dbrec — Music Recommendations Using DBpedia*

Alexandre Passant

Digital Enterprise Research Institute,
National University of Ireland, Galway
alexandre.passant@deri.org

Abstract. This paper describes the theoretical background and the implementation of dbrec, a music recommendation system built on top of DBpedia, offering recommendations for more than 39,000 bands and solo artists. We discuss the various challenges and lessons learnt while building it, providing relevant insights for people developing applications consuming Linked Data. Furthermore, we provide a user-centric evaluation of the system, notably by comparing it to last.fm.

Keywords: Semantic Web Applications, Linked Data, Recommendation Systems, Semantic Distance, DBpedia.

1 Introduction

Since its first steps in 2007, the Linking Open Data (LOD) cloud has grown considerably, as shown in Fig. 1[1]. However, besides recent initiatives outreaching how to build applications using it [5] [7], there is still room for more end-user applications (*i.e.* not semantic search engines nor APIs) that *consume* Linked Data. While we can argue that the data itself is the most valuable component, building innovative applications would lead to a virtuous circle enriching the value of this global network, by analogy with Metcalfe's law [11].

In this paper, we describe dbrec — `http://dbrec.net` —, a music recommendation system based on Linked Data (in particular on DBpedia) offering recommendations for more that 39,000 bands and solo artists. In addition, a core component of dbrec is its explanation feature, provided as a side effect of using Linked Data for computing the recommendations. We provide a user-centric evaluation of the system in order to identify how it compares to existing systems, in particular with last.fm[2], and how users rate its novel recommendations. Furthermore, besides presenting the theoretical background and the architecture of the system, we also discuss some lessons learnt when building it, in terms of data quality, architecture considerations as well as query patterns

* The work presented in this paper has been funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2).

[1] Based on [2] and `http://richard.cyganiak.de/2007/10/lod/`

[2] `http://last.fm`

P.F. Patel-Schneider et al.(Eds.): ISWC 2010, Part II, LNCS 6497, pp. 209–224, 2010.

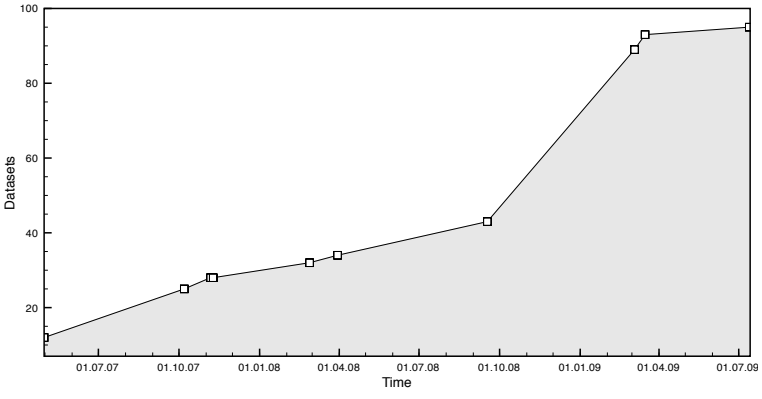

Fig. 1. The growth of datasets in the Linking Open Data cloud

and scalability. Thus, our aim is to provide a set of insights and best practices that can be re-used when building end-user applications that consume Linked Data.

The rest of the paper is organised as follows. In Section 2, we briefly describe the *LDSD* algorithm — *Linked Data Semantic Distance* —, used as a basis of our recommendation engine. In addition, we detail its related ontology, used to represent these distances and their explanations in RDF. In Section 3, we discuss the dbrec architecture, explaining how the previous algorithm has been applied to DBpedia to compute recommendations for more than 39,000 resources. Section 4 describes the evaluation of the system, including comparison with last.fm, evaluation of the novel recommendations and of the system as a whole. Then, in Section 5, we discuss three particular lessons learnt when building dbrec, which are however relevant in the broader Linked Data context: (1) data quality; (2) architecture considerations; and (3) SPARQL query patterns and scalability. We then discuss related work in Section 6 before concluding the paper with an overview of future challenges for dbrec.

