

# DBpedia Profiler Tool: Profiling the Connectivity of Entity Pairs in DBpedia

José E. Talavera Herrera<sup>1</sup>, Marco Antonio Casanova<sup>1</sup>, Bernardo Pereira Nunes<sup>1,4</sup>,  
Giseli Rabello Lopes<sup>2</sup>, Luiz André P. Paes Leme<sup>3</sup>

<sup>1</sup> Department of Informatics – Pontifical Catholic University of Rio de Janeiro, RJ, Brazil  
{jherrera,casanova,bnunes}@inf.puc-rio.br

<sup>2</sup> Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil  
giseli@dcc.ufrj.br

<sup>3</sup> Federal Fluminense University, Niterói, RJ, Brazil  
lapaesleme@ic.uff.br

<sup>4</sup> Federal University of the State of Rio de Janeiro, Rio de Janeiro, RJ, Brazil  
bernardo.nunes@uniriotec.br

**Abstract.** A connectivity profile for a pair of entities provides a concise explanation about how the entities are related. This paper describes a tool, called DBpedia Profiler, which implements a strategy to generate connectivity profiles for entities represented in DBpedia. The tool uses SPARQL queries to generate relationship paths and follows a technique called path streams to analyze the collected paths. The paper also presents a comparison between DBpedia Profiler and the state-of-the-art tool RECAP. The experiments show that the proposed tool outperforms RECAP in terms of performance and usability. Additionally, DBpedia Profiler provides extra support for mining the connectivity of entities.

## 1 Introduction

This paper explores DBpedia to discover how two entities are related. For example, it is now (but probably not in 20 years from now) common knowledge that “Barack Obama” and “Michelle Obama” are related in at least two ways: they are married and they are both alumni of Harvard Law School. It is not so obvious, though, to discover how “Albert Einstein” and “Kurt Gödel” are related. By exploring DBpedia, for example, one may find that:

- a) “Albert Einstein” was influenced by “Ernst Mach”, which was influenced by “David Hume”, which influenced “Edmund Husserl”, which influenced “Kurt Gödel”.
- b) “Albert Einstein” was influenced by “Baruch Spinoza”, which was influenced by “Giordano Bruno”, which influenced “Gottfried W. Leibniz”, which influenced “Kurt Gödel”.

These two relationship paths can be abstracted by saying that

- c) X was influenced by a philosopher, which was influenced by a philosopher, which influenced a philosopher, which influenced Y; with X instantiated by “Albert Einstein” and Y by “Kurt Gödel”.

Note that (c) provides a concise explanation of the connectedness (or relationship) between “Albert Einstein” and “Kurt Gödel” in the Philosophy domain.

When two entities are directly related, such as “married to”, or related by a short composition of relationships, such as “coauthor”, that have a natural language term to denote the composition, it is not too difficult to explain how the two entities are related. However, this is not the case with the example relating “Albert Einstein” and “Kurt Gödel” described in the previous paragraph. To address this issue, we introduce the concept of *connectivity profile* for a pair of entities, defined as a concise explanation about how the entities are related in a knowledge base (KB), and rephrase the goal of this paper as:

- “Given a knowledge base and a pair of entities, create a *connectivity profile* for the pair of entities”.

Several tools were developed [1,2,3,4,5] to explore knowledge bases and generate connectivity profiles. In general, such tools: (1) search for relationship paths between pairs of entities – the larger the number of paths found, the stronger the connectivity between the entities is likely to be; (2) group the paths found; and (3) summarize the groups of paths to present a connectivity profile of the pair of entities to the user. However, this simple strategy poses three challenges:

- (1) *How to find relationship paths between a given pair of entities in a knowledge base?*
- (2) *How to group the relationship paths in a meaningful way?*
- (3) *What characteristics of the groups of paths must be selected to generate a connectivity profile?*

In this paper, we describe a tool, called DBpedia Profiler, which implements a strategy to generate connectivity profiles for pairs of entities represented in DBpedia. To address the first challenge, the strategy resorts to specialized queries over DBpedia to identify RDF paths that connect the given pair of entities. As for the second challenge, it adopts a strategy based on semantic annotations, as in [7], which uses similarity metrics to group annotations about an itemset and summarize the observed groups. The major difference lies in that our strategy works over sets of RDF paths rather than itemsets. The strategy covers the last challenge by resorting to the topic categories assigned to the entities and to the relationships between the entities found in the RDF paths. We also compare the proposed tool with RECAP [3].

