ICDE 2014
2014 IEEE International Conference on Data Engineering
March 31 - Apr 4, 2014 , Chicago, IL, USA

Reviews For Paper

Track	Semi-Structured and RDF Data
Paper ID	359
Title	Querying Knowledge Graphs by Example Tuples

Masked Reviewer ID: Assigned_Reviewer_1
Review:

Question	
Overall Rating	Accept
Reviewer Confidence	High
Novelty	Median
Technical Depth	Median
Presentation Quality	High
List of Strong Points	1. The problem is well-motivated. Writing queries on knowledge graphs is not easy for an average user. 2. The proposed solution using hidden maximal query graph (MQG) derived from user examples and the performing approximate matching of the MQG with the data graph is well thought out and quite reasonable. 3. Experiments on real data are reasonably convincing.
List of Weak Points	1. There is some uncertainty to the validity/trustworthiness of results based on MTurk. 2. There are some recent references missing. A google search on "Queries by Example" revealed an EDBT 2013 paper with a related title. 3. How would the user know what examples to give to the system and what should the structure/schema of those examples be ?
Detailed Comments	Overall, the paper is a pleasure to read. The problem is somewhat novel and the proposed method while not completely novel is quite reasonable. There is still a gap in the usability aspect of the work. How would a user come up with the examples and what structure those examples would take ? For the system to be truly useful, there needs to be a good story for that. The MQG is obtained from the neighborhood graph which is obtained from the example tuples. It would seem that the size of the neighborhood graph would have an effect on the quality of the MQG. How does the size of the

	neighborhood graph affect the precision and recall of the final query result ?

Masked Reviewer ID: Assigned_Reviewer_2
Review:

Question	
Overall Rating	Reject
Reviewer Confidence	Median
Novelty	Median
Technical Depth	High
Presentation Quality	High
List of Strong Points	- High technical depth. A formal definition of the problem is introduced and algorithms are given for finding the maximal query graph and for efficiently exploring the answer space. - The problem and the solution are clearly presented.
List of Weak Points	- The usability of the proposed approach is questionable. - The experimental evaluation is not convincing.
Detailed Comments	This paper proposes an approach for querying large knowledge graphs by using example tuples. The motivation lies on the fact that non-expert users have no knowledge of the graph schema and cannot form structured queries. Thus, the authors suggest to query the graph by providing examples tuples. For example, one could query the graph with the tuple [Jerry Yang, Yahoo!] to get results such as [Steve Wozniak, Apple Inc.], [Sergey Brin, Google] and [Bill Gates, Microsoft]. To achieve this, the proposed method first tries to "guess" a maximal query graph from the input tuple(s), and then top-k answer graphs are returned that approximately match the query graph. The method involves finding the neighborhood graph containing the query entities, selecting edges by importance, efficiently exploring the query lattice and scoring the answer graphs. An experimental evaluation concerning accuracy and efficiency is conducted on graphs from Freebase and DBPedia. The paper is clearly written. The authors formalize the problem, defining the various concepts involved, they specify the steps for a solution and present the respective algorithms. However, the paper has some important weak points, especially regarding the motivation and usability for this particular approach. Indeed, the authors are right that it cannot be expected that non-expert users formulate structured queries to retrieve information from a knowledge

