Explaining and Querying Knowledge Graphs by Relatedness

Valeria Fionda
University of Calabria
Via Pietro Bucci 30/B
Rend (CS), Italy
fionda@mat.unical.it

Giuseppe Pirrò
ICAR-CNR
Via Pietro Bucci 41/C
Rend (CS), Italy
pirro@icar.cnr.it

ABSTRACT

We demonstrate RECAP, a tool that *explains relatedness* between entities in Knowledge Graphs (KGs) and implements a *query by relatedness* paradigm that allows to retrieve entities related to those in input. One of the peculiarities of RECAP is that it does not require any data preprocessing and can combine knowledge from multiple KGs. The underlying algorithmic techniques are reduced to the execution of SPARQL queries plus some local refinement. This makes the tool readily available on a large variety of KGs accessible via SPARQL endpoints. To show the general applicability of the tool, we will cover a set of use cases drawn from a variety of knowledge domains (e.g., biology, movies, co-authorship networks) and report on the concrete usage of RECAP in the SENSE4US FP7 project. We will underline the technical aspects of the system and give details on its implementation. The target audience of the demo includes both researchers and practitioners and aims at reporting on the benefits of RECAP in practical knowledge discovery applications.

1. INTRODUCTION

There is an increasing amount of structured knowledge available on the Web. Projects like Linked Open Data (LOD) aim at creating a global infrastructure for the publishing and interlinking of knowledge on the Web. One peculiarity of LOD is the usage of W3C standards like RDF and SPARQL for the representation and querying of structured data, respectively. On the other hand, also major search engines are now backed up by structured knowledge in the form of Knowledge Graphs (KGs). The Google's KG and Microsoft Satori are just a few examples. While sharing many commonalities from the data representation point of view (both use triples), the two approaches are radically different in terms of query capabilities. LOD datasets can be queried via SPARQL, an expressive query language, while Google's KG and the like are typically accessible only via keywords. In the first case, users require knowledge of the SPARQL query language while in the second case, the lack of a query language makes it difficult to precisely express what one is looking for. One peculiarity of KGs underlying Google and other search engines is that they also *suggest* entities related to the entity in input. As an example, when searching for for D. Knuth, Google suggests L. Lamport as a related entity. Nevertheless, neither an explanation is provided about *why* a pair of entities is related nor any support is given in terms of capabilities to *infer a query* that can be used to capture further related results. This feature is especially useful when the user knows a pair of entities in the results but has no clear idea (or expertise) about how to write a query that captures all the results. To make easier the availability of structured knowledge even to novice users, the research community came up with different techniques. Examples are: *exemplar queries* [6, 7, 8] based on the idea that it is enough for users to provide even a single exemplar result to automatically infer the desired answer set, *query learning* approaches [2] that feature interactive join specification supports with minimal interactions, and reverse engineering of queries [3] where the goal is to learn queries given a set of tuple in the results. Other approaches [4, 9] specifically focus on providing relatedness explanations.

In this demo we demonstrate a novel approach that *combine relatedness explanations and querying capabilities* called RECAP [11, 12]. The user inputs a pair of entities and the system produces different types of explanations that capture the essence of their relatedness according to different aspects (e.g., path diversity, informativeness). The usage of multiple KGs offers a more comprehensive relatedness perspective as compared to the single KG case (see e.g., Fig. 2). Explanations form the basis for the automatic construction of queries that can be used to retrieve entities related to those in input via a *query by relatedness paradigm*. One peculiarity of RECAP is the possibility to work with any existing KG without data preprocessing. Because of this design choice, the algorithmic techniques are reduced to the execution of a set of queries plus some local refinement. We will focus on an implementation of RECAP using the SPARQL query language for RDF data, which makes the tool readily available on a variety of KGs accessible via SPARQL endpoints.