The contributions of this paper can be summarized as: (1) description of the DBpedia Profiler tool; and a (2) presentation of a comparison between DBpedia Profiler and the state-of-the-art tool RECAP.

The rest of this paper is structured as follows. Section 2 summarizes related work. Section 3 describes the proposed tool in detail. Section 4 compares DBpedia Profiler with our implementation of RECAP. Finally, Section 5 presents the conclusions.

## 2 Related Work

*Explaining Relationships in Knowledge Bases.* The work proposed in [4] used entity classes and property labels to create descriptors and to facilitate the understanding of

the explanations discovered. However, the number of descriptors may be excessive, which complicates the exploration of the explanations returned. Ranking of explanations was proposed in [1,2,3,4,5]. These works used measures of importance to identify interesting explanations, such as frequency, rarity and informativeness. A ranking excludes information, with the side effect that the user cannot infer the hidden meaning of each explanation, and thus still decide which to explore. Differently from these works, in this paper, we create a connectivity profile for a pair of entities that uses the class hierarchy and the topic categories of the underlying knowledge base, summarizes a large set of explanations, retaining only the most representative ones (without losing relevant information), and classifies each explanation using topic categories.

*Visualizing connectivity information between entities.* Some tools are available online to visualize connectivity information between entities, such as [1,4,16]. In this paper, we compare DBpedia Profiler with RECAP [3], which mines RDF graphs and visualizes information using simple paths and graph patterns. Since RECAP is not available online, we implemented it and made it available online. DBpedia Profiler uses meta-paths as in [4,8] and the categories of Wikipedia to visualize information.

Table 1 provides an overview of related approaches based on five attributes: (i) Graph support; (ii) Output types; (iii) Querying capabilities (ability to explore RDF graphs without the use of SPARQL and SPARQL-like languages); (iv) Availability of the tools; and (v) Connectivity Profile (ability to generate a concise explanation between entities in KBs).

**Table 1.** Comparison of DBpedia Profiler with related tools.

	<b>RDF Graph</b>	<b>Output Types</b>	<b>Querying Capabilities</b>	<b>Availability</b>	<b>Connectivity Profile</b>
<b>DBpedia Profiler</b>	DBpedia	Graph, Paths, Explanation Patterns, Topic Categories	Yes	Yes	Yes
<b>RECAP [3]</b>	Any	Graph, Paths	Yes	No	No
<b>RelFinder [1]</b>	Any	Graph	No	Yes	No
<b>Explass [4]</b>	DBpedia	Paths	No	Yes	No

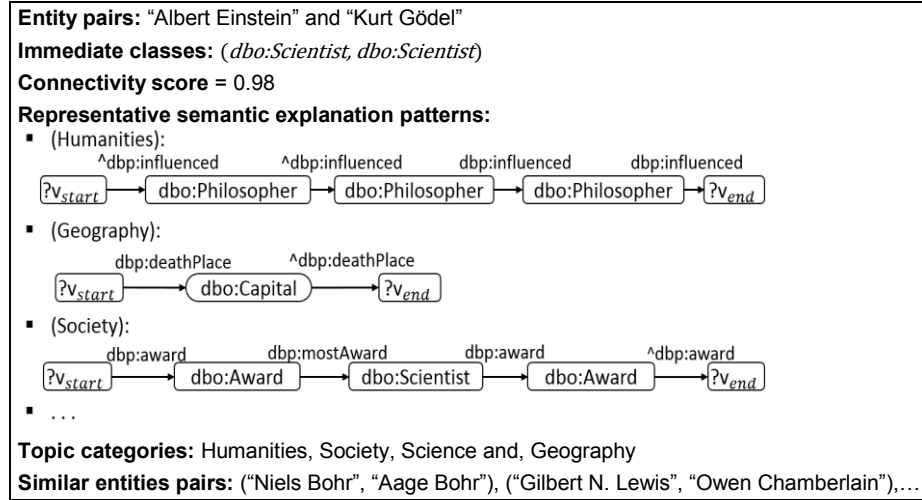
### 3 The DBpedia Profiler Tool

#### 3.1 Definition of Connectivity Profile

We define a *connectivity profile* for a pair of entities  $u$  and  $v$ , with maximum distance  $k$ , as a tuple  $C = ((C_u, C_v), s, EXP[u, v], T, E)$ , where  $C_u$  and  $C_v$  are the immediate classes of  $u$  and  $v$ ,  $s = cs(u, v)$  is the connectivity score of  $u$  and  $v$ ,  $EXP[u, v]$  is a set of representative semantic explanation patterns (of length at most  $k$ ),  $T$  is the set of top weighted topic categories for  $u$  and  $v$  and  $E$  is a set of entity pairs similar to  $u$  and  $v$ .

