X_2Q: Your Personal Example-based Graph Explorer

Matteo Lissandrini[◊] Davide Mottin[§] Yannis Velegrakis[◊] Themis Palpanas[‡]
[◊]University of Trento [§]Hasso Plattner Institute [‡]Paris Descartes University
ml@disi.unitn.eu davide.mottin@hpi.de velgias@disi.unitn.eu themis@mi.parisdescartes.fr

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 X_2Q, a system that facilitates the exploration of knowledge graphs with a hands-on approach. X_2Q 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. X_2Q returns answers in real-time on Freebase, one of the largest available knowledge graphs.

PVLDB Reference Format:
Matteo Lissandrini, Davide Mottin, Yannis Velegrakis and Themis Palpanas. X2Q:Your Personal Example-based Graph Explorer. *PVLDB*, 11 (12): 2026-2029, 2018.
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 language navigation. Such complexity has called for novel exploratory methods [12, 2, 4] and search paradigms [10] to support expert and novice users in retrieving the intended information.

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3236251

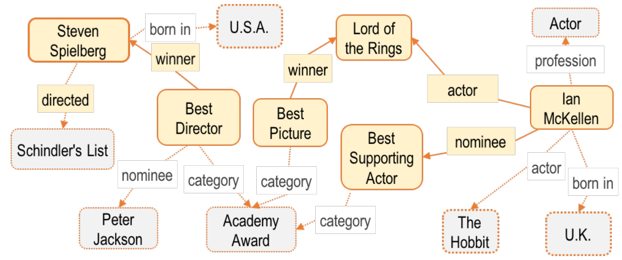

Figure 1: An excerpt from a Knowledge Graph.

Recent exploratory methods [11] 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 retrieves *American movie directors who won some award*, a user should 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 a detailed example requires a fair knowledge of both the schema and the data. Existing exploration systems do not help the user in formulating the proper query, since they are only convenient interfaces to the underlying query system [13, 2, 4]. Neither they help the user in reformulating their queries. Hence, they fail to guide the user towards novel interesting answers.

We propose X_2Q, a fully interactive, autonomous, example-based system for exploring large and complex knowledge graphs. X_2Q 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, X_2Q 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: Our system is the first that studies different, domain-agnostic, ranking functions for graph-query suggestions. 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, nor of a rigid query language.

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 flo(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 well defined. Hence, they may look up 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, $\texttt{X}_2\texttt{Q}$ 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, $\texttt{X}_2\texttt{Q}$ *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*. $\texttt{X}_2\texttt{Q}$ accepts single examples and also examples composed of distinct facts, providing the necessary flexibility for combining partial knowledge into a single answer.

We showcase $\texttt{X}_2\texttt{Q}$, a search-and-explore interactive system to assist (especially novice) users in exploring large knowledge graphs, without the need for a query language. $\texttt{X}_2\texttt{Q}$ 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 $\texttt{X}_2\texttt{Q}$ with Freebase (the most extensive knowledge graphs on which this kind of systems have ever been demonstrated), for which $\texttt{X}_2\texttt{Q}$ 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 $\texttt{X}_2\texttt{Q}$ SYSTEM

The $\texttt{X}_2\texttt{Q}$ system supports two main tasks: (i) example construction and expansion, and (ii) exemplar-query processing.

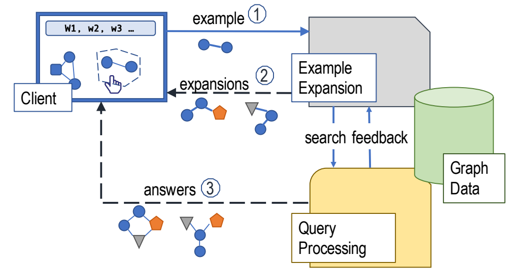

Figure 2: Overview of $\texttt{X}_2\texttt{Q}$ Architecture.

In the first phase (see Figure 2), while the user provides the example, or the examples, of interest, $\texttt{X}_2\texttt{Q}$ 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, $\texttt{X}_2\texttt{Q}$ 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.

A *knowledge graph* is a labeled multi-graph $\mathcal{K}{:}\langle V_\mathcal{K}, E_\mathcal{K}, \ell\rangle$, where $V_\mathcal{K}$ are the entities (also nodes or vertices), $E_\mathcal{K} \subseteq V_\mathcal{K} \times V_\mathcal{K}$ are the relationships among them (i.e., the edges), and ℓ is the labeling function to distinguish various kinds of edges. Moreover we say that $\mathcal{K}' \sqsubseteq \mathcal{K}$ means that $\mathcal{K}'$ is a subgraph of $\mathcal{K}$, i.e., $V_{\mathcal{K}'} \subseteq V_\mathcal{K} \wedge E_{\mathcal{K}'} \subseteq E_\mathcal{K}$.