RECAP may attract attention from a broad type of audience. This includes people that are aware of some kind of relationship between entities but need to have a more precise account of *why* these are related while at the same time having the possibility to discover new entities that share a similar relatedness perspective. Returning to our previous example, with RECAP one can discover that one of the per-

Proceedings of the VLDB Endowment, Vol. 10, No. 12

spectives that relate D. Knuth and L. Lamport is that both won both the IEEE J. von Neumann medal and the Turing Award. RECAP's query by explanation approach allows to discover that also T. Hoare and K. Nygaard and J. Hopcroft and M. Stonebraker share the same relatedness perspective. The usefulness of RECAP is also witnessed by the fact that it is currently used in the SENSE4US FP7 project[1], which aims at creating a toolkit to support information gathering, analysis and policy modeling. Here, relatedness explanations provided by RECAP are useful to investigate and show to the user topic connectivity[2], thus enabling to find out previously unknown relevant information, understand how it is of relevance, and navigate it.

This demonstration is about showing the functionalities of RECAP both from the relatedness explanation and query by relatedness perspective with particular emphasis on how RECAP can be concretely used in knowledge discovery tasks.

2. SYSTEM DESCRIPTION

RECAP has been implemented in Java. It leverages the Jena[3] framework to handle RDF data and JavaFX[4] for the GUI, which allows the tool to be accessible across different platforms. The architecture of RECAP is shown in Fig. 2. In this demo we focus on KGs encoded in the RDF data format and accessed via a SPARQL endpoint. Nevertheless, our approach is generic enough to capture data in other graph data formats. Working with SPARQL on top of RDF KGs makes it flexible the access and combination of data from different knowledge domains. In our context, an entity of interest is a node of the graph identified by a IRI.

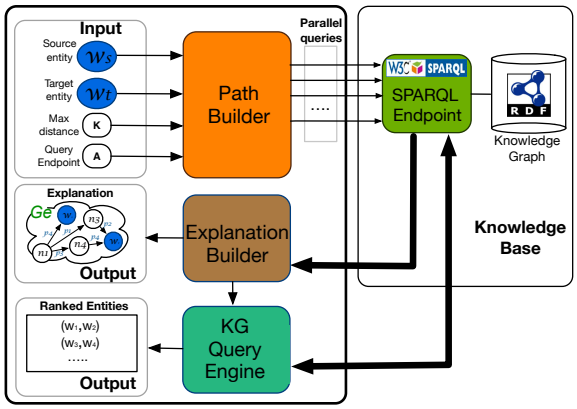

Figure 1: The RECAP architecture.

[**Path Builder**] A user information request is expressed as a pair of entities plus a maximal distance. To help users in precisely identifying the entities in one or more KGs, the system loads a short structured description of the entities (see Fig. 3). The first step of the evaluation of an information need consists in retrieving paths interlinking the specified entities by issuing a set of (automatically constructed) SPARQL queries to one or more endpoints. Parallelizing this aspect via multi-threading guarantees reasonable running times (see Pirrò [11] for further details about the performance evaluation of this approach).

[**Explanation Builder**] Paths interlinking entities form the basis to generate different kinds of query explanations. One immediate type of explanation is the merge of all retrieved paths. However, this kind of explanation becomes quickly unreadable as the number of intermediate entities involved can be very large. To cope with this issue, RECAP allows to flexibly chose the set of paths that will form an explanation. At the core of this idea there are different types of path ranking strategies. In the demo we will focus on the following ranking strategies [11]:

- *Informativeness*: it is estimated by investigating RDF predicates in a path via the notion of *Predicate Frequency Inverse Triple Frequency* [10].
- *Pattern informativeness*: a path pattern generalizes a path by replacing nodes with variables. Pattern informativeness is computed by counting the number of paths sharing a certain path pattern.
- *Diversity*: it takes into account the variety of predicates in a set of paths; diversity guarantees to rank high paths that contain rare predicates.

The user can use one or more path ranking strategies to build an explanation thus controlling the amount of information displayed. Table 1 gives an overview of the types of explanations available in RECAP.

Table 1: Types of explanations.