2 Linked Data Semantic Distance

2.1 Motivation

Our main motivation was to identify how semantic distance [21] measures could be applied to resources published on the Web as Linked Data [1]. Specifically, and while semantic distance have been studied over time in various contexts [21] [3] [6], our goal was to identify and to apply such measures by considering some of the main characteristics of Linked Data:

- relying only on *links* — *i.e.* not taking into account literal values and their linguistic proximity;

- relying only on *instance data* — *i.e.* not taking into account ontologies used to describe resources, since LOD is more oriented towards publishing instance data than using formal ontologies;
- considering *dereferencable URIs* — so that distances can be computed simply by accessing URIs and retrieving corresponding RDF data.

Our aim was then to identify the usefulness of the Linked Data principles for computing semantic distance between particular resources.

2.2 A Conceptual Model for Linked Data

While Linked Data is generally introduced using its four publishing principles [1] — which make sense from a programmatic point of view — there is a need to ground it into a theoretical framework to define algorithms using it. We thus provide the following definition of a Linked Data dataset, whether it is centralised or distributed on the Web — and we can then consider $LOD = \bigcup_i G_i$.

Definition 1. *A dataset following the Linked Data principles is a graph G such as $G = (R, L, I)$ in which $R = \{r_1, r_2, ..., r_n\}$ is a set of resources — identified by their URI —, $L = \{l_1, l_2, ..., l_n\}$ is a set of typed links — identified by their URI — and $I = \{i_1, i_2, ..., i_n\}$ is a set of instances of these links between resources, such as $i_i = \langle l_j, r_a, r_b \rangle$.*

This definition voluntary excludes literals, as we focused only on the URI-linking aspect of Linked Data, as discussed in [1]: *"The simplest way to make linked data is to use, in one file, a URI which points into another".*

2.3 LDSD — Linked Data Semantic Distance

Based on this definition, we defined a *Linked Data Semantic Distance* (LDSD) measure to compute the distance between two resources published as Linked Data[3], normalised in the $[0, 1]$ interval. So far, our measure considers only resources linked either directly or through a third resource, and recursive patterns such as SimRank [15] may be used in the future. Since LDSD — and some of its initial variants — has already been discussed in [19], we will not present it in too many details. At a glance, for two resources r_a and r_b, LDSD identifies four dimensions (direct and indirect links, both incoming and outcoming) to compute their distance, using the following definitions.

Definition 2. *C_d is a function that computes the number of* direct and distinct *links between resources in a graph G. $C_d(l_i, r_a, r_b)$ equals 1 if there is an instance of l_i from resource r_a to resource r_b, 0 if not. By extension C_d can be used to compute (1) the total number of direct and distinct links from r_a to r_b*

[3] Note that we use the term *distance* while the measure may actually not be symmetric.

($C_d(n, r_a, r_b)$) as well as (2) the total number of distinct instances of the link l_i from r_a to any node ($C_d(l_i, r_a, n)$).

Definition 3. *C_{io} and C_{ii} are functions that compute the number of* indirect and distinct *links, both* outcoming and incoming*, between resources in a graph G. $C_{io}(l_i, r_a, r_b)$ equals 1 if there is a resource n that satisfy both $\langle l_i, r_a, n\rangle$ and $\langle l_i, r_b, n\rangle$, 0 if not. $C_{ii}(l_i, r_a, r_b)$ equals 1 if there is a resource n that satisfy both $\langle l_i, n, r_a\rangle$ and $\langle l_i, n, r_b\rangle$, 0 if not. By extension C_{io} and C_{ii} can be used to compute (1) the total number of indirect and distinct links between r_a and r_b ($C_{io}(n, r_a, r_b)$ and $C_{ii}(n, r_a, r_b)$, respectively outcoming and incoming) as well as (2) the total number of resources n linked indirectly to r_a via l_i ($C_{io}(l_i, r_a, n)$ and $C_{ii}(l_i, r_a, n)$, respectively outcoming and incoming)*

$$LDSD(r_a, r_b) = \frac{1}{1 + \sum_i \frac{C_d(l_i, r_a, r_b)}{1+log(C_d(l_i, r_a, n))} + \sum_i \frac{C_d(l_i, r_b, r_a)}{1+log(C_d(l_i, r_b, n))} + \sum_i \frac{C_{ii}(l_i, r_a, r_b)}{1+log(C_{ii}(l_i, r_a, n))} + \sum_i \frac{C_{io}(l_i, r_a, r_b)}{1+log(C_{io}(l_i, r_a, n))}}$$

