An Entity Relatedness Test Dataset

José Eduardo Talavera Herrera[1(✉)], Marco Antonio Casanova[1], Bernardo Pereira Nunes[1,2], Luiz André P. Paes Leme[3], and Giseli Rabello Lopes[4]

[1] Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
{jherrera, casanova, bnunes}@inf.puc-rio.br

[2] Federal University of the State of Rio de Janeiro, Rio de Janeiro, RJ, Brazil

[3] Fluminense Federal University, Niterói, RJ, Brazil
lapaesleme@ic.uff.br

[4] Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
giseli@dcc.ufrj.br

Abstract. A knowledge base stores descriptions of entities and their relationships, often in the form of a very large RDF graph, such as DBpedia or Wikidata. The entity relatedness problem refers to the question of computing the relationship paths that better capture the connectivity between a given entity pair. This paper describes a dataset created to support the evaluation of approaches that address the entity relatedness problem. The dataset covers two familiar domains, music and movies, and uses data available in IMDb and last.fm, which are popular reference datasets in these domains. The paper describes in detail how sets of entity pairs from each of these domains were selected and, for each entity pair, how a ranked list of relationship paths was obtained.

Keywords: Entity relatedness · Relationship path · Path ranking · Linked data · Knowledge bases

1 Introduction

A Knowledge Base (KB) stores descriptions of entities and their relationships, often in the form of a very large RDF graph, such as DBpedia or Wikidata. A *relationship path* between an entity pair is a path in an RDF graph that connects the nodes that represent the entities. The *entity relatedness problem* refers to the question of computing the relationship paths that better describe the connectivity between a given entity pair.

Several approaches [1–6] have been proposed to address the entity relatedness problem. They apply a simple strategy: (1) search for relationship paths between the given entity pair – the larger the number of paths found, the stronger the connectivity between the entities is likely to be; and (2) sort the paths found and select the relevant ones. However, there currently is no adequate benchmarks to measure the effectiveness of such approaches. In some cases, expert users evaluate the results, and an apparently reliable method to judge the effectiveness of the approach is introduced. In others, a *ground truth* is created, which is a difficult and time-consuming task, and hardly the

C. d'Amato et al. (Eds.): ISWC 2017, Part II, LNCS 10588, pp. 193–201, 2017.
DOI: 10.1007/978-3-319-68204-4_20

authors make the resources available. Thus, an open challenge is: *How to evaluate and compare approaches that address the entity relatedness problem?*

The major contribution of this paper is a dataset created to support the evaluation of approaches that address the entity relatedness problem, which we refer to as the *Entity Relatedness Test Dataset*. The dataset contains entities and relationship paths extracted from DBpedia that pertain to two familiar domains, music and movies, and additional data extracted from the Internet Movie Database – IMDb and last.fm, which are popular reference datasets in these domains. The dataset and resources are available at [17–21].

The paper describes in detail the major steps and design decisions behind the construction of the dataset. The first design decision was to select DBpedia as the reference knowledge base, from which we extracted relationships paths. The second design decision was to select the movies and music domains, which are backed up by two well-known datasets, IMDb and last.fm, from which we extracted reliable domain-specific knowledge. The dataset construction process involved three major steps. The first step consisted in the selection of a set of entity pairs from the music and movies domains. The second step referred to the extraction of a set of relationship paths from DBpedia, for each entity pair. The final step was to rank the paths, based on information extracted from IMDb and last.fm, and to select the top-k ones.

This paper is structured as follows. Section 2 summarizes related work. Section 3 introduces a generic strategy to find and rank relationship paths. Section 4 describes the construction of the dataset. Finally, Sect. 5 presents the conclusions.

