Comparing entities in RDF graphs

Alina Petrova
supervised by Prof. Ian Horrocks
and Prof. Bernardo Cuenca Grau

Department of Computer Science
University of Oxford

alina.petrova@cs.ox.ac.uk

ABSTRACT

The Semantic Web has fuelled the appearance of numerous open-source knowledge bases. Knowledge bases enable new types of information search, going beyond classical query answering and into the realm of exploratory search, and providing answers to new types of user questions. One such question is how two entities are comparable, i.e., what are similarities and differences between the information known about the two entities. Entity comparison is an important task and a widely used functionality available in many information systems. Yet it is usually domain-specific and depends on a fixed set of aspects to compare. In this paper we propose a formal framework for domain-independent entity comparison that provides similarity and difference *explanations* for input entities. We model explanations as conjunctive queries, we discuss how multiple explanations for an entity pair can be ranked and we provide a polynomial-time algorithm for generating most specific similarity explanations.

1. INTRODUCTION

Information seeking is a complex task which can be accomplished following different types of search behaviour. Classical information retrieval focuses on the query-response search paradigm, in which a user asks for entities similar to the input keywords or fitting the formal input constraint. Yet there exists a broad area of exploratory search that is characterized by open-ended, browsing behaviour [18] and that is much less well studied. Exploratory search encompasses activities like information discovery, aggregation and interpretation, as well as *comparison* [13].

Comparing entities, or rather, information available about the entities, is an important task and in fact a widely-used functionality implemented in many tools and resources. On the one hand, systems that highlight similarities between entities can focus on *how much* entities are alike, giving a similarity score to a pair (or a group) of entities [6]. On the other hand, systems can focus on *how* or *why*, in which aspects two entities are similar and different, by comparing entities and finding similar features. Such comparison is done in many domains and for various types of entities: hotels,[1] cars,[2] universities,[3] shopping items,[4] to name a few. However, as a rule, such systems perform a side-by-side comparison of items in a *domain-specific* manner, i.e., following a fixed, hard-coded template of aspects to compare (e.g., in case of hotels, it could be price, location, included breakfast, rating etc.). In a few more advanced systems, similarities are computed with respect to the type of information available about the input entities rather than following a rigid pattern. One such example is the Facebook Friendship pages.[5] Given two Facebook users, a friendship page contains all their shared information, be it public posts, photos, likes or mutual friends, as well as their relationship, if any (e.g., married, friends etc.). However, as in the aforementioned examples, comparison is done over a limited set of attributes.

Relying on a fixed set of aspects is a reasonable solution for tabular data with rigid and stable structure. On the other hand, a more flexible approach to entity comparison is needed for Linked Data, namely for loosely structured RDF graphs. However, all current systems with such functionality compare items following a predefined, domain-specific list of values to compare. Thus, an interesting research problem would be to create a framework for entity comparison that is domain- and attribute-independent.

The Semantic Web has fuelled the appearance of numerous open-source knowledge bases (KBs). Such KBs enable both automatic information processing tasks and manual search, and they facilitate new types of information search, going beyond classical query answering and providing answers to new types of user questions. For example, using KBs one can answer questions like *how are the two entities similar* or *what differs them*, i.e., perform entity comparison.

In this paper we propose to study such questions posed over one of the most common types of KBs — RDF graphs. In particular, we provide a formal framework for posing such questions and we model answers to these questions as similarity and difference *explanations*. We then discuss how

Proceedings of the VLDB 2017 PhD Workshop, August 28, 2017. Munich, Germany.

[1]`www.flightnetwork.com/pages/hotel-comparison-tool/`
[2]`http://www.cars.com/go/compare/modelCompare.jsp`
[3]`http://colleges.startclass.com/`
[4]`http://www.argos.co.uk/static/Home.html`
[5]Original announcement (cashed by the Wayback Machine): `https://web.archive.org/web/20101030105622/http://blog.facebook.com/blog.php?post=443390892130`

multiple explanations to a question can be ranked and we provide a polynomial-time algorithm for generating most specific similarity explanations. Finally, we outline directions of future research.