Search on a knowledge graph is usually performed by means of a graph query $Q{:}\langle V_Q, E_Q, \ell\rangle$, $\texttt{X}_2\texttt{Q}$ processes queries following the exemplar query paradigm. In particular an exemplar graph query is a graph $Q \sqsubseteq \mathcal{K}$ of which the answers are all the similar structures (for some similarity relation $\sim$) that are found in the knowledge graph $\{A \sqsubseteq \mathcal{K} \,|\, A \sim Q\}$ [10, 6]. $\texttt{X}_2\texttt{Q}$ 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_ε, s.t. $Q \sqsubseteq Q_\varepsilon$. The expanded query Q_ε will then contain the same edges of Q, and some additional edges $E_\delta {=} E_{Q_\varepsilon} \setminus E_Q$. Hence, the focus will be on identifying the elements E_δ to add to the user query, it follows that the suggestion algorithm σ has the role of suggesting edge-expansion, i.e., $\sigma{:}\mathcal{P}(E_\mathcal{K}) {\mapsto} \mathcal{P}(E_\mathcal{K})$. 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 $E_\delta {\subset} E_\mathcal{K}$, $m = |E_\delta|$, with $\rho(Q,e){>}0$, $\forall e {\in} E_\delta$ and $\forall \bar{e} {\in} E_\mathcal{K} \setminus E_\delta$, $\exists e \in E_\delta$ such that $\rho(Q,e){>}\rho(Q,\bar{e})$. Where ρ is a relevance function that provides some score 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 $\texttt{X}_2\texttt{Q}$ 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_δ 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. Here, we describe here only their central intuition.

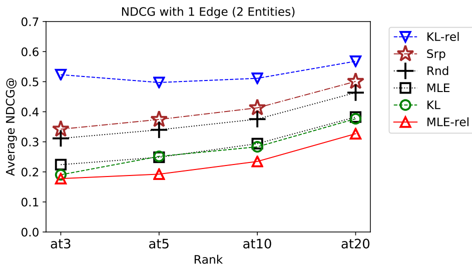

Figure 3: Effectiveness of different scoring in X$_2$Q.

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 is then smoothed $\hat{p}(l|Q) = \left(|E^l_Q| + \epsilon\hat{p}(l|\mathcal{K})\right) / \left(|E_Q| + \epsilon\right)$, where $|E^l_Q|$ 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 favors labels that are frequent around the query, but infrequent in the dataset.

The more advanced ranking techniques 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, in our case, this is the set of graphs that satisfy the original query (before expansion). Hence, we resolve the current graph-query and obtain some set $\mathcal{G}_{rel} = \{G_1, ..., G_k\}$. Once obtained the set $\mathcal{G}_{rel}$ of (pseudo-)relevant graphs, the relevance model is computed through maximum likelihood estimation as $\hat{p}(l|\mathcal{G}_{rel}) \approx \sum_{G \in \mathcal{G}_{rel}} \hat{p}(l|G)\hat{p}(Q|G)$, where $\hat{p}(Q|G) \propto \prod_{l \in Q} \hat{p}(l|G)$, and each $\hat{p}(l|G)$ is computed according to the maximum likelihood estimation above. Additional ranking methods exploit the concept of *surprise* trying to identify those elements that are *unexpectedly* frequent around the user query. The user feedback and interactions are then integrated into the probability $\hat{p}(l|G)$. The evaluation of our approach with 65 real queries from QALD-7 dataset (`qald.sebastianwalter.org`) and real users, shows the effectiveness of our suggestion methods (see Figure 3).

2.2 Answering Multi-Example Queries

The exemplar query Q may be either one single graph or be composed of multiple distinct graphs, each one representing a different aspect of the user need. For this reason, we say that each connected component in the query is a sample so that the query is composed of one or multiple samples. In this regard, the definition of multi-exemplar query, generalizes the concept of exemplar query as follows [8]:

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

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 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.

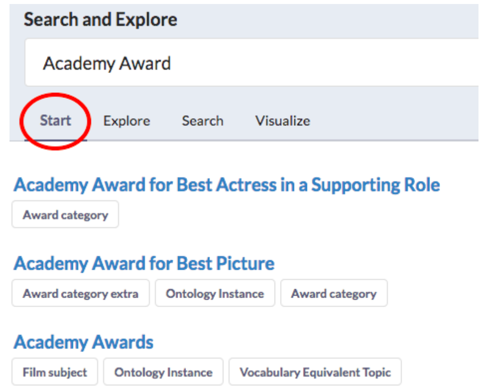

Figure 4: Entities matching "Academy Award".

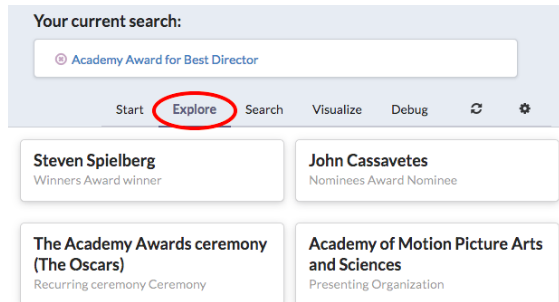

Figure 5: Suggestions for "Best Director"

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 $\mathcal{S}$ on the database $G : \langle V, E, \ell \rangle$, the task of *multi-exemplar* (*mExQ*) *answer search* requires to find the set of *connected* and *consistent* answers $A \subseteq G$ such that $\forall s \in \mathcal{S}, s \sqsubseteq A$.

