Searching with XQ: the eXemplar Query Search Engine

Davide Mottin
University of Trento
Trento, Italy
mottin@disi.unitn.eu

Matteo Lissandrini
University of Trento
Trento, Italy
ml@disi.unitn.eu

Yannis Velegrakis
University of Trento
Trento, Italy
velgias@disi.unitn.eu

Themis Palpanas
Paris Descartes University
Paris, France
themis@mi.parisdescartes.fr

ABSTRACT

We demonstrate *XQ*, a query engine that implements a novel technique for searching relevant information on the web and in various data sources, called *Exemplar Queries*. While the traditional query model expects the user to provide a set of specifications that the elements of interest need to satisfy, XQ expects the user to provide only an element of interest and we infer the desired answer set based on that element. Through the various examples we demonstrate the functionality of the system and its applicability in various cases. At the same time, we highlight the technical challenges for this type of query answering and illustrate the implementation approach we have materialized. The demo is intended for both researchers and practitioners and aims at illustrating the benefits of the adoption of this new form of query answering in practical applications and the further study and advancement of its technical solutions.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*

Keywords

Query paradigms; Exemplar queries; Labeled graphs

1. INTRODUCTION

Highly structured query languages are typically used by expert users that know well what they are looking for and how to express it using the constructs of the query language. Nowadays, a large number of users are technically novice or have no clear idea of the item they are looking for, thus, they are producing simple, vague, and most of the time, ambiguous queries. This has led into a large amount of work on trying to discover what the user had in mind when formulating the query. Techniques such as interactive query relaxation [6], semantic enhancement, or related queries using log-analysis [7], are only some examples among the many.

We demonstrate here a novel query technique we have recently introduced [5], in which the user knows an element among those expected to be in the desired result set, but cannot provide a set of specifications that actually describe all the elements in that desired set. In other words, the user "query" works as an example of what the user is looking for. We call this novel query paradigm *exemplar queries* to emphasize its different nature from those mentioned previously and the new evaluation methods it requires.

Although the idea of exemplar queries looks very similar to the well-known notion of query by example (QBE), it is fundamentally different. In QBE, the user query is used simply to communicate to the query evaluation engine the conditions that the elements in the result set should satisfy. In some sense, QBE works like a wild-card query, and is simply a more user-friendly method to describe the query conditions. In an exemplar query, on the other hand, the information provided by the user is a sample from the desired set, which means that the conditions characterizing all the elements in the desired answer may not even be explicitly stated in the user query.

Exemplar queries may attract considerable attention from many different types of end users. They are particularly suitable for users that need to investigate a topic in a field in which they have none or limited experience, which means that they do not know the terminology to formulate the right search queries, but instead they know of a specific instance (sample) of what they are looking for. For example, a student asked to investigate a new topic may be given a paper on that topic as a starting point. The challenge is that many aspects of the topic may not be explicitly stated in the provided paper, thus, they have somehow to be inferred. A different example is the one of a lawyer or a reporter that may have a specific case in their hands and are looking to find other similar cases.

Exemplar queries may also attract considerable attention from the search engine community since they can significantly advance the field with novel functionalities.

In particular, in parallel to the query evaluation that a search engine performs, the user query can also be seen as an exemplar query and be evaluated as such. The results of this evaluation can be appended to the results the search engine generates, increasing the probability to capture the user's intent. For instance, a query on the World War II will typically return documents related to this war. Evaluating

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.

http://dx.doi.org/10.1145/2588555.2594529.