2. PRELIMINARIES

In what follows we use the standard notions of conjunctive queries (CQs), query subsumption and homomorphism. We disallow trivial CQs of the form $\top(X)$. We model RDF graphs as finite sets of triples, where a triple is of the form $p(s,o)$, p and s being URIs and o being a URI or a literal. Furthermore, we use the notion of a *direct product* of two graphs, adapted to RDF graphs:

Definition 1. Let I and J be RDF graphs, $t_1 = R(s_1, o_1)$ and $t_2 = R(s_2, o_2)$ be two triples. The *direct product* of t_1 and t_2, denoted as $t_1 \otimes t_2$, is the triple $R(\langle s_1, s_2\rangle, \langle o_1, o_2\rangle)$. The *direct product* $I \otimes J$ of I and J is the instance:

$$\{t_1 \otimes t_2 \mid t_1 \in I \text{ and } t_2 \in J\}.$$

3. COMPARISON FRAMEWORK

There are multiple ways of how we can define similarity and difference explanations and how we can model entity comparison. In our framework the formalism of choice is conjunctive queries (CQs). We model formal explanations as conjunctive queries and we consider the problem of finding such explanations as an instance of the query reverse engineering problem.

3.1 Similarity explanations

We would like similarity explanations to highlight common patterns for input entities. Thus, we model them as queries that return both of these entities, i.e, they match patterns fitting both entities. Let $\langle I, a, b\rangle$ be a tuple consisting of an RDF graph I and two URIs from the domain of I $a, b \in dom(I)$ representing input entities. Furthermore, given a query Q, let $Q(I)$ be the answer set returned by Q over I.

Definition 2. Given $\langle I, a, b\rangle$, a *similarity explanation* for a and b is a unary connected conjunctive query Q_{sim} such that $\{a, b\} \subseteq Q_{sim}(I)$.

Example 1. Given two entities Marilyn_Monroe and Elizabeth_Taylor and the Yago RDF graph [14], a possible similarity explanation is:

$$\begin{aligned}Q_{sim}(X) = \ & \mathsf{hasWonPrize}(X, \mathsf{Golden\_Globe}),\\ & \mathsf{diedIn}(X, \mathsf{Los\_Angeles}),\\ & \mathsf{hasGender}(X, \mathsf{female}),\\ & \mathsf{actedIn}(X, Y1),\\ & \mathsf{isLocatedIn}(Y1, \mathsf{United\_States}),\\ & \mathsf{isMarriedTo}(X, Y2),\\ & \mathsf{hasGender}(Y2, \mathsf{male}),\\ & \mathsf{hasWonPrize}(Y2, \mathsf{Tony\_Award}), \textit{etc.}\end{aligned}$$

Q_{sim} can be interpreted the following way: both Monroe and Taylor received a Golden Globes award, died in Los Angeles, acted in movies that were shot in the US and were married to men who received a Tony Award.

Using this definition, we can formulate the following decision problem: given $\langle I, a, b\rangle$, $SimExp$ is a problem of whether there exists a CQ Q such that $\{a, b\} \subseteq Q(I)$. The corresponding functional problem is to compute a query Q such that $\{a, b\} \subseteq Q(I)$, given $\langle I, a, b\rangle$. Note that both the definition of Q_{sim} and $SimExp$ can be easily generalized from a pair of entities to a set of input entities.

We specifically chose the condition to be $\{a, b\} \subseteq Q(I)$ for two reasons. Firstly, the form of Q does not depend on the rest of the data: it does not matter whether there exist other entities that match the graph pattern described by the query; moreover, queries fitting the subsumption condition will not be affected if new data is added. This is very important in the context of *RDF* graphs, since web data is intrinsically incomplete.

Secondly, it is known that the definability problem is coNExpTime-complete for conjunctive queries [3,15]. On the other hand, $SimExp$ can easily be shown to be in PTime: for conjunctive queries, it is sufficient to take the full join of all tables in the database instance.

Let $\mathbf{Sim}(a, b)$ be the set of all similarity explanations for a given $\langle I, a, b\rangle$. Obviously $\mathbf{Sim}(a, b)$ can be quite big, containing numerous explanations, however, we are interested in the most informative ones. Our assumption is that the more specific a similarity explanation is, the better.

Definition 3. Given $\langle I, a, b\rangle$, a *most specific similarity explanation* is a similarity explanation Q^{msp}_{sim} s.t. for all similarity explanations Q'_{sim} wrt $\langle I, a, b\rangle$: $Q^{msp}_{sim} \subseteq Q'_{sim}$.

The decision problem $SimExp^{msp}$ is the problem of deciding whether Q_{sim} is a most specific similarity explanation for the given $\langle I, a, b\rangle$.The related functional problem is to compute a most specific Q_{sim}.

The subsumption relation divides the set of all similarity explanations $\mathbf{Sim}(a, b)$ into $\subseteq$-equivalent classes. If $\mathbf{Sim}(a, b)$ is not empty, then $\mathbf{Sim}(a, b)^{msp}$ is not empty, and there exists a finite most specific similarity explanation Q^{msp}_{sim}, whose size is bounded by the size of I. This explanation can in fact be constructed in PTime (see Section 4).