Fig. 2. The *LDSD* measure

2.4 The LDSD Ontology

In addition to the measure itself, and since we focus on a Linked Data approach, our aim was to provide the output of such measures also available on the Web as Linked Data. We thus designed a lightweight LDSD ontology[4], accompanying the previous measure and containing two main classes:

- `ldsd:Distance`, in order to represent the distance between two resources (using `ldsd:from` and `ldsd:to`) and its value (`ldsd:value`[5])
- `ldsd:Explanation` (and four subclasses: `ldsd:DirectIn`, `ldsd:DirectOut`, `ldsd:IndirectIn` and `ldsd:IndirectOut`), in order to store the links and the property-value pairs (`ldsd:property` and `ldsd:node`) used to measure the distance, and how much similar links appear in the dataset (`ldsd:total`).

Here lies one of the first advantages of using Linked Data to compute semantic distance. The links that are traversed by the algorithm are all typed, and this is consequently easy to know how the distance has been computed, as we will show when presenting dbrec's user-interface (Section 3.4). As an example, the following snippet of code (Listing 1.1) represents that Elvis Presley is at a distance of 0.09 from Johnny Cash, because (among others) both have the same value for their `rdf:type` property (`http://dbpedia.org/class/yago/SunRecordsArtists`), shared only by 19 artists in the `http://dbpedia.org` dataset.

[4] Available at `http://dbrec.net/ldsd/ns#`

[5] `rdf:value` was not used due to its lack of formalism — `http://www.w3.org/TR/rdf-schema/#ch_value`

```
@prefix ldsd: <http://dbrec.net/ldsd/ns#> .
<http://dbrec.net/distance/774a32aa-dede-11de-84a3
    -0011251e3563> a ldsd:Distance ;
   ldsd:from <http://dbpedia.org/resource/Johnny_Cash> ;
   ldsd:to <http://dbpedia.org/resource/Elvis_Presley> ;
   ldsd:value "0.0977874534544" .
<http://dbrec.net/distance/774a32aa-dede-11de-84a3
    -0011251e3563> ldsd:explain [
   a ldsd:IndirectOut ;
   ldsd:property <http://www.w3.org/1999/02/22-rdf-syntax
       -ns#type> ;
   ldsd:node <http://dbpedia.org/class/yago/
       SunRecordsArtists> ;
   ldsd:total "19" ] .
```

Listing 1.1. Representing distance between Johnny Cash and Elvis Presley.

3 The dbrec Recommendation System

3.1 System Architecture

Based on our previous findings, we implemented a music recommendation system in order to demonstrate the usability of the *LDSD* measure for an end-user application. To do so, we computed semantic distance for all artists referenced in DBpedia. While it does not involve cross-datasets recommendations, which are possible using our algorithm, it however offers two main advantages. First, there are more than 39,000 artists available in DBpedia for which recommendations can be built. Second, DBpedia also provides pictures and description of artists that can be used to build the system's user interface.

In order to build the system, we followed four steps (Fig. 3): (1) identify the relevant subset from DBpedia; (2) reduce the dataset for query optimisation; (3) compute distances using the *LDSD* algorithm and represent them using its ontology; (4) build a user-interface for browsing recommendations.

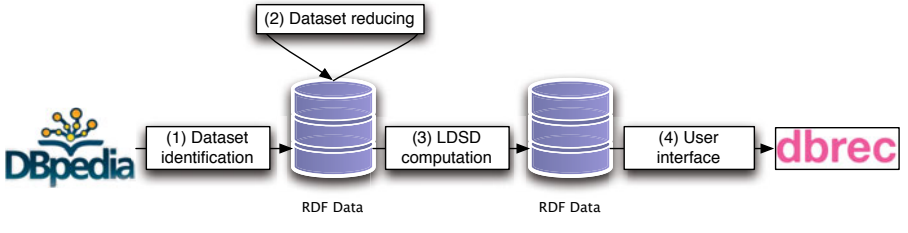

Fig. 3. The dbrec architecture

3.2 Identifying the Relevant Dataset from DBpedia

While the LDSD algorithm can be simply translated to SPARQL queries and applied to any public endpoint, this approach has some drawbacks. Indeed, DBpedia's public endpoint is limited to a certain number of answers per query, so each query must be split in sub-queries, and results must then be recomposed.

