neg)$ at each step of the interaction, assuming no external information (e.g. statistics for past user interaction). As preliminary results, we focus on the case where the user is

pattern	$E_{p,o}$	Pr.($\Box$)	Pr.($\Box\neg$)	E[$\lvert E_\varphi - E_{\varphi'}\rvert$
p_1(Closed-world)	4	0.75	0.25	$4 \cdot .75 + 6 \cdot .25 = 4.5$
p_2(Closed-world)	6	0.75	0.25	$6 \cdot .75 + 4 \cdot .25 = \mathbf{5.5}$
p_1 (Open-world)	4	0.75	0.25	$0 + 6 \cdot .25 = \mathbf{1.5}$
p_2 (Open-world)	6	0.75	0.25	$0 + 4 \cdot .25 = 1$

Table 1: Expectation for eliminated entities for different semantics (Example 3.2).

interested in finding a single entity, with each entity in E_φ having an a-priori equal probability of being the chosen one. Let the general probability of "don't care" be $\Pr_\Diamond$, then $\Pr_{\text{p,o}}(\Box) = (1 - \Pr_\Diamond)\frac{|E_{p,o}|}{|E_\varphi|}$, and similarly $\Pr_{\text{p,o}}(\Box\neg) = (1 - \Pr_\Diamond)\frac{|E_\varphi - E_{p,o}|}{|E_\varphi|}$. By substituting these expressions in the expectation formula above, we get that to maximize the expected number of eliminated entities with a single question $q_{p,o}$ it suffices to choose one that maximizes $|E_{p,o}|\,|E_\varphi - E_{p,o}|$.

To extend this estimation to sets of properties, one can define the probability of a combination of answers analogously to a single answer (or as the product of probabilities, assuming independence). One difficulty here, beyond the increased computational complexity, is that now the $\Pr_\Diamond$ factor no longer cancels out, as the number of "don't care" answers can vary; e.g., if $\Pr_\Diamond$ is high, feedbacks with more "don't care" answers are more likely, and should have more weight in the comparison between candidate question sets. Unfortunately, an estimation of $\Pr_\Diamond$ seems to be tightly coupled with external information, since it is related to *how* users define their needs (by how many requirements) rather than *what* needs can be defined by the knowledge graph. A possible simplification adopted by our current preliminary prototype is assuming that properties are roughly independent, namely, for any p, o, p', o' the fraction of entities in $\frac{|E_{p,o}|}{|E_\varphi|}$ resembles $\frac{|E_{p,o} - E_{p',o'}|}{|E_\varphi - E_{p',o'}|}$, and thus it suffices to choose the k questions $q_{p,o}$ with highest $|E_{p,o}|\,|E_\varphi - E_{p,o}|$ values.

For open-world semantics, the first term is equal to 0, but since the second term is proportional to $|E_{p,o}|\,|E_\varphi - E_{p,o}|$, it turns out that, in contrast to Example 3.2 where probabilities do not reflect frequency, for this choice of probabilities, the same questions are selected for Closed-world and Open-world semantics, even though the entities they eliminate are different.

4 Related Work

We overview related work with respect to multiple areas.

4.1 RDF and the Semantic Web

We have overviewed RDF [25] (Resource Description Framework) a standard used to represent information on the web. Many publicly available RDF knowledge bases (KBs) exist including large, general-purpose KBs, for instance, Yago[1]

[1] http://www.yago-knowledge.org/

and DBpedia[2], along with various domain-specific KBs, such as Hansard,[3] the official collection of all parlamentary debates of the UK, or GeoNames, [4] a geographical database that covers all countries. Some of those KBs are automatically generated, while others are built collaboratively by volunteer contributors. These structured KBs have a great potential for the end users in terms of the data they provide, and efforts to create interfaces to ease the interaction with such KBs are thus of great importance.

Several standards extend RDF with enriched semantics, enabling to express complex semantics of relationships between entities, predicates and classes (e.g., RDFS [25] and OWL [31]) or describe domain-specific data (e.g., FOAF for representing social networks [21]). The concrete model that we have developed has accounted for interaction that may be captured via SPARQL queries of particular expressive power; further accounting for semantic relationship between entities that is encoded in the Knowledge Graph itself is an interesting goal for future research.

4.2 Knowledge Graphs Exploration

Indeed, the challenges faced by users interacting with huge Knowledge Bases are well known and there is a vast body of research on knowledge graph exploration; we refer the reader to the comprehensive survey in [11] and the references therein. Still, as noted in [11], existing systems are generally not well-suited to large-scale repositories with a large number of entities each having a large number of properties.

Query by Example. Querying by means of providing examples is a convenient method for (interactive) query refinement. The general idea is that a query can be inferred based on positive/negative output examples and given some assumptions on the query type (e.g., conjunctive queries in relational databases [13, 16, 38]). In an interactive query by example process, the user is repeatedly presented with output examples, and may accept/reject them to refine the query, until the desired query is obtained [2, 13]. Such a process is especially helpful when the user has little or no prior knowledge of the structure of the database.