3.2 Difference explanations

Analogous to similarity explanations, we model difference explanations as CQs, but this time we require only one of the input entities to be in the answer set.

Definition 4. Given $\langle I, a, b\rangle$, a *difference explanation* for a wrt b is a unary connected conjunctive query Q^a_{dif} such that $a \in Q^a_{dif}(I)$, but $b \notin Q^a_{dif}(I)$.

The notion of a difference explanation can be generalized to sets of entities: given an RDF graph I, a set of entities Pos and a set of entities Neg, a *difference explanation* for I and Pos wrt Neg is a unary connected CQ Q^{Pos}_{dif} s.t. $\forall p \in Pos$: $p \in Q^{Pos}_{dif}$ and $Neg \cap Q^{Pos}_{dif} = \emptyset$.

Given $\langle I, a, b\rangle$, $DifExp$ is the problem of deciding whether there exists a difference explanation Q^a_{dif}. The generalized difference explanation problem $DifExp$ can be solved using the most specific similarity explanation problem $SimExp^{msp}$: given I, Pos and Neg, first construct a most specific similarity explanation Q for entities in Pos (done in PTime), and then check whether none of the elements of Neg are in the answer set of Q (conjunctive query evaluation is NP-COMPLETE). Hence, the complexity of generalized $DifExp$ is NP-COMPLETE.

Furthermore, we would like to introduce another definition of a difference explanation that is dependent on the similarities between a and b. We would like the difference explanation for a to be as relevant as possible, hence we model it to be dependent not only on the information about b, but also on the common patterns for a and b. One possible way to do so is the following: let $const(Q)$ be the set of constants appearing in a query Q and let $const(R(\bar{x}))$ be the set of constants appearing in an atom $R(\bar{x})$.

Definition 5. Given $\langle I, a, b\rangle$, a *difference explanation* for a wrt b and Q_{sim} is a different explanation $Q^{a,sim}_{dif}$ such that $\forall R(\bar{x}) \in Q^{a,sim}_{dif}$: $const(R(\bar{x})) \cap const(Q_{sim}) \neq \emptyset$.

Example 2. Let the input entities be $a = \mathsf{John\_Travolta}$ and $b = \mathsf{Quentin\_Tarantino}$. Let Q_{sim} for a and b be an explanation that both persons starred in Pulp_Fiction: $Q_{sim}(X) = \mathsf{starredIn}(X, \mathsf{Pulp\_Fiction})$. Relevant difference explanations could be that Travolta also starred in Grease and other movies, while Tarantino has directed several movies, including Pulp Fiction: $Q^{a}_{dif}(X) = \mathsf{starredIn}(X, \mathsf{Grease})$ and $Q^{b}_{dif}(X) = \mathsf{directed}(X, \mathsf{Pulp\_Fiction})$. On the other hand, an explanation that Travolta (unlike Tarantino) is married to Kelly Preston is rather irrelevant, since we have not compared the two persons with respect to their marital status.

4. TECHNICAL RESULTS

4.1 Algorithm for computing a most specific similarity explanation

We compute a most specific similarity explanation by constructing the *direct product* of the RDF graph, similar to the construction of the direct product of a database instances with itself [15]. Any RDF graph I $a \in dom(I)$ can be associated with a *canonical* unary conjunctive query $q_I(x_a)$ such that for each fact $R(c, d)$ in I there is an atom $R(x_c, x_d)$ in q_I, where x_c and x_d are variables and x_a is a free variable. Note that a is an answer to $q_I(x_a)$ over I. The following algorithm produces a most specific similarity explanation. In it, we first produce an instance with the domain from $dom(I)^2$, i.e., tuples $\langle c, d\rangle$ for $c, d \in dom(I)$, and then construct a canonical conjunctive query of this instance.

Claim 1. If $J \neq \emptyset$, then J is a maximal connected component of $I \otimes I$ such that $a \otimes b = \langle a, b\rangle \in dom(J)$.

Proof sketch: Firstly, if $J \neq \emptyset$, then $\langle a, b\rangle \in dom(J)$, by Step 1. Secondly, the while-loop on Step 5 is in fact the greedy procedure that generates the maximal connected component in $I \otimes I$. Indeed, the condition $R(c, e), R(d, f) \in I$ ensures that the fact $R(\langle c, d\rangle, \langle e, f\rangle)$ is in $I \otimes I$, and the condition that there must exist a fact in J that contains $\langle c, d\rangle$ or $\langle e, f\rangle$ ensures connectedness.