Consequently, we setup our own replica of the dataset to compute the recommendations locally. Instead of relying on a complete DBpedia dump, and as we aim at building music recommendations only, we limited ourselves to all instances of `dbpedia:MusicalArtist` and `dbpedia:Band` from DBpedia. In addition, according to the *LDSD* algorithm, we needed both incoming and outcoming links for each artist. Fortunately, each data file in DBpedia (retrieved when dereferencing the resource URI with the proper HTTP header) provides this information, for both incoming and outcoming links. This also means that the distance could be measured live, by dereferencing URIs of relevant resources, while it would obviously be more time consuming.

The original dataset, including more than 39,000 resources, included 3,004,351 triples. We then cleaned it to get a smaller and more accurate dataset, for two main reasons. On the one hand, we wanted to remove datatype properties, as they are not relevant for our experiment[6]. Removing them lead to a dataset containing 2,247,019 triples, thus reducing the original one from about 25.2% — implying that 1/4 of DBpedia assertions, in our dataset, involve literals. On the other hand, we identified lots of redundancy and inconsistencies in our DBpedia subset[7]. Especially, many links between resources are defined redundantly as `http://dbpedia.org/ontology/xxx` and at the same time as `http://dbpedia.org/property/xxx`. We then removed duplicates, leading to 1,675,711 triples, *i.e.* only 55.7% of the original dataset.

We also analysed the dataset to identify how artists are related to each other (by direct links) in DBpedia (Fig. 4). We observed that 21,211 of them (more than 50%) are not linked to any artists, and 9,555 are linked to three of them, the maximum being 14 links from one artist to 14 others. Then, by using indirect links for computing semantic distance with *LDSD*, we are able to provide recommendations for these 21,211 isolated artists.

3.3 Cleaning and Reducing the Dataset

While being optimised in the previous step, the computation time was still far from optimum. Even for a recommendation time of 40 seconds per artist (see Section 5), it would have taken 15 days to compute the whole recommendations dataset. We then focused on further optimisations not at the query-engine level, but at the dataset level, analysing it more deeply, and we identified that: (1) 188 distinct properties are used to link artists together directly; (2) 578 distinct properties are used to link an artist to any resource (including artists) ; (3) 767

[6] We agree that using and comparing literals may help in the distance measurement, but our focus was to consider only a link-based approach.

[7] We relied on DBpedia 3.3.

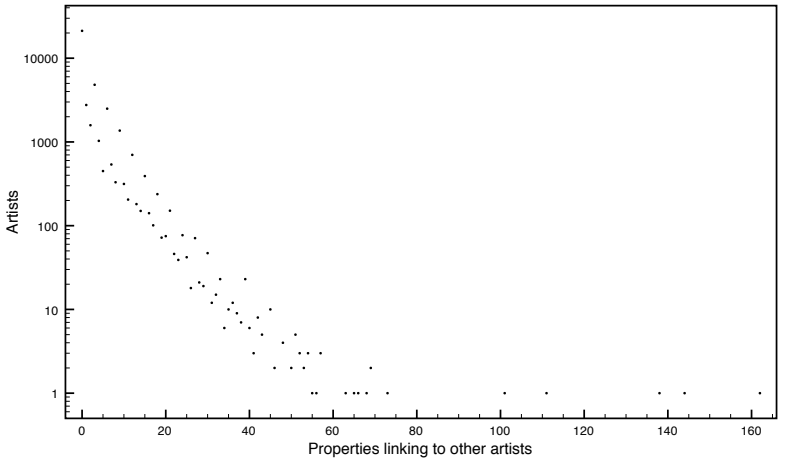

Fig. 4. Distribution of properties between artists in DBpedia

distinct properties are used to link any resource (including artists) to an artist. We then focused on data curation: (1) on the one hand to remove properties and property-values that are useless for computing the LDSD measures, and (2) on the other hand, to solve some data quality issues in DBpedia.

