X₂Q: Your Personal Example-based Graph Explorer

Matteo Lissandrini, Davide Mottin, Yannis Velegrakis and Themis Palpanas

ABSTRACT
Exploring knowledge graphs can be a daunting task for any user, expert or novice. This is due to the complexity of the schema or because they are unfamiliar with the contents of the data, or even because they do not know precisely what they are looking for. For the same reason there is a significant demand for exploratory methods for this kind of data. We propose X₂Q, a system that facilitates the exploration of knowledge graphs with a hands-on approach. X₂Q embodies the flexible multi-exemplar query paradigm, in which easy to express examples serve as the basis for formulating sophisticated, and hard to express queries. Our system helps building examples in an interactive fashion, by showing results of the partial exemplar query as well as suggestions for improving the current examples. Then, the user feedback is incorporated in our scores to filter the irrelevant suggestions upfront. X₂Q returns answers in real-time on Freebase, one of the largest available knowledge graphs.

PVLDB Reference Format:
DOI: https://doi.org/10.14778/3229863.3236251

1. INTRODUCTION
Recently, the number and the size of knowledge graphs, such as DBpedia (dbpedia.org), YAGO (yago-knowledge.org) and Wikidata/Freebase (wikidata.org), have significantly increased as a consequence of their extensive adoption for searching, organizing, and retrieving information. A knowledge graph is a network in which nodes are entities (e.g., Steven Spielberg, or USA), and edges are relationships among entities (e.g., born in).

Despite their expressiveness, knowledge graphs have no predefined structure nor natural search language. Such complexity has called for novel exploratory methods [11, 2, 4] and search paradigms [10] to support expert and novice users in retrieving the intended information. Recent exploratory methods ask for initial examples, such as a structure or a node, which is a representative of the intended results. An exemplar query is an example member of the answer set [9]. These by-example methods offer an intuitive alternative to convoluted query languages that require expertise in database query languages and in the specific domain. One shortcoming of by-example approaches is that the user has to provide a complete description of the examples [9, 3, 5], this prerequisite might result challenging. For instance, in Freebase to provide an exemplar query that will retrieve American movie directors who won some award, a user should be able to provide an example graph (the query) formed by edges like (Steven Spielberg, born in, U.S.A.), (Steven Spielberg, won, Best Director), and (Steven Spielberg, directed, Schindler’s List), since these facts describe the features of interest. To concoct such detailed example requires a fair knowledge of both the schema and the data. The existing exploration systems do not help the user in formulating the proper query, since they are only convenient interfaces to the underlying query system [12, 2, 4]. Neither they help the user in reformulating their queries. Hence, they fail to guide the user towards novel interesting answers.

We propose X₂Q, a fully interactive, autonomous, example-based system for exploring large and complex knowledge graphs. X₂Q bypasses the aforementioned shortcomings by introducing a simple exploration schema which engages the user in an interactive process, starting from a single entity (optionally, multiple unconnected entities) as the example query. During the exploration, X₂Q dynamically computes a set of additional nodes and edges to expand the query, clarifying the initial example, and providing the user further unknown details and explanations.

Existing Solutions: While previous approaches [2, 4] aim at facilitating query formulation by means of simpler interfaces, our system actively escorts the user towards the intended answers, without the need to have full knowledge of the domain, schema, or of a rigid query language. Our
system is the first that studies different, domain-agnostic, ranking functions for graph-query suggestions. The relevance of each proposed example is assessed through a principled approach, using ranking functions that, contrary to existing systems [5], do not require additional information, such as query logs (even though such information could also be incorporated), but rather integrate user feedback interactively. Moreover, our system is the first to support both queries composed of single examples [10] or multiple unconnected examples [8].

Motivating Example: Consider a person who enjoyed the movie The lord of the rings (LotR) and is especially interested in awards such as the Academy award (AA). Facts pertaining LotR and AA can be located in knowledge graphs like the simplified one in Figure 1. For any movie enthusiast, searching the knowledge graph is hard and the initial information need not be well defined. Hence, they may lookup for the entities LotR and AA, but they do not know how to identify the facts of interest. One possibility to help the user is to show all possible facts for such entities from which to choose the intended aspects. However, this becomes quickly impractical, especially in cases where the relevant entities have hundreds of edges. (For instance, the movie Lord of the Rings in Freebase has more than 400 outgoing edges, and about 40 different relationship types.) Instead, X2Q will engage the user by strategically proposing selected additional information that can be added to the initial example, similar to what happens with text search engines. In our example, the system would suggest one of the winners of the Awards (e.g., Steven Spielberg, or Peter Jackson) as the director of the movie, depending on the entity (see Section 3). Therefore, X2Q provides the functionalities of a query suggestion system, which conveniently helps the user describing instances of entities and relationships of interest. For instance, the system could help a user describing an exemplar query to retrieve structures in the database for movies, actors, and directors that won Academy Awards. X2Q accepts single examples and also examples composed of distinct facts, providing the necessary flexibility for combining partial knowledge into a single answer.