Claim 2. Let $\langle I, a, b\rangle$ be an input of Algorithm 1. Let $q_J(x_{\langle a,b\rangle})$ be the output, and J the instance obtained after the while loop on Step 5. Then all of the following hold.

(i) $\{a, b\} \subseteq q_J(I)$,

Algorithm 1: Algorithm for computing a most specific similarity explanation

```
   Input: an RDF graph $I$, entities $a, b$ from $dom(I)$.
   Output: a most specific similarity explanation for $a$ and $b$.
1  Let $J = \{R(\langle a,b\rangle, \langle c,d\rangle) \mid R(a,c), R(b,d) \in I\} \cup \{R(\langle c,d\rangle, \langle a,b\rangle) \mid R(c,a), R(d,b) \in I\}$;
2  if $J = \emptyset$ then
3      return empty query;
4  Let $J^* = \emptyset$;
5  while $J \neq J^*$ do
6      $J^* := J$;
7      $J := J \cup \{R(\langle c,d\rangle, \langle e,f\rangle) \notin J \mid R(c,e), R(d,f) \in I$, and $\exists$ a fact in $J$ that contains $\langle c,d\rangle$ or $\langle e,f\rangle\}$;
8  Construct $q_J(x_{\langle a,b\rangle})$;
9  foreach $x_{\langle c,c\rangle}$ in $q_J$, $c \notin \{a,b\}$ do
       // Replace $x_{\langle c,c\rangle}$ with constant $c$
10     $q_J(x_{\langle a,b\rangle}) := q_J(x_{\langle a,b\rangle})[x_{\langle c,c\rangle} \to c]$;
11 return $q_J(x_{\langle a,b\rangle})$.
```

(ii) For a connected unary conjunctive query $q'(x)$, if there exist homomorphisms $h_1, h_2 : q' \to I$ such that $h_1(x) = a$ and $h_2(x) = b$, then there exists a homomorphism $h : q' \to J$ such that $h(x) = \langle a, b\rangle$.

Corollary 1. Algorithm 1 produces a most specific similarity explanation.

4.2 Properties of the resulting query

The algorithm 1 runs in time polynomial to the size of the input RDF graph, and the size of resulting most specific similarity explanation is also polynomial to I. It should be noted that the output query tends to be non-minimal. For example, since Marilyn_Monroe and Elizabeth_Taylor acted in several movies that were shot in the US, $Q(X)$ will contain atoms like:

$\mathsf{actedIn}(X, Y1), \mathsf{isLocatedIn}(Y1, \mathsf{United\_States})$,
$\mathsf{actedIn}(X, Y2), \mathsf{isLocatedIn}(Y2, \mathsf{United\_States})$,
$\mathsf{actedIn}(X, Y3), \mathsf{isLocatedIn}(Y3, \mathsf{United\_States})$, etc.

To avoid such redundancy, we can take the *core* of the query (i.e., apply the query minimization algorithm). Taking the core is an NP-COMPLETE problem [9, 11], hence, obtaining a most specific similarity explanation without redundant atoms is an NP-COMPLETE task.

5. RELATED WORK

So far only few works have studied *explanations* over RDF graphs [5, 10, 12], and there is no single formal definition of an explanation over RDF data. A lot of attention has been paid to discovering connections ("associations") between nodes [12], which boils down to finding and grouping together paths in the graph that connect one input node to another one. Such connectedness explanations are orthogonal (rather than alternative) to the similarity explanations modelled as queries, which we propose to study. The two types of explanations are intended to capture different relations between nodes: the former explore possible paths that link the two nodes together, while the latter seek to find commonalities in the neighbourhoods of the input nodes.

The problem of *reverse engineering a query* given some examples originated in late 1970s and was first introduced for the domain of relational databases [20]. Later it was extensively researched with respect to different query formats: regular languages [1], XML queries [7], relational database queries [16,17,19], graph database queries [4] and SPARQL queries [2]. The problem of QRE for RDF data was first studied by Arenas et al. [2] and was implemented by Diaz et al. [8]. In [2], the authors consider three different variations of QRE problem: the basic variation that requires the input mappings to be part of the answer set ($\Omega \subseteq [\![Q]\!]_G$); the one that allows positive examples Ω together with negative examples $\bar{\Omega}$ (such that $\Omega \subseteq [\![Q]\!]_G$ and $\bar{\Omega} \cap [\![Q]\!]_G = \emptyset$); and the variation that requires the examples from Ω to be *exactly* the answer set of Q ($\Omega = [\![Q]\!]_G$). The complexity of these three variations is then provided for fragments of SPARQL with AND, FILTER and OPT.