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 $\rho{:}\mathcal{A}{\mapsto}\mathbb{R}^+$. In this case, X$_2$Q 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$_2$Q 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 those that are more likely to join with the other structures.

3. DEMONSTRATING X$_2$Q

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$_2$Q'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$_2$Q's entity search bar as shown in Figure 4.

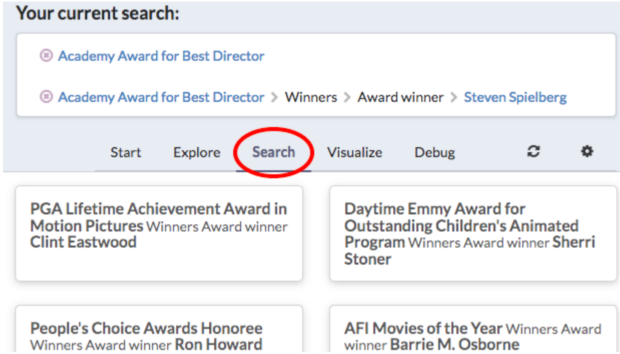

Figure 6: Exemplar Query results (right).

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. 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) such as other awards winners like "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, X$_2$Q 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 X$_2$Q allows to easily describe complex situations employing 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 useful for different purposes. For instance, given an actor and a ranking function that favors typical relationships for actor entities, the result will be movies where this actor had a role. Alternatively, one other ranking function can favor surprising relationships. For example, the system may reveal that Arnold Schwarzenegger was a politician as well as an actor. During the demonstration, the audience will be able to experiment with different rankings and see their effect on the system suggestion.

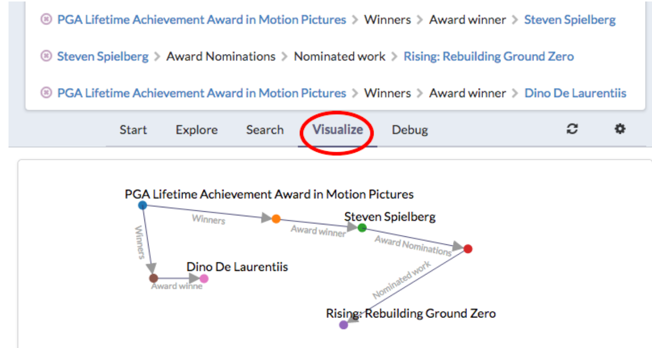

Figure 7: Visualizing the result of a graph search.

Dataset: The system will be running on a cleaned dump of the Freebase knowledge graph (`developers.google.com/freebase`) containing more than 70M entities of 11 thousand different types, and 300M edges with more than 4.3 thousands different relationships.

4. CONCLUSIONS

We demonstrated X$_2$Q, 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. X$_2$Q 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

[1] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR*, pages 243–250, New York, NY, USA, 2008.

[2] A. El-Roby, K. Ammar, A. Aboulnaga, and J. Lin. Sapphire: Querying rdf data made simple. *PVLDB*, 9(13):1481–1484, 2016.

[3] V. Fionda and G. Pirro. Explaining and querying knowledge graphs by relatedness. *PVLDB*, 10(12):1913–1916, 2017.

[4] K. Huang, S. S. Bhowmick, S. Zhou, and B. Choi. Picasso: exploratory search of connected subgraph substructures in graph databases. *PVLDB*, 10(12):1861–1864, 2017.

[5] N. Jayaram, S. Goyal, and C. Li. Viiq: Auto-suggestion enabled visual interface for interactive graph query formulation. *PVLDB*, 8(12):1940–1943, 2015.

[6] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying knowledge graphs by example entity tuples. *TKDE*, 27(10), 2015.

[7] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. *SIGIR Forum*, 51(2):251–259, Aug. 2017.

[8] M. Lissandrini, D. Mottin, Y. Velegrakis, and T. Palpanas. Simple multi-example search in complex information spaces. In *ICDE*, 2018.

[9] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Searching with xq: the exemplar query search engine. In *SIGMOD*, pages 901–904. ACM, 2014.

[10] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: a new way of searching. *VLDB J.*, pages 1–25, 2016.

[11] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. New trends on exploratory methods for data analytics. *PVLDB*, 10(12):1977–1980, 2017.

[12] D. Mottin and E. Müller. Graph exploration: From users to large graphs. In *SIGMOD*, SIGMOD '17, pages 1737–1740, New York, NY, USA, 2017. ACM.

[13] R. Pienta, F. Hohman, A. Tamersoy, A. Endert, S. Navathe, H. Tong, and D. H. Chau. Visual graph query construction and refinement. In *SIGMOD*, pages 1587–1590. ACM, 2017.