**Example 1.** Fig. 1 shows a connectivity profile for “Albert Einstein” and “Kurt Gödel”, indicating: (1) the immediate classes of the entities, which are both *dbo:Scientist*; (2) a connectivity score of 0.98; (3) three representative semantic explanation patterns; (4)

the topic categories of the explanations: “Humanities”, “Society”, “Science” and “Geography”; (5) a set of similar entities pairs: “Niels Bohr” and “Aage Bohr”, etc. Indeed, we observe that the pairs (“Albert Einstein”, “Kurt Gödel”) and (“Niels Bohr”, “Aage Bohr”) have in common the facts that they were awarded science prizes, died in the same place and were influenced by the same group of people.



**Fig. 1.** A schematic view of the connectivity profile of Albert Einstein and Kurt Gödel.

Let  $B$  stand for DBpedia,  $G_B = (N_B, E_B, e_B)$ ,  $C_B$  be the set of classes defined in DBpedia,  $H_B = (C_B, S_B)$  be the DBpedia class hierarchy and  $\mathcal{W}$  be the set of top-level categories of Wikipedia [12].

As explained in what follows, the proposed tool constructs connectivity profiles over DBpedia, using the following functions and thresholds:

- the entity class score  $ecs: EN_B \times C_B \rightarrow \mathbb{R}$
- the weighted topic category function  $wcat: C_B \times \mathcal{W} \rightarrow \mathbb{R}$
- the connectivity score  $cs: EN_B \times EN_B \rightarrow \mathbb{R}$
- the coherence measure  $coh: SEP \rightarrow \mathbb{R}$
- the similarity measure  $sim: SEP \times SEP \rightarrow \mathbb{R}$
- a coherence threshold  $\tau_c$
- a topic category threshold  $\tau_t$

DBpedia Profiler successively computes, for a pair of entities  $u$  and  $v$ :

- RDF paths from  $u$  to  $v$  of length  $k$
- explanation patterns for  $u$  and  $v$  of length  $k$
- semantic explanation patterns for  $u$  and  $v$  of length  $k$
- representative semantic explanation patterns for  $u$  and  $v$  of length  $k$

### 3.2 Path-stream approach

Fig. 2 depicts the overall structure of the DBpedia Profiler tool. Briefly, at the application layer, the user provides a pair of entities and interacts with the tool to browse the connectivity profile generated for the pair of entities.

At the data processing layer, DBpedia Profiler adapts the approach proposed in [10] for query execution over SPARQL endpoints to generate a connectivity profile for the given pair of entities (CPPE). First, it issues SPARQL queries to the DBpedia SPARQL endpoint to retrieve all RDF paths between the given pair of entities with a maximum predefined length, following the strategy defined in [1]. This process generates a stream of RDF paths that is shared between the different operations that generate the profile.

At the data pre-processing layer, DBpedia Profiler builds an index over the DBpedia class hierarchy to help identify the immediate classes of an entity [8,14,15] and map each of these classes into topic categories [12].

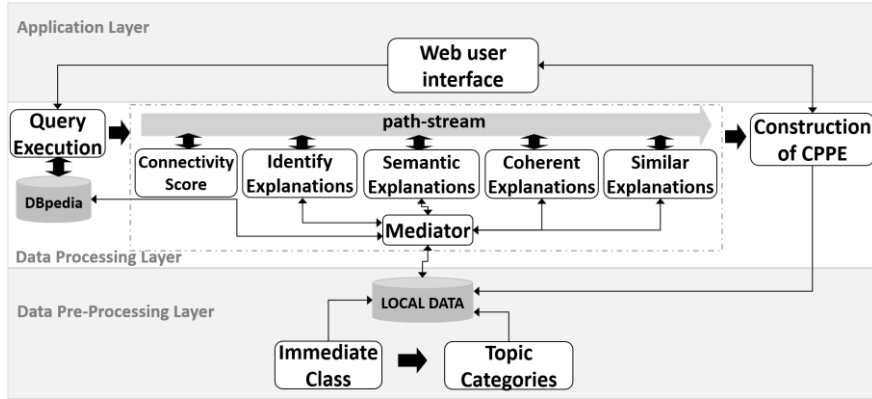


Fig. 2. Overall structure of the DBpedia Profiler tool.