2 Related Work

Finding and Ranking Relationship Paths Between a Given Entity Pair in a Knowledge Base. RECAP [3], EXPLASS [4] and DBpedia Profiler [6] implemented path finding processes in an RDF knowledge base with the help of SPARQL queries [9]. REX [2] used two breadth-first searches on the RDF graph to enumerate relationship paths between two entities, and considered the degree of a node as an activation criterion to prioritize nodes. Likewise, the work in [15] used the Jaccard similarity to compute an approximated minimal distance between the start and the end nodes, and to discover meaningful connection between the nodes.

Evaluating Relationship-Path Ranking in a Knowledge Base. Path-ranking measures were proposed in [3, 6, 8] to rank relationship paths in knowledge databases. Some approaches [3, 4, 6] evaluated relationship path rankings with the help of user experiments. However, the evaluation methods did not clearly define the capabilities of the approaches analyzed. The work proposed in [10] argued that entity similarity heuristics increase the relevance of the links between nodes. The authors compared and measured the effectiveness of different search strategies through user experiments.

In this paper, we describe a dataset containing entity pairs and relationship paths in two entertainment domains, music and movies, to compare approaches that address the entity relatedness problem.

3 A Generic Relationship Path Finding and Ranking Process

An RDF graph G is a set of RDF triples of the form $G = \{(\mathrm{s, p, o})\} = (V, E)$, where the subject is an entity $s \in V$, and it has property $p \in E$ whose value is an object $o \in V$, which is either another entity. Particularly, p is seen as the edge that link the entities s and o in an RDF Graph. We will use the terms *entity* and *node* of G interchangeably.

A *relationship path* in G between nodes w_0 and w_k in G is an expression of the form $(w_0, p_1, w_1, p_2, w_2, \ldots, p_{k-1}, w_{k-1}, p_k, w_k)$, where: k is the *length* of the path; w_i is a node of G such that w_i and w_j are different, for $0 \leq i \neq j \leq k$; and either (w_i, w_{i+1}) or (w_{i+1}, w_i) are edges of G labeled with p_{i+1}, for $0 \leq i < k$. Note that, since a relationship path is an undirected path, but G is a directed graph, we allow either (w_i, w_{i+1}) or (w_{i+1}, w_i) to participate in the path. Alternatively, one may assume that each property p has an inverse, denoted “ˆ*p*”, using SPARQL notation.

To construct the dataset, we adopt a generic path finding and ranking process, briefly described as follow.

The path finding algorithm receives an RDF graph G, two *target entities*, v_{start} and v_{end}, a *maximum distance k*, and an *activation function* $\boldsymbol{\tau}$. It implements two breadth first searches (BFS), executed in parallel, to find paths in G between the target entities [7, 10, 14]. A BFS is started from each target entity (line 6). Subpaths are generated in the expansion step, and full paths are created when one of the target entities is reached, or the subpaths S_{left} or S_{right} share a common entity (line 7). An activation function $\boldsymbol{\tau}$ optimizes the traversal of G; only entities that comply with the activation criteria are considered. The output of the algorithm is a set of RDF paths between v_{start} and v_{end}.

The path ranking algorithm receives a set of paths *Paths* and a *path ranking function* $\boldsymbol{f}$, and outputs a ranked subset of *Paths*.

The final algorithm calls the path finding algorithm and then the path ranking algorithm. It outputs a ranked list of paths.

```
PathFinding($G$, $v_{start}$, $v_{end}$, $k$, $\tau$): Paths
Input: an entity pair $v_{start}$ and $v_{end}$
       a maximum distance $k$
       an activation function $\tau$
Output: a set of paths Paths that link
       the given pair of entities

1:  $expanding \leftarrow 0$, $Paths \leftarrow \emptyset$
2:  $side \leftarrow 0$, $left \leftarrow 0$ , $right \leftarrow 1$
3:  $S_{left} \leftarrow \{subpath(v_{start}, \text{null}, \text{null})\}$
4:  $S_{right} \leftarrow \{ subpath(v_{end}, \text{null}, \text{null})\}$
5:  repeat
6:    $S_{side} \leftarrow expand(S_{side}, \tau)$
7:    $Paths \leftarrow Join(S_{left}, S_{right}, v_{start}, v_{end},)$
8:    $expanding \leftarrow expanding + 1$
9:    $side = (side + 1) \bmod 2$
10: until $expanding <= k$
11: return $Paths$
```

```
ReferencePathList($G$, $v_{start}$, $v_{end}$, $k$, $\tau$, $f$): Paths
Input: an entity pair $v_{start}$ and $v_{end}$
       a maximum distance $k$
       an activation function $\tau$
       a path ranking measure $f$
Output: a set of paths Paths sorted

1: $Paths \leftarrow$ PathFinding($G$, $v_{start}$, $v_{end}$, $k$, $\tau$)
2: $Paths \leftarrow$ PathRanking($Paths$, $f$)
3: return $Paths$
```