	Meaning
$E^{\cup}$	Merge all of paths without any pruning
E^{π}_{m}	Merge the *top-m* most informative paths
$E^{\overline{\pi}}_{m}$	Merge paths belonging to the *top-m* most informative path *patterns*
E^{δ}	Merge pairs of paths whose value of diversity falls in $[max, (max - r)]$ where max is the max diversity value over all pairs of paths and r is a % value.
$E^{\pi,\delta}$	Merge the results of E^{π}_{m} and E^{δ}
$E^{\overline{\pi},\delta}$	Merge the results of $E^{\overline{\pi}}_{m}$ and E^{δ}
$\mathcal{P}$	Set of all paths (without merge)

[**KG Query Engine**] Explanations capture the essence of relatedness between a pair of entities. RECAP leverages explanations to learn SPARQL queries that can be used to find other pairs of entities similar to those in input. The tool considers three different approaches for learning SPARQL queries: (i) edge-isomorphic, where nodes in an explanation are replaced by variables while keeping the same join structure; (ii) onto-relaxed queries obtained from edge-isomorphic queries by substituting a subset of predicates with more general ones (taken from the ontology); (iii) edge-relaxed queries obtained from edge-isomorphic queries by substituting a subset of predicates with variables. Predicates to be relaxed can be either indicated by the user or automatically selected on the basis of predicates' informativeness. The last component of the query engine involves the ranking of results. RECAP at the moment implements two different strategies; one based on Pagerank and the other based on the Katz index [5] introduced to estimate the relatedness of actors in a social network.

[1] `http://www.sense4us.eu`
[2] A module of the SENSE4US toolkit extracts topics from policy documents.
[3] `http://jena.apache.org`
[4] `http://docs.oracle.com/javafx`

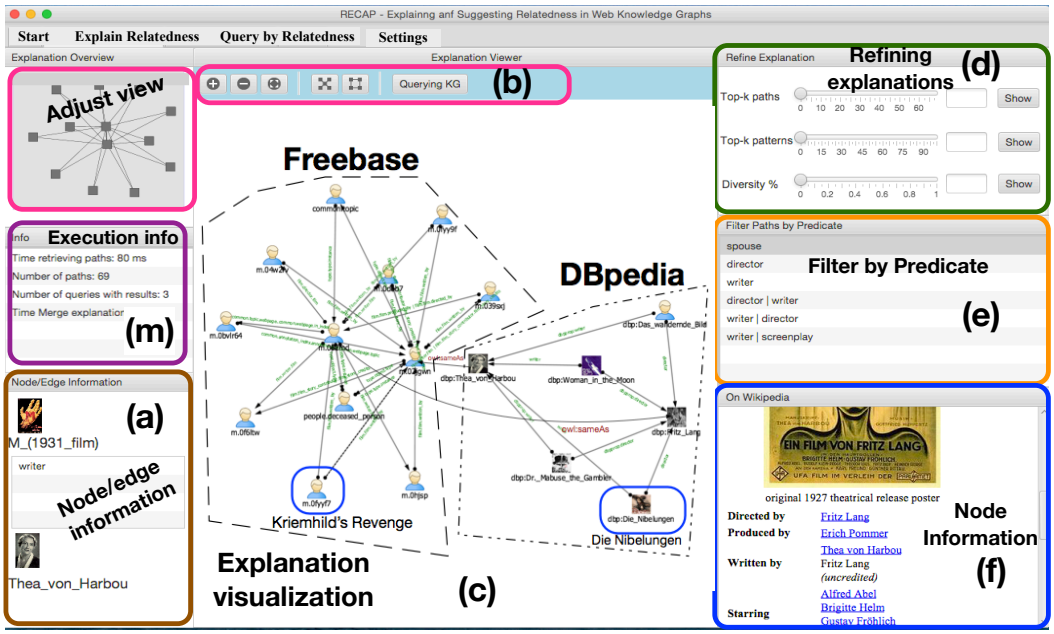

Figure 2: The Relatedness Explanation perspective.

3. RUNNING EXAMPLE

The goal of the demonstration is to cover a variety of examples rather than focusing on a single one. We will specifically stimulate the audience to pose examples from disparate knowledge domains; one starting point to pick datasets could be the LOD cloud [1] that at the moment collect thousands of SPARQL endpoints ranging from biology to movies. One peculiarity of RECAP is that differently from related research it does not require any data preprocessing thus being able to work with fresh data.

We have developed a main running example that will serve as a basis to explain the ideas underlying our framework. The example is on the domain of movies and involves the pair of entities *F. Lang* and *T. von Harbou*. The users inputs this pair to RECAP as shown in Fig. 3. The main GUI of the RECAP tool is shown in Fig. 3. The auto completion function in Fig. 3 (a) enables to find the correct entities of interest thanks to a short structured description provided for each entity. The interface also enables to chose the maximum path distance to be considered, as shown in Fig. 3 (b).