### 3.3 Exploring the Class Hierarchy of DBpedia

DBpedia Profiler implements the entity class score  $ecs: N_B \times C_B \rightarrow \mathbb{R}$  as

$$ecs(e, C) = distance(Root, C) * \frac{frequency(e, C)}{\log(frequency(C))} \quad (1)$$

where  $distance(C, Root)$  computes the distance between a class  $C$  and the root class in  $H_B$ ,  $frequency(e, C)$  is the local count of a class  $C$  in the description text of  $e$  in  $B$  and  $frequency(C)$  computes the overall count of  $C$  in  $H_B$ .

DBpedia Profiler implements a weighted class topic category function based on the top-level categories of Wikipedia as follows. We first recall that, in Wikipedia, articles are manually annotated with categories. Since each class  $C$  of DBpedia has a corresponding article  $A_C$  in Wikipedia [11], we use the categories of  $A_C$  to relate  $C$  to the main top categories of Wikipedia, as in Kawase et al. [12].

In detail, the *weighted class topic category function*  $wcat: C_B \times \mathcal{W} \rightarrow \mathbb{R}$  is defined as

$$wcat(C, t) = \sum_{w \in W(C)} weight(w, t) \quad (2)$$

where  $C$  is a class in the class hierarchy of DBpedia,  $t$  is a Wikipedia top category,  $W(C)$  are the categories of the Wikipedia article associated with  $C$  and  $weight$  measures the relation between a Wikipedia category  $w$  and a top category  $t$  and is defined as [12]

$$weight(w, t) = \frac{1}{popularity(t)} * \frac{1}{distance(w, t)} \quad (3)$$

where  $popularity(t)$  indicates the popularity of a given main top topic category  $t$  and,  $distance(w, t)$  is the distance of a category  $w$  to the main top category  $t$ .

### 3.4 Exploring the RDF graph of DBpedia

DBpedia Profiler adopts the *semantic connectivity score* (SCS) [6,17] as the connectivity score (cs). SCS is a variation of the Katz index [13], introduced to estimate the relatedness of actors in a social network, and is defined as

$$SCS(u, v) = \sum_{l=1}^k \beta^l * |L^{<l>}| \quad (4)$$

where  $|L^{<l>}|$  is the number of RDF paths from  $u$  to  $v$  of length  $l$  and  $k$  is the maximum distance considered between  $u$  and  $v$ . The damping factor  $\beta$  is responsible for exponentially penalizing longer paths; the smaller this factor, the smaller the contribution of longer paths to the final score. The final score is normalized to fall in the interval  $[0, 1]$ .

To compute the set  $P[u, v]$  of all paths of  $B$  that start on  $u$  and end on  $v$  with maximum length  $k$ , DBpedia Profiler adopts a strategy similar to that of RelFinder [1]. Briefly, the tool issues SPARQL queries to the RDF graph of DBpedia to retrieve all paths that start on  $u$  and end on  $v$  and capture the length of each RDF path. Just as in [6,17], DBpedia Profiler allows the user to filter out entities that belong to certain classes from the RDF paths. The execution of these queries creates a path-stream processing.

After, DBpedia Profiler computes the partition  $\{P_1, \dots, P_n\}$  of  $P[u, v]$  and the set  $\{SEP_1, \dots, SEP_n\}$  of semantic explanation patterns for  $u$  and  $v$  of length  $k$  exactly as described earlier, using the entity class score introduced in Section 3.3.

To compute the set of representative semantic explanation patterns, DBpedia Profiler implements the coherence measure and the similarity metric for semantic explanation patterns as follows.

Recall that a semantic explanation pattern  $SEP$  for a given pair of entities is coherent iff it satisfies the following two conditions: (i) the properties in  $SEP$  frequently co-occur in the knowledge graph; (ii) the entity classes in  $SEP$  have similar topic categories. To satisfy the first condition, DBpedia Profiler uses the Pointwise Mutual Information (PMI) of each property pair [18]. PMI measures the co-occurrence strength between two items; it works by relating the probabilities of the individual occurrence of the items to the probability of both items occurring together. Based on the co-occurrence of the properties, we estimate the PMI score of each property pair in a semantic explanation pattern.