From the 188 properties linking two artists, we identified that 18 were used as links between artists while it was not their main purpose[8], such as the property `dbprop:notableInstruments` — used to link an artist to its instrument(s) — or `dbprop:nationalAnthem` — linking a country to its anthem. Moreover, we identified 35 properties that were wrongly defined — while however used two times of less —, such as `http://dbpedia.org/property/extra18` and `http://dbpedia.org/property/klfsgProperty`. Then, from the 578 properties used to link artists to resources, 183 were used only one time and were consequently useless for our recommendations, since it imply there is no more than one artist using on. In addition, 36 of these properties were wrongly defined. Furthermore, we identified 11 useless property-value combinations to compute our recommendations, by being too generic such as `rdf:type foaf:Agent`. Finally, from the 767 properties used to link any resource to an artist, 336 were removed as used only to link to a single artist, and 115 were wrongly written.

We then cleaned-up the dataset and reduced it to a total of 1,073,077 triples. We eventually ran LDSD on this dataset. The computation time took a total of 9,797 minutes[9], and resulted in 50,753,494 new triples describing the recommendations (and the explanations) modelled using the previous LDSD ontology. The time to compute the recommendation for a single resource obviously depended on the artist and related properties, as one can see in Fig. 5[10]. In addition,

[8] At least from what their general usage on DBpedia can tell, since they do not have any domain or range.

[9] On a 2 x AMD Opteron 250 with 4GB memory running Ubuntu 8.10/x86_64.

[10] Average time of 5 consecutive runs.

Artist	Time (sec.)
Ramones	25.20
Johnny Cash	61.16
U2	50.06
The Clash	43.34
Bar Religion	34.98
The Aggrolites	7.35
Janis Joplin	23.12

Fig. 5. Computation time of recommendations for various artists

Artist	Distance
Elvis Presley	0.0977874534544
June Carter Cash	0.105646049225
Willie Nelson	0.13221654708
Kris Kristofferson	0.140717564665
Bob Dylan	0.146635674481
Marty Robbins	0.167300943904
Rosanne Cash	0.17826142135
Charlie McCoy	0.183656756953
Gene Autry	0.191014026051
Carl Smith	0.198003626307

Fig. 6. 10 first recommendations for Johnny Cash using *LDSD*

Fig. 6 displays the result of the computation (distance only, 10 first results) for `dbpedia:Johnny_Cash`.

3.4 User-Interface

Thanks to the use of Linked Open Data, building the user-interface was quite straightforward. As each artist and band is identified by a reference URI, abstracts and pictures can be obtained by simply dereferencing it. We then build a front-end providing recommendation (ranked by distance) for any of the 39,000

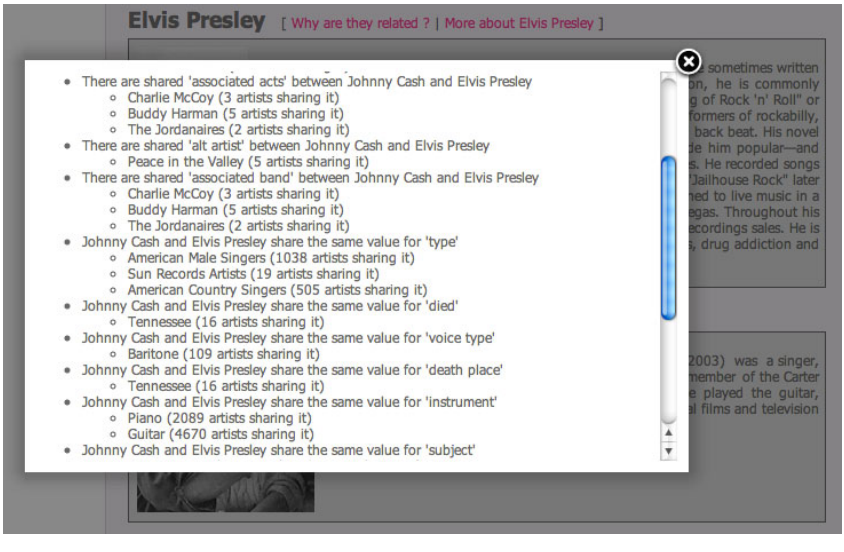

Fig. 7. Example of dbrec explanations

artists and bands of our dataset, including related pictures and abstracts. Recommendation pages are rendered *via* SPARQL queries ran over the computed LDSD data, and pages also provide links to YouTube videos, Twitter messages and last.fm profiles in order to enhance the browsing experience and let users listen to related songs. In addition, in order to let developers build third-party applications on top of dbrec, recommendations are also available as RDFa using the LDSD ontology.