6. FUTURE WORK

As part of my PhD, I would like to continue studying the problem of entity comparison using RDF graphs in several research directions. So far we have investigated similarity and difference explanations, and we rank the former according to the preference condition based on subsumption. In particular, we assume that the highest ranked explanations are most specific similarity explanations. On the one hand, we would like to apply a similar rationale to difference explanations and to study most general difference explanations as most preferred ones. On the other hand, these may not be the optimal choices for a given user, hence we need to investigate other possible ranking conditions as well as means of user-specific ranking of explanations.

RDF graphs are inherently incomplete, hence it would be useful to consider a scenario where an explanation is produced over an RDF graph and a domain ontology that contains knowledge not explicitly present in the graph. Consider a graph G consisting of two facts: $Teacher(Bob)$ and $teaches(Alice, CS)$, — and a simple EL ontology O consisting of one axiom: $\exists teaches.Class \sqsubseteq Teacher$. Let the two input entities be $Alice$ and Bob. Then a similarity explanation wrt G, O could be $Q(X) = Teacher(X)$, while we are unable to generate such a CQ using only graph data.

In our framework explanations are modelled as CQs, and while CQs are formulas with relatively high readability for a user, it is of interest to be able to verbalize explanations, transforming them into natural language sentences. For example, a formal explanation $Q(X) = \mathsf{livesIn}(X, \mathsf{London}), \mathsf{friendsWith}(X, Y), \mathsf{worksAt}(Y, \mathsf{Oracle})$ could be transformed into an English sentence *"Both input entities live in London and are friends with someone who works at Oracle"*.

While CQs correspond to a large part of queries issued over relational databases, i.e., they have relatively high expressivity, they cannot express things like negation or disjunction, which is a limitation. Hence, an interesting problem would be to consider more expressive languages, in particular, union of CQs and CQs with inequalities and numeric comparison.

Lastly, we are planning to implement a comprehensive comparison system that would compute most specific similarity explanations and most general difference explanations, to test it on real-world RDF graphs and to perform usability tests.

7. REFERENCES

[1] D. Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[2] M. Arenas, G. I. Diaz, and E. V. Kostylev. Reverse engineering SPARQL queries. In *Proc. of the 25th Int. Conf. on World Wide Web*, pages 239–249, 2016.

[3] P. Barcelo and M. Romero. The complexity of reverse engineering problems for conjunctive queries. In *Proc. of 20th Int. Conf. on Database Theory*, 2017.

[4] A. Bonifati, R. Ciucanu, and A. Lemay. Learning path queries on graph databases. In *18th Int. Conf. on Extending Database Technology (EDBT)*, 2015.

[5] G. Cheng, Y. Zhang, and Y. Qu. Explass: exploring associations between entities via top-k ontological patterns and facets. In *International Semantic Web Conference*, pages 422–437. Springer, 2014.

[6] S.-S. Choi, S.-H. Cha, and C. C. Tappert. A survey of binary similarity and distance measures. *J. Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.

[7] S. Cohen and Y. Y. Weiss. Learning tree patterns from example graphs. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 31, 2015.

[8] G. Diaz, M. Arenas, and M. Benedikt. SPARQLByE: Querying RDF data by example. *Proceedings of the VLDB Endowment*, 9(13), 2016.

[9] G. Gottlob and A. Nash. Efficient core computation in data exchange. *Journal of the ACM*, 55(2):9, 2008.

[10] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann, and T. Stegemann. RelFinder: Revealing relationships in RDF knowledge bases. In *Int. Conf. on Semantic and Digital Media Technologies*, pages 182–187, 2009.

[11] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109(1):117–126, 1992.

[12] J. Lehmann, J. Schüppel, and S. Auer. Discovering unknown connections - the DBpedia relationship finder. *CSSW*, 113:99–110, 2007.

[13] G. Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.

[14] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217, 2008.

[15] B. ten Cate and V. Dalmau. The Product Homomorphism Problem and Applications. In *18th International Conference on Database Theory (ICDT 2015)*, pages 161–176, 2015.

[16] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *Proc. of the 2009 ACM SIGMOD Int. Conf. on Management of data*, pages 535–548, 2009.

[17] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query reverse engineering. *The VLDB Journal*, 23(5):721–746, 2014.

[18] R. W. White and R. A. Roth. Exploratory search: beyond the query-response paradigm, 2009.

[19] M. Zhang, H. Elmeleegy, C. M. Procopiuc, and D. Srivastava. Reverse engineering complex join queries. In *Proc. of the 2013 ACM SIGMOD Int. Conf. on Management of Data*, pages 809–820, 2013.

[20] M. M. Zloof. Query-by-example: A data base language. *IBM systems Journal*, 16(4):324–343, 1977.