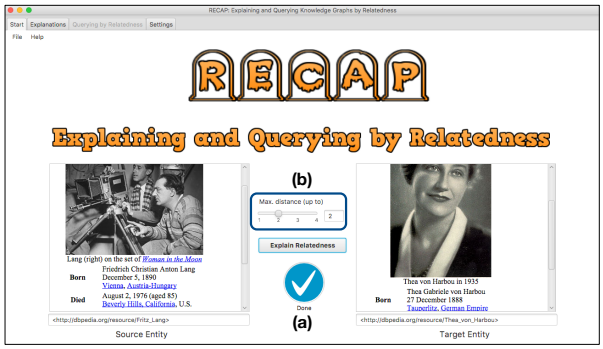

Figure 3: The RECAP main GUI.

When the user clicks on the *Explain Relatedness* button, the tool shows the progress of the execution. At this point the tool can produce different types of explanations, possibly combining knowledge from multiple KGs. An example of explanation is shown in Fig. 2 (c). This explanation combines information from Freebase and DBpedia. The explanation includes the top-20 most informative paths (out of 240) at max. distance 2 for the pair F. Lang T. von Harbou.

The interface provides statistics about the explanation building process such as number of paths and execution time Fig. 2 (m). When clicking on a node in an explanation, the user can visualize structured information about such node as shown in Fig. 2 (f). When clicking on an edge, information about the edge will be also visualized as shown in Fig. 2 (a). The visualization can be adjusted via the panel in Fig. 2 (b). Part (e) in Fig. 2 allows to filter an explanation according to certain types of predicates. In the previous example, the combination of Freebase and DBpedia allowed to refine knowledge about *Die Nibelungen* series by providing information about the episode titled *Kriemhild's Revenge*, co-written by F. Lang and T. von Harbou.

The interface shown in Fig. 4 handles the query by explanation functionalities. Fig. 4 (a) shows path patterns, that is, paths where nodes are replaced by variables; the instances associated to each pattern are listed at the bottom of Fig. 4 (a) and visually displayed in Fig. 4 (b). The system suggests different types of SPARQL queries as described in Section 2. Fig. 4 part (c) and (d) show an example of edge-isomorphic query. The query can be (manually) refined and then executed to find other pairs of entities. The top-5 pairs of entities found, and ranked by their popularity, are show in Fig. 4 (l). As an example, the pair (Gale Ann Hurd, James Cameron) share the same relatedness perspective as F. Lang T. von Harbou shown in Fig. 4 (b): J. Cameron was married to G. A. Hurd, he wrote the movie The Terminator where L. Hamilton (also married to J. Cameron) starred.

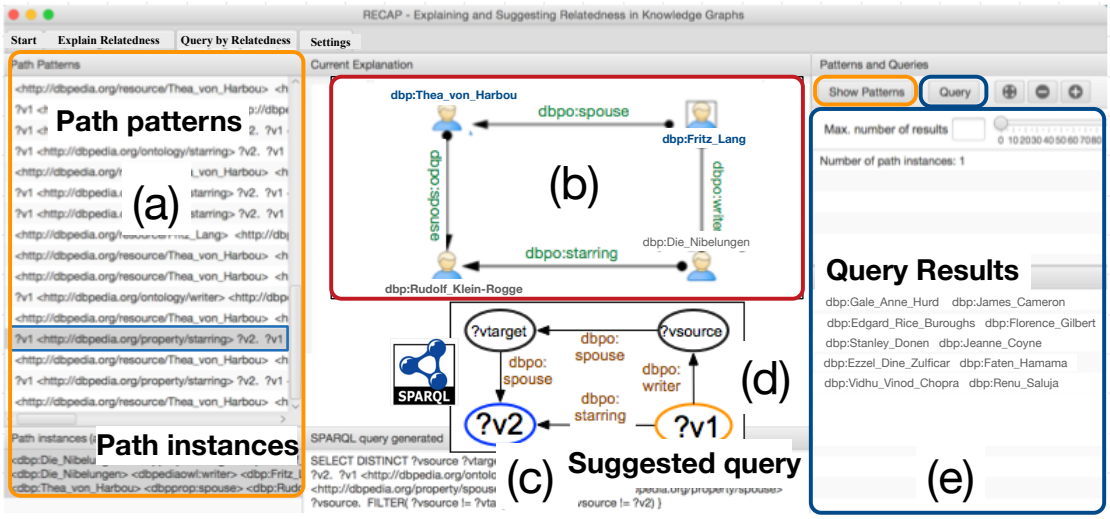

Figure 4: The Query by Relatedness Perspective.