Explanation are provided on demand, through a *"Why are they related?"* link that opens a pop-up launching another query to retrieve the explanations. These explanations are provided using human-readable labels of the property and their values, as seen in Fig. 7, explaining the recommendation of Elvis Presley for one user browsing the page about Johnny Cash.

4 Evaluation

4.1 Context of the Evaluation

In order to evaluate dbrec, we focused on standard user evaluations protocols for recommender systems, both off-line an on-line [12]. We interviewed 10 participants: 2 women and 8 men, ages ranging from 24 to 34. Interviews were conducted face to face (besides one that has to be done by phone) and last between 35 and 55 minutes. Before the evaluation, we asked users to submit a list of 10 to 15 bands they listen to and appreciate, from which we randomly selected 10 bands (ensuring that all belong to the dbrec dataset).

Then, the interviews involved two main steps. First, an *off-line evaluation*: for five (randomly chosen) bands from the previous list, we provided users with two sets of ten recommendations each. One was generated from dbrec, the other one from last.fm. Users were not aware of this and were just given the two lists randomly, simply telling them they came from different systems. We asked interviewees to rate to each recommendation (from 1 — poor — to 5 — excellent —) or to mention if that was an unknown recommendation. Note that we asked them to rank the relevance of the recommendations, not if they like that particular band or artist.

Then, we conducted an *on-line evaluation* for the five remaining bands. Users sat in front of the system and were asked to browse the recommendation list and to rate the first 10 recommendations. However, as opposed to the first part, when a band was unknown, users could read the description of each recommended artist and check the explanations provided by dbrec. We also told users that checking the explanations was not mandatory, as we wanted to observe how often they use it or not[11].

4.2 dbrec versus Last.fm

Regarding the off-line evaluation, the average mark for dbrec recommendations was $3.37(\pm 1.19)$ (and $3.44(\pm 1.25)$ when combined with the results from the

[11] For the phone interview, we asked the user to tell us if he was using them, since we were not able to setup a screen-sharing teleconference.

Table 1. Precision of recommendations: dbrec *versus* last.fm

	dbrec (off-line only)	**dbrec (off-line and on-line)**	**last.fm**
t=2	92.05	90.59	98.32
t=3	76.63	77.72	87.91
t=4	49.06	51.23	58.05
t=5	20.09	25	25.165

on-line part), while the average mark for last.fm was 3.69(±1.01). We also evaluated the precision of both recommendations, considering the number of relevant items provided in both lists. To do so, we considered different threshold in the multi-point scale used to evaluate the recommendations. Table 1 shows the different values for both dbrec and last.fm (t=x means that we consider a recommendation as being relevant it it is ranked x or higher). In spite of a slight advantage for last.fm, dbrec achieves a reasonable score, especially considering that it does not use any collaborative filtering approach, and relies only on links between resources. Measuring the recall of recommendation was however not possible, as it would have implied users to know and check all bands of the dbrec dataset.

4.3 Evaluating Novel Recommendations

An interesting outcome of the evaluation was that many recommendations were unknown to users: 62% for dbrec (59.6% when combining off-line and on-line parts) and 40.4% for last.fm. However, as argued by [4]: *"novel recommendations are sometimes necessary in order to improve the users experience and discovery in the recommendation workflow"*. To that end, we used the on-line setup to evaluate quality of the novel recommendations provided by dbrec. For that on-line part, 310 recommendations (on a total of 500[12]) were identified as unknown, *i.e.* being novel. Among these 310, 274 have been evaluated. One user justified that, without listening to the music and even with the explanations, he was not able to provide any mark for them, while other users were able to do so, judging the recommendations by reading descriptions and explanations. Among these 274 remaining recommendations, the average rate for novel recommendations was 3.05(± 1.09). In [4], the authors also showed that, based on user-centric evaluation, the average mark for novel recommendations was less than 3[13] and argued *"this probably emphasises the need for adding more context when recommending unknown music. Users might want to understand why a song was recommended"*. We hence believe than the features provided by dbrec, namely the description of each recommendation and most of all its explanation, made users better understand and appreciate the recommendations — and consequently put this average mark higher.