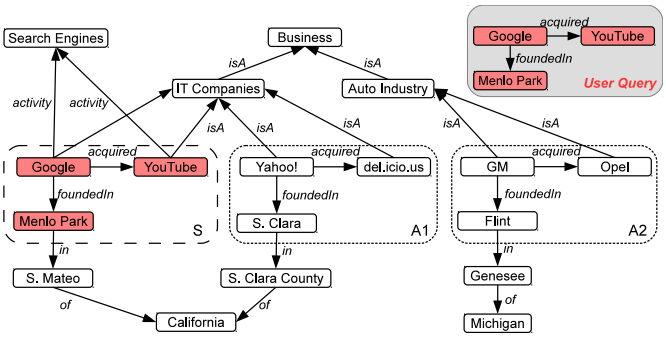

Figure 1: Exemplar Query Example

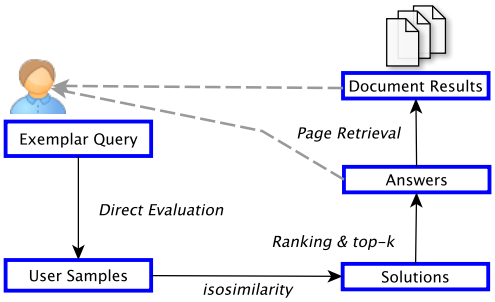

Figure 2: Exemplar Query Evaluation Steps

the query as an exemplar query will result to many other big wars in history. Exemplar queries can also be used to enhance the "related information" sections that existing search engines are currently offering, like for instance, the related entities or the queries other users have also searched that Google provides on the side bar. In the query about World War II, for example, the side bar may contain also link on information about World War I, or the Roman Empire wars.

This demonstration is about XQ, a system that implements this new search paradigm of exemplar queries, and can complement to existing search engines and significantly enrich the current user experience.

2. RUNNING EXAMPLE

The demonstration will not be around a single running example, but the participants will have the opportunity to try many different searches, either proposed by the demonstrators or by the participants themselves. The principal running example illustrating the functionality of the system, however, and the one that will be first presented is the one of the exemplar query of *Google* and *YouTube*. Consider the case of a university student who has been given an assignment to perform a study on company acquisitions in California. The student is not really an expert in the field, and not familiar with the related terminology. Writing a query with the terms "acquisitions" and "California" will in the best case return documents talking about the topic of acquisitions, that are also mentioning California. An article on the takeover of *del.icio.us* by *Yahoo!* may not be returned if the actual words of acquisition and California are not explicitly mentioned in the text. The student knows that a good case of the type of acquisition she is looking for is the one of *YouTube* by *Google*. Thus, she issues the query: *"Google founded in Menlo Park acquired YouTube"*. The search engine typically responds with results related to *Google*, *Menlo Park*, and *YouTube*, but will not return anything related to an acquisition of *del.icio.us* by *Yahoo!*. The same query without the geographic specification place is still valid, but any further specification in the example is useful to restrict the search space. If there is a significant number of users that have performed similar searches in the past, an analysis of the query logs may reveal that information and the search engine may be able to propose, in the related searches section, queries on *Yahoo!* and *del.icio.us*. (A simple test in existing search engines reveals that this is not actually happening.) Relaxing one or more of the query conditions does not help in a significant way, since the results are still focused around the *Google* case. Nevertheless, XQ is able to perform the right reasoning and return the *Yahoo!-del.icio.us* case.

Consider now a second candidate answer for the user query: *Opel* that was acquired by *General Motors* (*GM*). Among the *Yahoo!-del.icio.us* and *GM-Opel*, it is more likely that the former is among the company acquisitions that the user is interested in, and not the latter. This is because even though *Yahoo!* was founded in a different city than *Google*, that city is still in California (just like *Google*), while the city that *GM* was founded is in Michigan. Furthermore, the example of *Google*-YouTube is about IT companies, and so are the *Yahoo!-del.icio.us*, while *GM-Opel* belong to the automotive industry. XQ is able to also discover and return *GM-Opel* as an answer to the exemplar query but will rank it lower than the one of *Yahoo!-del.icio.us*.

Figure 1 illustrates a portion of the data repository with the information described previously, the exemplar query (top left) and the relationships that lead to the inference of the respective exemplar query answers.

Given the case of *Google-YouTube*, our system is able to identify not only that both the cases of *Yahoo!-del.icio.us* and *GM-Opel* are related to the case of *Google-YouTube* but also that the former is more related to it than the latter.

3. TECHNICAL DESCRIPTION

Figure 2 illustrates the individual steps of an exemplar query evaluation task. It is assumed that data is modeled using a graph based data model in which an element of interest is represented as a node and its characteristic attributes as edges. The model is generic enough to capture data in many different application scenarios.