4. DEMONSTRATION SCENARIO

The demo will start by providing a very brief overview about the notion of query explanation, also linking this idea to Google's KG. Then, we will run a set of examples and comment on the results by leveraging the RECAP GUI. In particular, the input will be provided as shown in Fig. 3. Then, for each example, we will show the different kinds of explanations that can be built. This part of the demo will focus on the second tab of the RECAP GUI shown in Fig. 2. We will pick specific examples having small and large explanations and underline the following aspects: (i) filtering explanations is a necessary step when the size of the explanation starts to grow; (ii) each type of explanation has pros and cons. Then, we will show how explanations can be used to build SPARQL queries that allow to retrieve other pairs of entities. This part of the demo will focus on the third tab of the GUI shown in Fig. 4. We will underline the following aspects: (i) types of queries that can be built from explanations; (ii) how to refine the suggested queries.

After introducing the functioning of the tool, we will let the audience have the opportunity to both pick their own examples and execute them. We will stress the fact that the tool can work with any SPARQL endpoint, without the need to preprocess data. To involve a broader audience we will make available, besides the standalone version of the tool, also a Web based interface so that anybody can experience with the tool from his/her own laptop.

4.1 Demonstration goals

The main goal of the demonstration is to introduce both the explanation and query by explanation approach. The desideratum is that, by practically experiencing with the RECAP tool, the audience will realize its potentiality in terms of knowledge discovery capabilities from a variety of KGs. Attention will also be paid to stress the fact that existing search engines equipped with KGs offer very limited querying and explanation capabilities. The flexibility of the tool to work with any KG and combine knowledge from multiple KGs will also be underlined. Another goal that we set is to stimulate new research on this topic. This will possibly happen by discussing the technical challenges and the design choice that we made to solve them. The idea is to underline the pros and cons of our approach with particular emphasis on the following aspects: (i) balance between path length and running time; (ii) motivate the design choice to work on top of KGs and access them only via endpoints.

5. REFERENCES

[1] LOD Cloud `http://lod-cloud.net`.

[2] A. Bonifati, R. Ciucanu, and S. Staworko. Interactive Join Query Inference with JIM. *PVLDB*, 7(13):1541–1544, 2014.

[3] G. Diaz, M. Arenas, and M. Benedikt. SPARQLByE: Querying RDF Data by Example. *PVLDB*, 9(13):1533–1536, 2016.

[4] Fang, Lujun and Sarma, Anish Das and Yu, Cong and Bohannon, Philip. REX: Explaining Relationships between Entity Pairs. *VLDB*, 5(3):241–252, 2011.

[5] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[6] S. Metzger, R. Schenkel, and M. Sydow. QBEES: query-by-example Entity Search in semantic Knowledge Graphs based on Maximal Aspects, Diversity-awareness and Relaxation. *JIIS*, pages 1–34.

[7] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Searching with XQ: the Exemplar Query Search Engine. In *SIGMOD*, pages 901–904, 2014.

[8] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar Queries: a New Way of Searching. *VLDB Journal*, 25(6):741–765, 2016.

[9] N. Nakashole, G. Weikum, and F. Suchanek. Discovering and Exploring Relations on the Web. *VLDB*, 5(12):1982–1985, 2012.

[10] G. Pirrò. REWOrD: Semantic Relatedness in the Web of Data. In *AAAI*, pages 129–135, 2012.

[11] G. Pirrò. Explaining and Suggesting Relatedness in Knowledge Graphs. In *ISWC*, pages 622–639, 2015.

[12] G. Pirrò and A. Cuzzocrea. RECAP: Building Relatedness Explanations on the Web. In *WWW*, pages 235–238, 2016.