Let  $SEP$  be a semantic explanation pattern and  $r$  and  $s$  be two properties that occur in  $SEP$ . The PMI of  $r$  and  $s$  is defined as:

$$PMI(r, s) = \log \left( \frac{f(r, s)}{f(r) * f(s)} \right) \quad (5)$$

where  $f(r, s)$  measures the co-occurrence frequency of  $r$  and  $s$  and  $f(r)$  and  $f(s)$ , the individual frequency of  $r$  and  $s$ , as shown in Table 2.

**Table 2.** SPARQL Queries to compute the PMI.

$f(r, s)$	<code>SELECT COUNT (DISTINCT ?e) as ?count WHERE { ?s1 r ?e . ?e s ?o2 }</code>
$f(r)$	<code>SELECT COUNT (DISTINCT ?e) as ?count WHERE { ?s1 r ?e }</code>
$f(s)$	<code>SELECT COUNT (DISTINCT ?e) as ?count WHERE { ?e s ?o2 }</code>

DBpedia Profiler covers the second condition with a strategy based on the intersection of the topic categories of the classes in the semantic explanation pattern  $SEP$ . Recall that the class topic category set function  $scat : C_B \rightarrow 2^w$  maps each class into a set of topic categories. The *topic coherence* of a semantic explanation pattern  $SEP$  is defined as

$$th(SEP) = \frac{|\cap_{C \in cl(SEP)} scat(C)|}{|\cup_{C \in cl(SEP)} scat(C)|} \quad (6)$$

where  $cl(SEP)$ , we recall, is the set of classes that occur in a semantic explanation pattern  $SEP$ . Then, the higher  $th(SEP)$  is, the more coherent  $SEP$  will be.

Finally, DBpedia Profiler computes the *coherent measure* of  $SEP$  as

$$coh(SEP) = \frac{\text{median}\{PMI(r_i, r_j) \mid r_i \text{ and } r_j \text{ are properties in } SEP \text{ and } i < j\}}{th(SEP)} + \quad (7)$$

where the first factor is the PMI median score of the property pairs in  $SEP$  and the second factor is the coherence of the classes.

DBpedia Profiler adopts the Jaccard distance as the similarity measure for pairs of semantic explanation patterns. Given two semantic explanation patterns  $SEP_1$  and  $SEP_2$ , the *Jaccard distance* between  $SEP_1$  and  $SEP_2$  is defined as

$$Jaccard(SEP_1, SEP_2) = \frac{|URIs(SEP_1) \cap URIs(SEP_2)|}{|URIs(SEP_1) \cup URIs(SEP_2)|} \quad (8)$$

where  $URIs(SEP_i)$  denotes the set of RDF terms in a semantic explanation pattern  $SEP_i$ .

DBpedia Profiler computes the Jaccard distance in the path stream processing for each pair of semantic explanation patterns, after filtering. Then, it executes the One-Pass Microclustering algorithm, proposed in [7], to group the semantic explanation patterns. With the Jaccard distance, we expect to extract clusters such that the distance between inner-cluster explanations are bounded. The algorithm terminates when the minimal distance between clusters becomes larger than a user-specified threshold. The medoid of each cluster is selected as the representative semantic explanation pattern. This algorithm processes the set of semantic explanation patterns in one pass and, thus, is more efficient than those proposed in [7]. Using the set of representative semantic

explanation patterns, DBpedia Profiler finally constructs the set of weighted topic categories and the set of similar entity pairs.

### 3.5 Construction of the Connectivity Profile

Finally, DBpedia Profiler constructs the connectivity profile for a pair of entities  $u$  and  $v$  in DBpedia, with maximum distance  $k$ , as the tuple  $C = ((C_u, C_v), s, EXP[u, v], T, E)$ , where

- $C_u$  and  $C_v$  are the immediate classes of  $u$  and  $v$
- $s = SCS(u, v)$
- $EXP[u, v]$  is the set of representative semantic explanations patterns, of length at most  $k$
- $T$  is the set of top weighted topic categories for  $P$
- $E$  is the set of similar entity pairs, using the set of representative semantic explanation patterns in  $P$

## 4 Implementation and Evaluation

We compare and evaluate the process of explanation generation of DBpedia Profiler (available at <http://lod2.inf.puc-rio.br/scs/dbprofiles>) with that of RECAP (available at <http://lod2.inf.puc-rio.br/scs/recaps>). We do not provide a detailed comparison of DBpedia Profiler with Exlass [4] and RelFinder [1] because RECAP has already been shown to outperform these tools in [3,4].