There is a broad line of work [1, 13, 16, 19, 32, 38] on query by example in relational databases, but these techniques are not directly suitable for querying RDF data because of its rich semantics, non-uniform schema, and because of the open world assumption described above. Query-by-example solutions are scarcer for knowledge graphs, compared to their prominence in the relational settings. Existing query-by-example frameworks for knowledge graphs (e.g., [2, 6, 15, 23]) require the user to provide seed examples; this may be non-trivial as demonstrated above. (In the relational setting there exist systems that do not require such seed examples, e.g., [13, 16].) This may be explained by the unique features of knowledge graphs that render reasonable convergence by enquiring on proactively chosen entities to be unlikely.

[2] http://dbpedia.org
[3] https://hansard.parliament.uk
[4] http://www.geonames.org

Faceted Search. Faceted Search is a popular technique for refining search results by proposing further constraints, such as features of products in e-commerce websites (e.g., [22, 24, 26, 29, 30, 33, 34]) or properties from Wikipedia infoboxes [10, 9]. There is a large body of work employed faceted search over knowledge graphs (e.g., [8, 20, 33]). However, these studies focus on facet computation, hierarchical browsing and visualization. To our knowledge, no previous work in this context has studied the alleviation of the exploration problem by combining query-by-example with principles of faceted search. Further, the formal semantics we have proposed for the interaction process and their formal analysis are novel to our knowledge, and may serve as foundations for the developments of solutions in this context.

4.3 Choosing Examples

Finally, we briefly overview works that are related to the choice of entity examples that we have discussed in Section 3. Recall that at each point of the interactive example selection process, the entities proposed to the user should be selected carefully from the numerous candidate entities that may exist in the large RDF KB, in order to facilitate the interactive entity search while avoiding overwhelming the user. In particular, the set of examples may be restricted in size. This topic is considered also in previous work on query by example [16], mentioned above, and these techniques my also be adapted to our setting. We next further consider related work from the fields of RDF querying and search, and from information retrieval.

RDF result ranking. The first line of works that we mention in this context is related to RDF query output ranking. One option, given keywords provided by the user, one can rank entities by relevance to the keywords [17, 27, 37]. Another option is to rank entities by popularity or frequency estimates. For instance, one can issue keyword queries for each entity or property against a major search engine (such as Google) and store the reported result size in indexes as a popularity estimate [18].

Similarity Search. Similarity search is a means of querying RDF by extending an initial query. For instacne, Zheng et al. introduced in [40] a novel similarity measure of RDF graphs, the semantic graph edit distance. This measure can be used to efficiently select similar subgraphs and thereby extend a given query to capture further similar results.

In our context, we may leverage such techniques in two ways. First, due to the open-world assumption and to the non-uniform schema, finding the relevant candidates for the constructed query may not be straightforward. For instance, some entities that are relevant to the user needs may lack required properties due to incompleteness of the KB or because they have these properties expressed in a different RDF structure. In this case we would like to estimate which entities are similar or likely to be relevant to the constructed query.

Second, we may allow the user to provide feedback of the form "Give me similar results" with respect to a given example. In this case again similarity metrics may be used to rank candidate entities by their similarity to the chosen example.

Diversity. Finally, a related notion of relevance of examples is based on metrics of diversity. Beyond increasing the likelihood that the user will find relevant examples, diversity can also be helpful in the interactive query refinement process, as feedback on diverse examples is intuitively more likely to apply to larger parts of the search space. There are different works on diversifying search results [3, 5, 35, 36, 39]. This can be done, e.g., by clustering the results and choosing examples from each cluster [12, 39], or by using metrics of diversity on the selected results to further filter them [5, 36].

5 Conclusion and Future Work

We have presented a novel approach for the exploration of knowledge graphs that "marries" query-by-example and faceted search techniques, and have provided formal foundations for the approach. The foundations have included formal semantics for the interactive process of interaction and an analysis of their properties. We have further outlined principles of our implementation and interface design, including the choice of entities on which to exemplify properties appearing in questions to be posed to the users.

We have focused here on a basic type of queries ("star" queries with negation), which should be extended to a wider range of queries. Some RDF languages, such as OWL, encode inference rules that allow to derive or contradict facts based on existing facts. An important future research direction is thus studying the interplay between inference and query search in more depth. Finally, our interaction scheme may be seen as complementary to other technologies such as knowledge graph visualization, textual search and reverse-engineering queries from entity relevance information. Combining our approach with these solutions is an important task for future work.

Acknowledgements This research was supported by the Israel Science Foundation (grants No. 1157/16 and 2015/21) and by a grant from the Israel Ministry of Science and Technology. The final authenticated version [4] is available online at `https://doi.org/10.1007/978-3-031-15743-1/_19`, `https://link.springer.com/chapter/10.1007/978-3-031-15743-1_19`.

References

[1] A. Abouzied, D. Angluin, C. H. Papadimitriou, J. M. Hellerstein, and A. Silberschatz. Learning and verifying quantified boolean queries by example. In *PODS*, 2013.

[2] E. Abramovitz, D. Deutch, and A. Gilad. Interactive inference of SPARQL queries using provenance. In *ICDE*, 2018.

[3] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, 2009.