In summary, we showcase X2Q, a search-and-explore interactive system to assist (especially novice) users in exploring large knowledge graphs, without the need of a query language. X2Q harnesses the power and flexibility of the multi-exemplar query paradigm [8], which allows an exemplar query to contain multiple fragments and produces answers that match such structures. The audience will be able to experiment with the expressiveness of the multi-exemplar query paradigm, as well as compare different suggestion mechanisms and ranking functions. We will showcase X2Q with Freebase (the largest knowledge graphs on which this kind of systems have ever been demonstrated), for which X2Q provides suggestions and answers at interactive response times.

With this demo, the participants will appreciate this novel way of searching, i.e., using examples. They will see the translation mechanism and the reasoning, alongside the solutions to the scalability and the semantic challenges that have been implemented in the system.

2. THE X2Q SYSTEM

The X2Q system provides support for two fundamental tasks: (i) example construction and suggestion, and (ii) exemplar-query processing (refer to Figure 2). In the first phase, while the user provides the example, or the examples, of interest, X2Q interactively suggests possible refinements in the form of interesting relationships that can be added to the current query-graph, allowing for queries of arbitrary complexity. During query processing, instead, X2Q will process the query and retrieve top-k results based on the classical exemplar-query [10], or on the multi-exemplar query paradigm [8]. In the following, we briefly overview the technical aspects of these two processes.

Formally, a knowledge graph, is a labeled multi-graph $G=(V_G, E_G, \ell)$, where $V_G$ are the entities (also nodes or vertices), $E_G \subseteq V_G \times V_G$ are the relationships among them (i.e., the edges), and $\ell$ is the labeling function to distinguish various kinds of edges. Moreover we say that $G' \subseteq G$ means that $G'$ is a subgraph of $G$, i.e., $V_{G'} \subseteq V_G \land E_{G'} \subseteq E_G$.

Search on a knowledge graph is usually performed by means of a graph query $Q: (V_Q, E_Q, \ell)$, X2Q processes queries following the exemplar query paradigm. In particular an exemplar graph query is a graph $Q \sqsubseteq K$ of which the answers are all the similar structures (for some similarity relation $\sim$) that are found in the knowledge graph $\{A\sqsubseteq K \mid A \sim Q\}$ [10, 6]. X2Q offers also a suggestion system to help the user in specifying and refining $Q$ interactively.

2.1 Example Construction and Suggestion

The system implements a suggestion algorithm, which takes as input an example $Q$, and returns an expanded query $Q_\varepsilon$, s.t. $Q \sqsubseteq Q_\varepsilon$. The expanded query $Q_\varepsilon$ will then contain the same edges of $Q$, and some additional edges $E_\varepsilon = E_Q \setminus E_Q$. Hence, the focus will be on identifying the elements $E_\varepsilon$ to add to the user query, it follows that the suggestion algorithm $\sigma$ has the role of suggesting edge-expansion, i.e., $\sigma : \mathcal{P}(E_\varepsilon) \rightarrow \mathcal{P}(E_\varepsilon)$. Hence, the first task is Graph Query Suggestion, where given a knowledge graph $\mathcal{K}$ and a query $Q$ the system retrieves a set of $m$ edges $E_\varepsilon \subseteq E_\varepsilon$, $m = |E_\varepsilon|$, with $\rho(Q,e) > 0$, $\forall e \in E_\varepsilon \land \forall e \in E_{\varepsilon} \setminus E_{\varepsilon}$, $\exists e \in E_\varepsilon$ such that $\rho(Q,e) > \rho(Q,e')$. Where $\rho$ is a relevance function that provides scores for each candidate edge expansion. Hence, we model the task of graph query-exploration in a way reminiscent of query expansion for keyword queries.

We note that X2Q does not lead to a vertical construction of a query, in the form of descriptions of the required objects [2, 4], but rather explores the space horizontally, expanding the user knowledge of the query and interacting with the user in order to identify examples of desired results. Consequently, we rank edges in $E_\varepsilon$ according to scores based on the edge labels $l$, as they describe the types of facts of interest. In this demonstration we implement both naïve ranking functions, and advanced scores.

For the sake of brevity, we describe here only their main
intuition. The first is based on a simple maximum likelihood estimation, where the score of a label \( l \) is proportional to its relative frequency around the graph-query and in the entire repository. The likelihood of label \( l \) is then smoothed \( \hat{p}(l|Q) = (|E_Q| + \epsilon p(|K|)) / (|E_Q| + \epsilon) \), where \( |E_Q| \) is the number of edges in \( Q \) and around it, with label \( l \). The second simple method computes the score of a label with the KL-divergence [7], which will favor labels that are frequent around the query, but infrequent in the dataset.

The more advanced ranking techniques, instead, are implemented within the pseudo-relevance feedback framework[1]. With this approach, we would estimate the likelihood of a candidate expansion-edge based on its relative frequency within the pseudo-relevance set, apart from those matching the exemplar query as follows [8]:

**Definition 1.** An answer to a multi-exemplar query represented by the set of user samples \( S \) on the database \( G = (V, E, \ell) \) is a subgraph \( A \subseteq G \), such that \( \forall s \in S, s \subseteq A \).