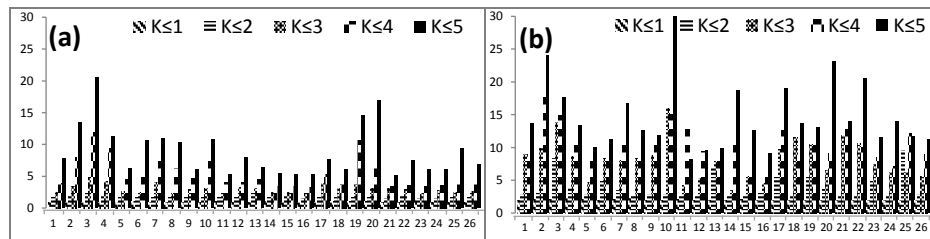
The experiments were performed on an Intel Core i7 CPU 950 with 12 GB. The data processing layer of DBpedia Profiler was implemented in LUA and the application layer in PHP. The SPARQL query requests use LuaSocket and the parallel processing is supported by GNU parallel. The RDF data is encoded and stored in memory with Redis, for the effective processing of the path-stream. Since RECAP was originally not an online tool, we also implemented the explanation generation of RECAP, which is now available for the community scientific. DBpedia Profiler and RECAP were implemented to explore only DBpedia.

### 4.1 Performance Evaluating

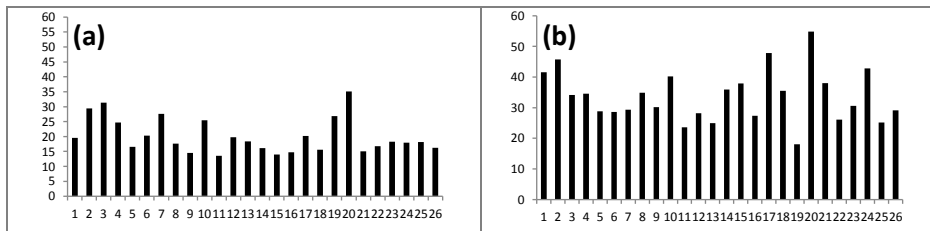
We first compared the performance of DBpedia Profiler and RECAP for increasing values of  $k$  up to  $k=5$  for the entire process of explanation generation, using the dataset with 26 pairs originally used to evaluate RECAP [3]. As in [3], for each  $k$ , we generated all paths of length less than or equal to  $k$ . The evaluation considered the number of results returned for the SPARQL queries to retrieve the paths between two entities as a parameter, called LIMIT in the SPARQL query language [9]. We do not have information about this parameter in the evaluation of [3]. Since some queries require long processing times, LIMIT prevents of a possible timeout and improves the processing time. Considering a desired response time, this parameter was set to LIMIT=300 in DBpedia Profiler and to LIMIT=50 in RECAP. This configuration, apparently unjust to DBpedia

Profiler, compensates for the extra time RECAP takes to de-duplicate and rank the patterns [2]. DBpedia Profiler was set to disregard the 10% less coherent explanations, which is a different form of setting the coherence threshold; the Jaccard distance between the explanations was set to 0.92. The Jaccard distance is configured according to the experiments executed in [7]. RECAP computes the amount of information of a path and considers only the top-5 most informative paths.

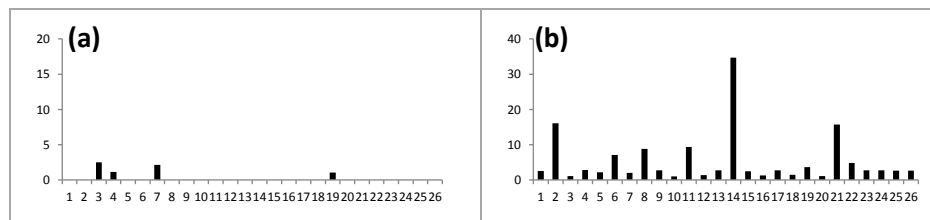
Figs. 3 (a) and (b) show the average runtime of 5 runs for each of the 26 pairs. The average runtime for DBpedia Profiler was ~3.9 sec and that for RECAP, ~7.8 sec. Even with LIMIT=300, DBpedia Profiler achieved a shorter runtime than RECAP, with LIMIT=50. We recall that RECAP executes several online SPARQL query requests to compute the most informative path/pattern, whereas DBpedia Profiler has a pre-processing stage to build an index over the DBpedia class hierarchy to help compute the immediate classes, topic categories and the co-occurrence of pairs of properties.