[**Direct Evaluation**] Given the user query, the first step is to identify in the data repository the structure to which the user is referring, i.e., those representing the example that the user already knows to be part of the desired result set. We refer to that structure as the *user sample*. The reason we consider that structure as the example and not what the user has actually provided in query expression is two-fold. The first is that what the user provides may be vague and imprecise since it is typically expressed in some user friendly language such as a set of keywords. The second is that in order to construct the exemplar query answers, we need to find similar elements to the element the user knows about (the one specified in the user query), but to do so we need to know as much information as possible about that element. Unfortunately, most of the time the user query contains only the information necessary to identify the required element and is not intended for providing a full description of the element. Thus, in order to obtain as much information as possible, there is a need to recognize it in the data repository and retrieve all the information that is available for it.

For this task there is already a large volume of literature on methods that can be used. In our system we are based on some semantic-based technique we have recently developed [1], but this is not the focus of our current demonstration, so we will not elaborate on it.

[IsoSimilarity] Once the user sample has been identified, we search in the database elements similar to it. Note that the previous step may generate more than one sample. If this happens, then the current step is repeated for each of the samples identified. Checking for similar elements is a challenging task due to the number of similarity checkings that need to be done, a solution that does not scale well.

To reduce the number of similarity searches that need to take place, XQ uses an intelligent pruning and similarity evaluation technique. The idea is to remove in advance nodes that are unlikely to be of interest to the user. The first intuition is that nodes in the graph that are located far from the user sample will be also semantically distant from the user's intention as expressed in the query. Conversely, the portion of the graph that is likely to contain relevant answers is called *Relevant Neighborhood*, and it is constructed from the subset of nodes with higher proximity, in terms of path length, to the nodes of the user sample. For this reason XQ uses a principled way of measuring the relative distance and for pruning the graph, which builds upon the well known concept of the Personalized PageRank (PPV) [2]. In XQ, user preferences are expressed through the nodes and edges in the exemplar query. Hence, the XQ algorithm does not treat all edges equally as it happens for links between webpages, instead it adapts to the various edges and their labels, and assign weights proportional to the amount of information carried by each edge compared to other edges. Thus the main difference between the original PPV model and the one XQ uses, i.e., the Adaptive Personalized PageRank Vector (APPV), lays on the fact that we build the adjacency matrix $\boldsymbol{A}$ of the database D in order to take into account also the different importance of the edges. The APPV $\boldsymbol{v}$ is then defined as for the PPV and is computed over the weighted matrix $\boldsymbol{A}$ with: $\boldsymbol{v} = (1-c)\boldsymbol{A}\boldsymbol{v} + c\boldsymbol{p}$. In the computation of the vector $\boldsymbol{v}$, $\boldsymbol{p}$ is the vector that represents the starting probability in the PPV algorithm and which is conveniently biased towards the nodes in the user sample. In order to obtain higher performance XQ approximates the computation of the APPV vector by implementing an iterative function that does not need to traverse the entire graph but guesses the value of the PageRank using an approach similar to the *weighted particle filtering procedure* proposed in [4]. The final values contained in the APPV vector $\boldsymbol{v}$ represent an estimate of the distances of the nodes in the graph from the subset of nodes in the user sample. The distance values in $\boldsymbol{v}$ are then seen as a *relevance* measure, and we keep those nodes with a value higher than a threshold.

For similarity, XQ uses a method based on graph isomorphism on edge labels. Sub-graph isomorphism is known to be an NP-complete problem. To improve the performance, XQ uses an effective way to prune the search space even further than what was pruned by the relevance measure, and restricts the list of database nodes that have to be matched to the nodes of the user sample in order to find isomorphic structures. For this, it uses an efficient technique for comparing nodes, and an algorithm for effectively rejecting pairs of nodes that are bound to not participate in any isomorphic mapping. For each node we store a signature precomputing the set of edge-labels of the edges at a fixed distance from the node. The verification process matches the query node signatures with each node signature in the Relevant Neighborhood and prunes the non-matching nodes. The effectiveness of the method is further improved by exploiting the concept of simulation, which is a computationally tractable notion of graph matching. Although this technique may lead to false positives, the schema is effective and reduces further the search space and time. The false positives are subsequently removed by running the traditional isomorphic verification algorithm on them. A more detailed description of this step can be found in the conference version of this work [5].