	graph. However, the proposed form of querying (i.e., by the use of example tuples) is not very intuitive. First, it requires that the user knows some (very similar) example, e.g. that she can specify the founder of one company in order to get the founder of another company. Second, it assumes that the user understands how the algorithm works (i.e., how the system chooses the query graph and the results) in order to choose example tuples that are more likely to yield the desired output. This is not trivial, it does not seem very easy how to come up with good query tuples that would guide the system to guess the user intention and provide desired examples. Consequently, it is also difficult for the user to understand why the particular results come up (instead of others), and moreover there is no specific way to guide the system to get better results. Instead, there are many other approaches for exploring large datasets without knowledge of the schema, which the authors indeed mention in the related work: keyword search, query suggestion/autocompletion, natural language questions, interactive query formulation, faceted browsing, etc. The authors do not provide arguments to support why the proposed approach is more suitable than these methods, and there is no comparison with any such method in the experimental evaluation. Although the experimental evaluation includes results for accuracy, the methodology used is not convincing. In case (A), it is questionable if it makes sense to select the ground truth in the proposed way. In case (B), the answers presented to the Amazon Mechanical Turk users are all from the proposed method, instead of comparing results retrieved from different methods. This can serve to measure the accuracy of the ranking, but it cannot tell if these results are better or more accurate than those another method would return. Moreover, the results regarding execution time also yield concerns. Query processing times are in the order of 10 seconds in many cases. If the user needs to wait so long to get some results, why not use some other method for exploring and navigating the graph? And this does not include the time needed for creating the query graph, which for some queries it is up to 60 seconds. Another observation is that this time varies a lot among queries, but there is no discussion offering some explanation. The method uses some parameters, in particular the path length threshold d and the number of edges m in the maximal query graph. Also, as mentioned, various edge weighting schemes are plausible. However, there is no discussion how these parameters and weights are determined and how much they affect the results. These aspects should be taken into account in the experimental evaluation.

Masked Reviewer ID: Assigned_Reviewer_3

Review:

Question

Overall Rating	Neutral

Reviewer Confidence	High
Novelty	High
Technical Depth	Median
Presentation Quality	Median
List of Strong Points	S1. The idea of extending query-by-example to graphs is very nice. S2. The notions of unimportant edges and scoring functions are interesting. S3. The idea of the query-graph lattice is nice, although its practical use should be better justified.
List of Weak Points	W1. There seem to be inconsistencies between the definition of MQG and the notion of answer graphs (see D2 below). W2. The technical development should be justified; in particular, concepts, entities, and relationships commonly found in a knowledge graph should be considered (see D3, D4 and D5). W3. The experimental evaluation should compare with the approach of [21], by treating keywords as input tuples (see D6).
Detailed Comments	The paper proposes GQBE, an extension of QBE to query knowledge graphs (KGs). Given an input tuple, GQBE extracts a (maximum) query graph MQG from the KB. It then queries the KB by leveraging the MQG and a lattice of query graphs by graph subsumption, to compute a top-k set of answers. The answers are then projected into answer tuples. Scoring functions are studied for this purpose. An experimental study was conducted, using real-life data, compared with [17]. The idea of GQBE is very cute! The development of scoring functions is also interesting. Detailed comments. D1. It is a very cute idea to extend QBE to graphs! D2. The definition of MQG needs to be justified: on one hand, it uses a very loose notion of neighborhood, leaving out certain structural properties of knowledge graph; on the other hand, it requires subgraph isomorphism, a strong notion for graph matching, when defining answer graphs. These are rather inconsistent. That is, even if both the computation of MQG (Theorem 1) and subgraph isomorphism were tractable and even if you could get exact

answers for both, the accuracy of answer tuples found would still be in question.

D3. It seems two rounds of approximation are needed by your approach: first get MQG, and then use a top-k (heuristic) for query graphs. This seems to be an overkill and may incur unnecessary loss of accuracy. Why not to compute top-k query graphs directly by capitalizing on a refined scoring function?

D4. In addition to relevance, one often wants diversity when computing answer graphs in practice, to cover as many different aspects of answers as possible. Please justify your lattice when diversification is considered.

D5. As you are to query knowledge graphs (KGs), you may have to revise your approach by taking into account of entities, concepts and relationships, which are typical KG constructs.

D6. Please experimentally evaluate your approach by comparing its accuracy with, e.g., [21], which considers features of knowledge bases.

[21] J. Pound, I. F. Ilyas, and G. E. Weddell. Expressive and flexible access to web-extracted data: a keyword-based structured query language. SIGMOD 2010.