4 Constructing the Entity Relatedness Test Dataset

The construction of the Entity Relatedness Test Dataset poses three major challenges: (1) how to select entity pairs; (2) how to find relationship paths for the entity pairs selected; and (3) how to rank the relationship paths. We addressed these challenges in the movies and music domains.

The dataset and resource are available at [17–21]. Examples and a more detailed evaluation of how use this dataset can be found in [16].

4.1 Selecting Entity Pairs

We focused on best-selling music artists[1], in the music domain, and on famous classic actors and actresses[2], in the movies domain. We considered the box office sales and the actor's fame as relevance criteria for the music and movies domains.

After selecting a list of entities from each of these two domains, we submitted each entity to Google Search to select a set of related entities. Then, for the possible entity pair, we computed their semantic connectivity score[3] [11] in DBpedia, with maximum length 4, to discover entity pairs with high connectivity. The maximum path length between two entities was set to 4, since it is a value backed up by the small world [12] phenomenon, which says that a pair of nodes is separated by a small number of connections, and since it was confirmed in previous experiments [15].

4.2 Finding Relationship Paths

For each of the 40 entity pairs of our dataset, we used the path finding algorithm, described in Sect. 3 (and introduced in [16]), to create 40 sets, each with 50 relationship paths. We applied the algorithm to the RDF graph of DBpedia, and used an activation function that prioritizes entities which are instances of classes of the DBpedia ontology that pertain to the domain in question. The classes or types of an entity in DBpedia are defined through the `rdf:type` property. The classes of the DBpedia ontology in music and movie domains are defined in Tables 7 and 8 in Sect. 5 at [16]. The entities that belong to previous classes are considered in the generations of relationship paths in DBpedia. The path finding algorithm uses as single activation function the classes of the DBpedia ontology in the domain concerned, the expansion process analyses the types of each entity, if an entity belongs to a class of the ontology domain, then it is prioritized to generate relationship paths.

To define which classes of the DBpedia ontology pertain to each of the domains in question, we adopted as reference the Music Ontology, for the music domain, and the Movie Ontology, for the movies domain. Then, we manually selected classes of the DBpedia ontology that could be paired with the major classes of each reference ontology.

[1] https://en.wikipedia.org/wiki/List_of_best-selling_music_artists.

[2] http://www.IMDb.com/list/ls000035399/.

[3] http://lod2.inf.puc-rio.br/scs/SemConnectivities.

4.3 Mapping Entities

As a preparation to the path ranking process, we mapped entities in DBpedia to entities in the reference datasets, as explained in this section.

Music Domain. To map DBpedia entities to last.fm, we used the keyword search API of last.fm[4]: `api:artist.getInfo`, `api:album.getInfo` and `api:track.getInfo`.