[Ranking] Once the set of solutions of the user samples have been computed, they need to be ranked according to the likelihood that they are of interest to the user. Finding the right ranking function requires taking into consideration the various factors that may affect such a list. We claim that such a decision depends on two main parameters: the structural similarity and the importance of the label structures. For the first, we have adopted a metric that is based on a vectorial representation of nodes using its neighborhood [3] extended to capture the differences among nodes that emerge when taking into account the edge-labels of the neighbors. For the second, XQ uses the Personalized Page Rank (computed in the previous step) that among others embeds the distance information in that score to take into consideration the distance of a node from the user sample. As the final ranking score we take the linear combination

$$\rho(n_s, n) = \lambda \mathcal{S}(n_s, n) + (1-\lambda)\boldsymbol{p}[n]. \quad (1)$$

where $\boldsymbol{p}[n]$ is the APPV score and $\mathcal{S}(n_s, n)$ is the structural similarity. λ is a diversification parameter that depends on the user and on the data. A value close to 1 favors results similar to the neighbor nodes of the user sample, while a value close to 0 favors solutions close to the sample.

[Page Retrieval] The ranked solutions are elements in the data repository that are related to the user sample. They can be used as already mentioned in the side bar of existing search engines to provide other related information to the user query. For instance, they can form a section (or enhance an existing one) on related queries, or related entities. However, it may be of interest to actually retrieve pages about these elements and merge them into the existing search results of a traditional search engine. For this, an optional step is the one of page creation that retrieves pages from a search engine related to each of the solutions generated in the previous step.

4. SYSTEM DESCRIPTION

The interface of the XQ System looks similar to the interface of known search engines, with the typical search field and the list of results below it (see Figure 3). However, the results are fundamentally different than those search engines typically provide since it implements the exemplar query evaluation task of Figure 2. Apart from the system functionality that can be observed on this main page, the system offers the interested researcher the opportunity to dig into the individual steps and observe the results. For instance, given an exemplar query and the generated result answer, the option "Samples" at the top leads the user into a different panel that illustrates the samples that have been found

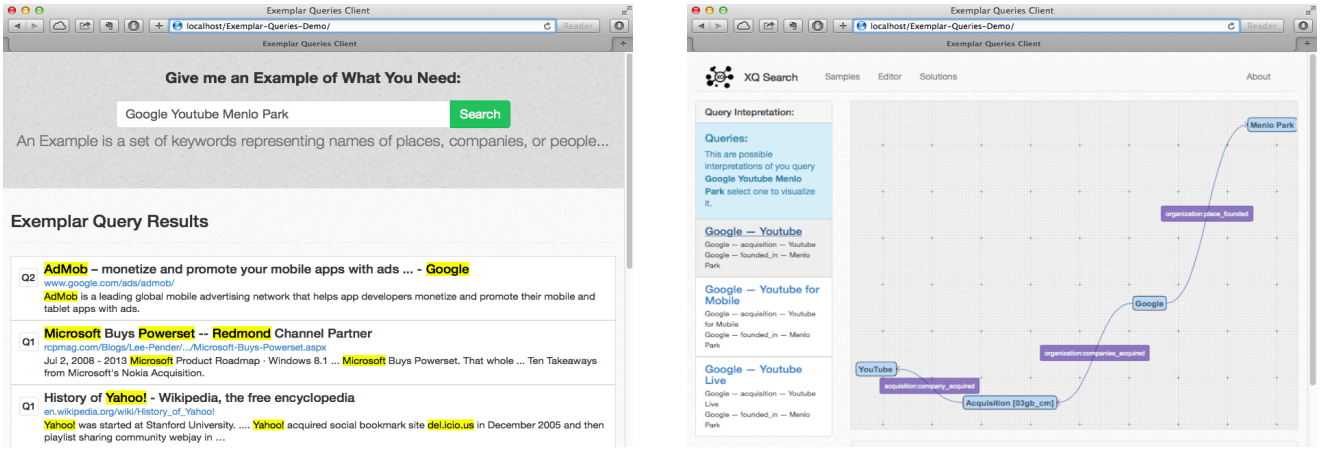

Figure 3: The XQ System Interface: The main panel (left) and the sample panel (right).