**Fig. 3.** (a) DBpedia Profiler process; (b) RECAP process. Y-axis: time(s); X-axis: the entity pairs.



**Fig. 4.** First Run of (a) DBpedia Profiler; (b) RECAP. Y-axis: time(s); X-axis: entity pairs.



**Fig. 5.** Computing Explanation to (a) DBpedia Profiler; (b) RECAP. Y-axis: time(s); X-axis: entity pair.

Fig. 4 shows the runtime of the first run to generate the explanation patterns for DBpedia Profiler and RECAP, with  $k=5$  and the configuration mentioned above. These measurements reflect the behavior of the tools with respect to a query user in real-time. The average runtime of DBpedia Profiler was  $\sim 20$  secs and of RECAP,  $\sim 34$  sec. For a second run, DBpedia responds with cached data, which explains the results in Fig. 3. This experiment therefore reveals the efficiency of the stream processing of DBpedia Profiler, when compared with the parallel processing of RECAP.

Fig. 5 plots the time to compute explanations patterns for DBpedia Profiler and RECAP, just for one run, with  $k=4$  and  $LIMIT=300$ , for DBpedia Profiler, and  $LIMIT=50$ , for RECAP. These measurements were collected after the experiment of Fig. 3; thus, both tools took advantage of the cached data. RECAP computed explanations in  $\sim 5.3$  sec, on the average. We recall that RECAP spends time computing the most informative paths and then combines the paths using an approach similar to that adopted in [2]. The combination of paths to compute explanations requires checking for duplicates [2], which has a high computational cost. Finally, RECAP computes the most informative patterns and presents them to the user. DBpedia Profiler computed explanations in  $\sim 0.15$  sec, on the average. We recall that DBpedia Profiler uses the One-Pass Micro-clustering algorithm to group the explanations. The complexity of this algorithm is linear, with respect to the number of explanations. The Jaccard distance (set to 0.92) was computed in the stream processing, together with the coherence score, using data computed at the pre-processing stage.

## 4.2 User Evaluation of the Explanations

This experiment aims at investigating whether DBpedia Profiler provides useful explanations to the user and comparing DBpedia Profiler with RECAP.

The experiment involved 20 graduate students, with some experience in Semantic Web technology. Each participant was assigned 6 random pairs among the 26 entity pairs. Prior to the experiment, the participants had a training stage to familiarize themselves with the tools. Following the methodology in [4], the participants were given a task and a set of seven questions; the response to each question was in the form of an agreement value, ranging from 1 (min) to 5 (max). Q6 and Q7 were not considered in [4]. Q6 aimed at knowing whether the tools provided useful information to resolve the different tasks, since typically redundant information does not help the user. Q7 allowed the user to make a qualitative evaluation on the tool.

The results are reported in Table 3 and capture the experience of the users with the tools. We used Student's  $t$ -distribution to verify whether the mean ratings were statistically significant, with  $p < 0.05$ . The reliability analysis of questionnaire had a consistence of 0.88. According to Q2 and Q5, DBpedia Profiler provides better support to explain the tasks than RECAP. In Q6, even when RECAP combines the top-10 paths some explanations are redundant. In Q7, the users expressed their satisfaction with the topic categories, immediate classes and the connectivity score returned by DBpedia Profiler. The SPARQL query patterns generated by RECAP were more complex and, in some cases, the entity pairs recovered were more specific than the pairs recovered by DBpedia Profiler. In general, according to average of the result of the Table 3, DBpedia

Profiler provides better information quality about the connectivity of the entities than RECAP.