We first determined whether the entity represented an artist or a musical content by analyzing the `rdf:type` property, as in [6]. For example, the entity `dbr:Michael_Jackson` has type `dbo:Artist`. If the entity represented an artist, we extracted keywords from its URI (such as "`Michael + Jackson`") and submitted them to `api:artist.getInfo`[5] to search for the entity. If the search was successful, we had an exact mapping, otherwise we used other keywords. It the entity represented musical content (an album, song or single), we had to identify its main artist in DBpedia, through the property `dbp:artist`. For example, the main artist of `dbr:Thriller_(album)` is `dbr:Michael_Jackson`. If the entity represented a musical album, we called `api:album.getInfo`[6] to search for the entity. Similarly, it the entity represented a song or a single, we called `api:track.getInfo`.

Movies Domain. In DBpedia, we used the property `rdf:type` to decide if an entity was a movie. In any other case, we considered the entity as a participant of a movie. We identified the immediate type of an entity using the method proposed in [6].

To map DBpedia entities to IMDb, we imported the IMDb[7] database to a local PostgreSQL database and re-created data about names, movies and casts (people who worked in a movie). Usually, the entities in DBpedia have an auto description in the URI. For example, the URL `dbr:Cleopatra_(1963_film)` indicates the name of a movie, "`Cleopatra`", and its release year, "`1963`". We used this basic description to find the same entity in IMDb through classic SQL queries. For those cases where the queries returned more than one result, we used the Levenshtein Distance [13] to choose the IMDb entity most similar to the DBpedia entity.

4.4 Ranking the Relationship Paths

We ranked the paths in each of the 40 sets using semantic information extracted from IMDb and last.fm to compute entity ratings, and information extracted from DBpedia to compute property relevance scores.

To obtain the ranked lists, we first computed the *score* of each path π as the average of the *rating* of the entities involved in the path. Recall that π is a path in the DBpedia graph. Each entity e used in π was first mapped to an equivalent entity e' in IMDb or

[4] http://www.last.fm/api.

[5] http://ws.audioscrobbler.com/2.0/?api_key=YOUR_API_KEY&method=artist.getinfo&artist=Michael+Jackson.

[6] http://ws.audioscrobbler.com/2.0/?api_key=YOUR_API_KEY&method=album.getInfo&artist=Michael+Jackson&album=Thriller.

[7] http://www.imdb.com/interfaces.

last.fm, as explained in Sect. 4.3; the rating of *e'* was computed from data in IMDb or last.fm, as described below, and assigned to *e*. Finally, the score of π was computed as the average of the ratings of the entities that occur in π.

For each entity pair, we ranked the paths using their scores and retained the top 50 paths. However, since the path score ignores the relevance of the properties, paths that involve the same entities will have the same score. As a further step, we inspected each ranked list and used the *relevance scores* of the properties, computed in DBpedia, to help rank the paths with the same entities.

This ranking process is justified for two basic reasons. On one hand, we intended to create a dataset that would help evaluate approaches that address the entity relatedness problem, which typically involve a path ranking measure. Therefore, it would not be reasonable to adopt a path ranking measure from the literature (which would create ranked lists biased to that measure). On the other hand, it would be infeasible to manually rank the relationship paths that connect two entities (in DBpedia), whose number is typically very high [16]. Hence, we opted to: (1) select two domains – music and movies – for which specialized data were available; (2) filter the paths in DBpedia so that they traverse only entities in each of these domains; (3) use specialized domain data to pre-rank the paths found; (4) manually inspect and sanction the pre-ranking, which proved to be a feasible task. The computation of entity ratings and property relevance scores is detailed below.

Entity Rating in the Music Domain. In last.fm, each artist and musical content has two relevance scores: the listeners score and the play count score. This information can be accessed through the search API of last.fm. The listeners score represents the number of different users who listen a song, and the play count score is the number of times a person listens to a song. An album, depending on the number of songs, receives as play count score (or listener score) the sum of the play count scores (or listeners scores) of the songs in the album. Similarly, an artist receives a play count score and a listener score. We used the play count score to create an entity rating in the music domain; if the entity is not identified in the mapping, we assigned a zero score.