[4] Y. Amsterdamer and L. Gáspár. Interactive knowledge graph querying through examples and facets. In *ADBIS*, volume 1652, 2022.

[5] A. Anagnostopoulos, A. Z. Broder, and D. Carmel. Sampling search-engine results. *World Wide Web*, 9(4):397–429, 2006.

[6] M. Arenas, G. I. Diaz, and E. V. Kostylev. Reverse engineering SPARQL queries. In *WWW*, 2016.

[7] M. Arenas, B. C. Grau, E. Kharlamov, S. Marciuska, and D. Zheleznyakov. Faceted search over ontology-enhanced RDF data. In *CIKM*, 2014.

[8] M. Arenas, B. C. Grau, E. Kharlamov, S. Marciuska, and D. Zheleznyakov. Faceted search over RDF-based knowledge graphs. *J. Web Sem.*, 37-38, 2016.

[9] M. Atzori and C. Zaniolo. Swipe: searching wikipedia by example. In *WWW*, pages 309–312, 2012.

[10] S. Auer and J. Lehmann. What have innsbruck and leipzig in common? extracting semantics from wiki content. 2007.

[11] N. Bikakis and T. K. Sellis. Exploration and visualization in the web of big linked data: A survey of the state of the art. In *LWDM*, 2016.

[12] R. Boim, T. Milo, and S. Novgorodov. Diversification and refinement in collaborative filtering recommender. In *CIKM*, 2011.

[13] A. Bonifati, R. Ciucanu, and S. Staworko. Interactive inference of join queries. In *EDBT*, 2014.

[14] W. Dakka, P. G. Ipeirotis, and K. R. Wood. Automatic construction of multifaceted browsing interfaces. In *CIKM*, pages 768–775, 2005.

[15] G. I. Diaz, M. Arenas, and M. Benedikt. SPARQLByE: Querying RDF data by example. *PVLDB*, 9(13), 2016.

[16] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: an automatic query steering framework for interactive data exploration. In *SIGMOD*, 2014.

[17] S. Elbassuoni and R. Blanco. Keyword search over RDF graphs. In *CIKM*, 2011.

[18] S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum. Searching RDF graphs with SPARQL and keywords. *IEEE Data Eng. Bull.*, 33(1):16–24, 2010.

[19] A. Fariha and A. Meliou. Example-driven query intent discovery: Abductive reasoning using semantic similarity. *PVLDB*, 12(11), 2019.

[20] L. Fuenmayor, D. Collarana, S. Lohmman, and S. Auer. FaRBIE: A faceted reactive browsing interface for multi RDF knowledge graph exploration. In *ISWC*, 2017.

[21] J. Golbeck and M. Rothstein. Linking social networks on the web with FOAF: A semantic web case study. In *AAAI*, pages 1138–1143, 2008.

[22] M. A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, 2006.

[23] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying knowledge graphs by example entity tuples. In *ICDE*, 2016.

[24] S. Kamath, D. Manjunath, and R. Mazumdar. On distributed function computation in structure-free random wireless networks. *IEEE Trans. Information Theory*, 60(1):432–442, 2014.

[25] G. Klyne, J. J. Carroll, and B. McBride. Resource description framework (RDF): Concepts and abstract syntax. *W3C rec.*, 10, 2004.

[26] J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *WWW*, 2008.

[27] W. Le, F. Li, A. Kementsietsidis, and S. Duan. Scalable keyword search on large RDF data. *IEEE Trans. Knowl. Data Eng.*, 26(11), 2014.

[28] Y. Li, H. Yang, and H. V. Jagadish. NaLIX: an interactive natural language interface for querying XML. In *SIGMOD*, 2005.

[29] Y. Mass, M. Ramanath, Y. Sagiv, and G. Weikum. IQ: the case for iterative querying for knowledge. In *CIDR*, 2011.

[30] Y. Mass and Y. Sagiv. Knowledge management for keyword search over data graphs. In *CIKM*, pages 2051–2053, 2014.

[31] D. L. McGuinness, F. Van Harmelen, et al. OWL web ontology language overview. *W3C rec.*, 10:10, 2004.

[32] F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri. S4: top-k spreadsheet-style search for query discovery. In *SIGMOD*, pages 2001–2016, 2015.

[33] B. Qarabaqi and M. Riedewald. User-driven refinement of imprecise queries. In *ICDE*, 2014.

[34] S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. K. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *CIKM*, pages 13–22, 2008.

[35] I. Saidi, S. Amer-Yahia, and S. N. Bahloul. An approach to diversify entity search results. In *ICAASE*, 2014.

[36] Y. Song, D. Zhou, and L. He. Post-ranking query suggestion by diversifying search results. 2011.

[37] T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In *ICDE*, 2009.

[38] Y. Y. Weiss and S. Cohen. Reverse engineering SPJ-queries from examples. In E. Sallinger, J. V. den Bussche, and F. Geerts, editors, *PODS*, 2017.

[39] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *Computer Networks*, 31(11-16), 1999.

[40] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song, and D. Zhao. Semantic SPARQL similarity search over RDF knowledge graphs. *PVLDB*, 9(11), 2016.