**Table 3.** Questions/responses: mean (standard deviation).

Question	DBpedia Profiler	RECAP
Q1: Information overview	4.03 (0.20)	2.96 (0.48)
Q2: Easiness in finding information	4.09 (0.19)	3.28 (0.47)
Q3: Easiness in comparing/synthesizing info	4.02 (0.22)	3.09 (0.61)
Q4: Comprehensive support	4.07 (0.21)	3.00 (0.43)
Q5: Sufficient support to the task	3.96 (0.26)	3.08 (0.44)
Q6: Redundancy in information	3.54 (0.36)	2.64 (0.58)
Q7: General Comments about the task and tool		

## 5 Conclusions

We first introduced the DBpedia Profiler tool, which implements a strategy to generate connectivity profiles for pairs of entities for DBpedia and follows an architecture based on RDF path stream processing. We also compared DBpedia Profiler with our implementation of RECAP and indicated the differences between the tools in terms of performance and quality of the explanations returned.

The major features of DBpedia Profiler can be summarized as follow. By exploring the metadata aspects of DBpedia and the top main categories of Wikipedia, DBpedia Profiler: implements an algorithm to find the immediate class of entities; and enriches the uncovered explanations. By exploring the DBpedia graph, DBpedia Profiler: computes a connective score to indicate the degree of relatedness of the entities; implements a new metric that measures the coherence of an explanation, through the co-occurrence of the properties and the intersection of topic categories; groups explanations using the Jaccard Distance and the One-Pass Microclustering algorithm [7] to summarize sets of explanations.

Although the categories of Wikipedia add semantics to the explanation, some categories have a very broad context, which complicates the classification of explanations. For example, People, Society and Humanities share many contexts with Person. A major goal for future research is to develop an optimized classifier to improve the semantic of the explanations.

## References

1. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., and Stegemann, T. RelFinder: Revealing Relationships in RDF Knowledge Bases. SAMT, LNCS vol. 5887 (2009), page 182-187.
2. Fang, L., Sarma, A. D., and Yu, C. & Bohannon, P. REX: Explaining Relationships between Entity Pairs. PVLDB, 5, 241-252. 2011.
3. Pirrò, G. Explaining and Suggesting Relatedness in Knowledge Graphs. ISWC 2015.
4. Gong, C., Yanan, Z., and Yuzhong, Q. Explax: Exploring Associations between Entities via Top-K Ontological Patterns and Facets. ISWC 2014, pp. 422-437.

5. Mohan, Y., Bolin, D., Surajit, C., and Kaushik Chakrabarti. Finding patterns in a knowledge base using keywords to compose table answers. *PVLDB* 7, 14, 1809-1820, 2014.
6. Nunes, B. P., Dietze, S., Casanova, M.A., Kawase, R., and Fetahu B., Nejd W. Combining a co-occurrence-based and a semantic measure for entity linking. *ESWC* 2013.
7. Mei, Q., Xin, D., Cheng, H., Han, J., and Zhai, C. Generating semantic annotations for frequent patterns with context analysis. *SIGKDD* 2006.
8. Meng, C., Cheng, R., Maniu, S., Senellart, P., and Zhang, W. Discovering meta-paths in large heterogeneous information networks. *WWW* 2015, pp. 754–764.
9. Harris, S. and Seaborne, A. *SPARQL1.1Query Language W3C Recommendation*, 2013.
10. Demter, J., Auer, S., Martin, M., and Lehmann, J. Lodstats – an extensible framework for high-performance dataset analytics. *EKAW* 2012.
11. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal*. 2013.
12. Kawase, R., Siehndel, P., Nunes, B. P., Herder, E., and Nejd, W. Exploiting the wisdom of the crowds for characterizing and connecting heterogeneous resources. *HT* 2014.
13. Katz, L. A new status index derived from sociometric analysis. *Psychometrika* 18(1), 39–43, 1953.
14. Assis P., Casanova, M.A., Laender A., and Milidiú R. Improving Relation Extraction by Using an Ontology Class Hierarchy Feature. *WISE* 2015, LNCS vol. 9419 pp. 241-249.
15. Böhm, C., Kasneci, G., and Naumann, F. Latent topics in graph-structured data. *CIKM* 2012, pp. 2663-2666.
16. Hartig, O., Bizer, C., and Freytag, J.C.: Executing SPARQL queries over the Web of Linked Data. *ISWC* 2009, pp. 293–309.
17. Nunes, B. P., Herrera, J., Taibi, D., Lopes, G. R., Casanova, M. A., and Dietze, S. SCS connector - Quantifying and visualising semantic paths between entity Pairs. *ESWC* 2014.
18. Church, K.W. and Hanks, P. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22-29. 1990.