Entity Rating in the Movies Domain. IMDb publishes user-generated ratings for movies; an IMDb registered user can cast a vote (from 1 to 10) for every released movie in the database. Users can vote as many times as they want, but each vote will overwrite the previous one. In the case of people (actors, directors, writers) involved in a movie, we computed the average rating of the movies where the person participated to generate his/her rating. We imported the movies ratings to our local database and, with the table Cast, we related movies and actors to compute the artist rating. Again, if the entity is not identified in the mapping, we assign a zero score.

Property Relevance Score in DBpedia. We used the *inverse triple frequency* (ITF) [3] as the property relevance score, defined as $itf(p, G) = log\frac{|G|}{|G_p|}$, where $|G|$ is the number of triples in a knowledge base and $|G_p|$ is the number of triples in G whose property is p.

Example: Consider the following paths of the DBpedia RDF graph:

P₁. **Elizabeth_Taylor** ^producer **The_Taming_of_the_Shrew** starring **Richard_Burton**
P₂. **Elizabeth_Taylor** ^starring **The_Taming_of_the_Shrew** starring **Richard_Burton**

where "**Elizabeth_Taylor**", "**Richard_Burton**" and "**The_Taming_of_the_Shrew**" actually are abbreviations for the URIs of these DBpedia entities, and likewise for the properties.

The first step is to compute the entity rating of these entities using information from IMDb, which involves finding these DBpedia entities in IMDb. The path scores are computed as the average of the rating of the entities in the path. Since these two paths involve the same entities, they will have the same score. The second step is then to compute the ITF in DBpedia of the properties "^starring" and "^producer" to help disambiguate the ranking. Since "^producer" is less frequent in DBpedia than "^starring", it has a higher ITF. Path P_1 should then be ranked before P_2. However, this is subjected to manual inspection to confirm the preference of P_1 over P_2, which was the final decision in this case, on the grounds that P_1 is perhaps more informative to the user than P_2.

5 Conclusions and Future Work

In this paper, we described a dataset created to support the evaluation of approaches that address the entity relatedness problem. The dataset contains entity pairs in the movies and music domains, and lists of relationship paths in DBpedia, ranked based on information about their entities found in IMDb and last.fm, and on information about their properties computed from DBpedia.

The dataset can be used to test activation functions, based on entity similarity measures, and path ranking measures directly on the DBpedia graph. To use the dataset in the context of another knowledge base *K*, one should remap the entities and properties used in our reference dataset to *K*, much as we described in Sect. 4.

The construction process can be replicated to other domains where, intuitively: (1) entities with high reputation help select "meaningful" paths; (2) less frequent properties, or more discriminatory properties, also help select "meaningful" paths. In fact, the construction process described in Sect. 4 is as interesting as the resulting dataset. Therefore, as future work, we plan to focus on other domains, such as Sports, Video Games and Academic Publication, to increase the size of the Entity Relatedness Test Dataset described in the paper.

Acknowledgments. This work was partly funded by CNPq under grants 444976/2014-0, 303332/2013-1, 442338/2014-7 and 248743/2013-9 and by FAPERJ under grant E-26/201.337/2014 and E-26/010.000794/2016.