Note that Definition 1 does not constrain the size of the answer. However, with no bounds, even the entire graph is accepted by definition, which is useless. On the same token, answers should not include information (regarding nodes and edges) that is extraneous to the user request and should represent a complete concept or situation.

Therefore, two properties need to be satisfied: first, connectedness, so that the subgraphs that are isomorphic to each sample should be connected in the answer graph; and second, consistency, so that no additional node/edge is included into the answer graph, apart from those matching the samples. Those properties are formally defined as conditions for the subgraph-isomorphism relations between samples and answers [8]. Combining Definition 1 with connectedness and consistency, we obtain the definition of our proposed query-paradigm. Formally, given a set of samples \( S \) on the database \( G = (V, E, \ell) \), the task of multi-exemplar (mExQ) answer search requires to find the set of connected and consistent answers \( A \subseteq G \) such that \( \forall s \in S, s \subseteq A \).

Given a single query, the number of isomorphic graphs that exists within a large knowledge base is usually extreme. Often, the user is interested only in the top-\( k \) answers, for a specific ranking function based on some answer score, where the score of an answer is given by a function \( p: A \rightarrow \mathbb{R}^+ \). In this case, \( x_2Q \) employs a ranking score based on the topical and topological proximity between the query fragments and the answers [8]. To efficiently retrieve the set of answers that match all conditions presented by the user sample(s), \( x_2Q \) implements a filter-and-refine approach. The first step takes as input the graph and detects a set of candidate regions that most probably contain the exemplar answers. The second step searches for answers only in such regions [8], reducing the set of subgraphs to compute to only on those that are more likely to join with the other structures.

3. **Demonstrating \( x_2Q \)**

During the demonstration the attendees will be able to propose their own queries, which will be answered in real time, with no precomputed results. Next, we present example tasks covering the main \( x_2Q \’ s \) features. Assume a curious movie aficionado, who is exploring award winning movies and celebrities. The starting examples could be the entities “Steven Spielberg”, or “Academy Award”.

**1) Example exploration:** Assume a query for Academy Award in the \( x_2Q \’ s \) entity search bar as shown in Figure 4.

![Figure 4: Entities matching “Academy Award”.](image4.png)

**Figure 4:** Entities matching “Academy Award”.

![Figure 5: Suggestions for “Best Director”](image5.png)

**Figure 5:** Suggestions for “Best Director”
Among the various suggestions, the user chooses for instance “Academy Award for Best Director”. After the selection, the system provides a set of candidate expansions (Figure 5), which are interesting relationships with other entities. The user can then select the expansion which best describes their need, such as “Steven Spielberg” as one of the winners of the award. Assume the user selects Spielberg, the query now has two nodes and an edge. Again, after the selection the system offers other possible expansions to proceed with the reformulation process. Additionally, the system lists the results of the exemplar search (Figure 6): in this case other awards winners such as “Clint Eastwood” and the “PGA Lifetime Achievement Award”. These are facts similar to the user query. Such related facts are exemplar query answers [10], which can be selected to enrich the original search.

2) Multi-example search: The user can also perform additional searches for other entities/facts, until all the example fragments of interest are represented (e.g., they can add a person born in a place if this kind of biographical information is of interest). When the selected query is composed of multiple disconnected fragments, the system will automatically treat it as a multiple exemplar-query [8]. Hence, XQ searches the knowledge graph for subgraphs matching all those fragments. For instance, in the example above, after the selection of Clint Eastwood and his award, the example will contain two people who won two different awards (Spielberg and Eastwood), and may lead to answers containing two winners for the same award (Figure 7). A fundamental goal of the demonstration is to showcase the flexibility of the multi-exemplar query paradigm. Users will be challenged to identify examples of some interesting situations (e.g., award winning celebrities that are members of the same family). They will then experience how XQ allows to easily describe complex situations by means of multiple simple examples, providing an effective way for knowledge graph exploration.

3) Comparing ranking functions: The system provides different ranking scores for both the exemplar query answers and the graph suggestions. During the exploration different scoring techniques may prove effective for different purposes. For instance given an actor and a ranking function that favors typical relationship for that type of entity, we will get as a result movies where this actor had a role. Alternatively we could use a ranking function that favors surprising relationships. In this case for example, the system may reveal that Arnold Schwarzenegger, a part from an actor, was also a politicians. During the demonstration, the audience will be able to experiment with different rankings, and see their effect on the system suggestion.

4. CONCLUSIONS

We demonstrated XQ, an interactive and progressive system for exploring large knowledge graphs. The user is guided in the exploration in an example-based approach, in which they first propose a set of entities as partial examples of the intended results, and the system gradually suggests reformulation to the example. XQ incorporates the exemplar query paradigm and algorithms, allowing for a flexible and expressive generation of results. This system is the first that allows for partial queries, and compares multiple ranking scores.

References