for the provided exemplar query. This panel is illustrated on the right-hand side of Figure 3. The left part of this panel is the list of the found samples and by clicking on one of them, the canvas part illustrates its properties and neighbors in the data repository. This can help in understanding why a structure in the repository is considered as a sample.

Selecting a sample from the list and then the option "Solutions" from the top menu-bar, the user can observe all the solutions that the system has generated for the selected sample, i.e., the structures that are considered related to the sample. The list of solutions is ranked according to the believed relevance to the sample. Clicking a solution in the list displays in the canvas its structure, so that a participant can understand why a specific solution has been selected.

Retrieving documents related to each of these solutions generates the set of documents returned as an answer to the exemplar query and are shown on the main page (Figure 3 left-hand side) under the search text field.

5. DEMONSTRATION SCENARIO

The demo will start with a very brief introduction on the idea of exemplar queries followed by a series of exemplar query executions and the explanation of the retrieved results. This will be done mainly on the main screen of the XQ Engine. The first exemplar queries to be executed will be proposed by the demonstrators (with the very first being the one described in Section 2) and will be particular queries that will help the audience in quickly and fully understanding the notion of exemplar queries. Then, the audience will have the opportunity to try their own queries and evaluate the quality of the retrieved results.

As a dataset for the demonstration, we will be using freebase, which is a knowledge base modeled as an RDF graph containing around 53 million nodes and 213 million edges. All the queries tried during the demonstration will be executed at real-time, with no pre-computed results used.

For every query executed in our system, we will be able to show the computation that has taken place in each of the processing steps of the exemplar query evaluation algorithm. This will be done through the panels "Samples" and "Solutions" described in the previous section.

Statistics about execution time and data characteristics will also be available to the demonstration audience to show the challenging performance issues ands solutions. The participant will be able to change the λ parameter of the ranking function and observe the behavior of the algorithms.

In a separate window, we will also send the user queries to be executed by one or more search engines (either semantic or traditional), and look at the results they return alongside other complementary information they provide such as related searches or related entities. We will compare this cumulative information to the results that our system returns as an answer to the exemplar queries, and highlight the differences between the two approaches.

[**Demonstration Goals**] The main goal of the demonstration is to introduce the SIGMOD community to this new form of query answering and highlight its benefits. Through the executed queries the participant will realize the opportunities that this new form of query answering has to offer.

Furthermore, through the direct comparison with existing search engines, it will be shown how hard, or even impossible, is to obtain these results using existing technologies.

At the same time, the current demonstration has a highly educational goal. It aims at raising awareness of the technical challenges that this form of query answering brings on the table, how they have been solved in the current version of the system, initiate interesting discussions and exchange of ideas, and, hopefully, stimulate a number of researchers on working on the topic.

6. REFERENCES

[1] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis. Keyword search over relational databases: a metadata approach. In *SIGMOD*, pages 565–576, 2011.

[2] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279, 2003.

[3] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao. Neighborhood based fast graph search in large networks. In *SIGMOD*, pages 901–912, 2011.

[4] N. Lao and W. W. Cohen. Fast query execution for retrieval models based on path-constrained random walks. In *KDD*, pages 881–888, 2010.

[5] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: Give me an example of what you need. *PVLDB*, 7(5), 2014.

[6] D. Mottin, A. Marascu, S. B. Roy, G. Das, T. Palpanas, and Y. Velegrakis. A probabilistic optimization framework for the empty-answer problem. *PVLDB*, 6(14):1762–1773, 2013.

[7] D. Mottin, T. Palpanas, and Y. Velegrakis. Entity Ranking Using Click-Log Information. *IDA Journal*, 17(5):837–856, 2013.