References

1. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: revealing relationships in RDF knowledge bases. In: Chua, T.-S., Kompatsiaris, Y., Mérialdo, B., Haas, W., Thallinger, G., Bailer, W. (eds.) SAMT 2009. LNCS, vol. 5887, pp. 182–187. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10543-2_21
2. Fang, L., Sarma, A.D., Yu, C., Bohannon, P.: REX: explaining relationships between entity pairs. PVLDB **5**(3), 241–252 (2011)
3. Pirrò, G.: Explaining and suggesting relatedness in knowledge graphs. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 622–639. Springer, Cham (2015). doi:10.1007/978-3-319-25007-6_36
4. Cheng, G., Zhang, Y., Qu, Y.: Explass: exploring associations between entities via top-*K* ontological patterns and facets. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8797, pp. 422–437. Springer, Cham (2014). doi:10.1007/978-3-319-11915-1_27
5. Mohan, Y., Bolin, D., Surajit, C., Chakrabarti, K.: Finding patterns in a knowledge base using keywords to compose table answers. PVLDB **7**(14), 1809–1820 (2014)
6. Herrera, J., Casanova, M.A., Nunes, B.P., Lopes, G.R., Leme, L.A.: DBpedia profiler tool: profiling the connectivity of entity pairs in DBpedia. In: IESD 2016 (2016)
7. Le, W., Li, F., Kementsietsidis, A., Duan, S.: Scalable keyword search on large RDF data. IEEE TKDE **26**(11), 2774–2788 (2014)
8. Hulpuş, I., Prangnawarat, N., Hayes, C.: Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 442–457. Springer, Cham (2015). doi:10.1007/978-3-319-25007-6_26
9. Färber, M., Ell, B., Menne, C., Rettinger, A.: A comparative survey of DBpedia, freebase, OpenCyc, Wikidata and YAGO. Semant. Web J. (1), 1–5 (2015)
10. De Vocht, L., Beecks, C., Verborgh, R., Mannens, E., Seidl, T., Van de Walle, R.: Effect of heuristics on serendipity in path-based storytelling with linked data. In: Yamamoto, S. (ed.) HIMI 2016. LNCS, vol. 9734, pp. 238–251. Springer, Cham (2016). doi:10.1007/978-3-319-40349-6_23
11. Nunes, B.P., Herrera, J., Taibi, D., Lopes, G.R., Casanova, M.A., Dietze, S.: SCS connector - quantifying and visualising semantic paths between entity pairs. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8798, pp. 461–466. Springer, Cham (2014). doi:10.1007/978-3-319-11955-7_67
12. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393** (6684), 440–442 (1998)
13. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. In: Soviet Physics Doklady, vol. 10, p. 707 (1966)
14. De Vocht, L., Coppens, S., Verborgh, R., Sande, M., Mannens, E., de Walle, R.: Discovering meaningful connections between resources in the web of data. In: LDOW CEUR-WS.org, vol. 996 (2013)
15. Nunes, B.P., Dietze, S., Casanova, M.A., Kawase, R., Fetahu, B., Nejdl, W.: Combining a co-occurrence-based and a semantic measure for entity linking. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 548–562. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38288-8_37
16. Herrera, J.: On the connectivity of entity pairs in knowledge bases. Ph.D. thesis, Department of Informatics, Pontifical Catholic University of Rio de Janeiro (2017). http://www-di.inf.puc-rio.br/~casanova/Publications/Dissertations-Theses/2017-Jose-Talavera.pdf
17. Herrera, J., Casanova, M.A., Nunes, B.P., Lopes, G.R., Leme, L.A.: Entity Relatedness Test Dataset.figshare (2017). https://doi.org/10.6084/m9.figshare.5234701

18. Herrera, J., Casanova, M.A., Nunes, B.P., Lopes, G.R., Leme, L.A.: RDF Version of the Entity Relatedness Test Dataset. GitHub (2017). https://github.com/lapaesleme/EntityRelatednessTestData
19. Herrera, J., Casanova, M.A., Nunes, B.P., Lopes, G.R., Leme, L.A.: SPARQL Endpoint of the Entity Relatedness Test Dataset (2017). http://swlab.ic.uff.br/fuseki/DatasetDescriptions/sparql
20. Herrera, J., Casanova, M.A., Nunes, B.P., Lopes, G.R., Leme, L.A.: Root Resources of the Entity Relatedness Test Dataset (2017). http://swlab.ic.uff.br/void.ttl#EntityrelatednessTestData_v3
21. Herrera, J., Casanova, M.A., Nunes, B.P., Lopes, G.R., Leme, L.A.: Source Code of the Entity Relatedness Test Dataset. GitHub (2017). https://github.com/jtherrera1/EntityRelatedness