Furthermore, we also evaluated the precision of these recommendations, considering various thresholds as previously (Table 2). We the observed than even

[12] 10 *users* × 5 *bands* × 10 *recommendations*.

[13] Respectively 3.03(±1.19) for Collaborative Filtering, 2.77(±1.20) for Hybrid and 2.57(±1.19) for Audio Content-Based recommendations.

Table 2. Precision for novel recommendations on dbrec

Precision			
t=2	**t=3**	**t=4**	**t=5**
89.42	70.80	37.59	7.3

with a threshold of 3 (*i.e.* only good, very good or excellent recommendation), the precision is more than 70%, while still more than 37% considering only very good or excellent ones.

Overall, in terms of recommendations, dbrec achieves respectable performances comparable to last.fm and to other systems. However, instead of relying on collaborative-filtering algorithms (based on proprietary data from million of users), it only requires a set of publicly available open-data. This clearly shows the advantage of the Linking Open Data initiative for building such recommender systems.

4.4 Evaluating the UI and the Explanations

Finally, in addition to the recommendations themselves, we asked users to agree (or not) on a set of adjectives describing (1) the system and its user-interface in general, and (2) the explanations in particular. As results show (Table 3), all users positively acknowledged both the system and its explanations. There are however efforts to be made regarding the explanations and their related presentation, still considered as "Too geeky" by six users.

Furthermore, we observed that users relied on explanations for 198 of the 310 unknown recommendations. In addition, they relied on it for 24 of the 190 known recommendations, wanting to understand the reason of the recommendation, as discussed in [16].

Table 3. User-feedback on the overall dbrec system

	User-interface	**Explanations**
Enjoyable	9	7
Useful	9	9
Enriching	8	10
Easy to use	10	9
Confusing	0	2
Complicated	0	2
Too geeky	1	6

5 Lessons Learns and Discussions

5.1 Data Quality

A first lesson learnt concerns the data quality within the LOD cloud, which is far from perfect to build applications using it. As exposed in Section 3.3, we had to

rely on (manual) curation of the dataset, and identified issues with the underlying data model, such as similar properties defines at both `/property` and `/ontology` URLs in DBpedia, or many having neither domain nor range, making difficult to identify inconsistencies. While we focused only on the DBpedia dataset, similar observations have been identified more globally on the Web, implying a need for more data curation in the LOD cloud [13].

5.2 Use, but Replicate

Then, while data is openly available on the Web, and while some services provide public SPARQL endpoints (such as DBpedia), local mirroring is required to ensure scalability and efficiency in the development process. For example, due to results restrictions on the public DBpedia endpoint, simply retrieving all bands and artists from DBpedia implies to (1) get the number n of results satisfying that pattern (which furthermore relies on `COUNT`, not supported by SPARQL 1.0); (2) split the query into $\lceil n/5000 \rceil$ queries using the `LIMIT` and `OFFSET` clauses; (3) run the queries and recompose the results, while also taking care of network issues that may break that loop. Then, we had to replicate data in a local store, which conforms to what [10] discussed, by proposing a reference architecture for Semantic Web applications based on empirical analysis of existing services. Such solutions however raise the issue of synchronising datasets between original services and local repositories, but also shows business opportunities for Linking Open Data services providers that could deploy commercial SPARQL capabilities with enhanced quality of service.

5.3 SPARQL: Be Quick or Be Neat

Another lesson learnt concerns the use of SPARQL, where we observed that decomposing queries provides much faster answering time than running single queries covering complex paths.

For example, in order to translate *LDSD* to SPARQL queries, one of our need was to identify, from a resource r_i, all resources r_j that are linked to a third resource r_x through the same path as r_i is linked to r_x — that is $\langle l_i, r_i, r_x \rangle, \langle l_j, r_j, r_x \rangle$ — looking for resource sharing a common property-value. In addition, we had to ensure that r_x was also either a band or a solo artists. To do so, we considered three different options:

1. running a single query covering the full pattern, thus retrieving at the same time all the property-value pairs, as well as the corresponding resources;
2. running a first SPARQL query to identify all the property associated to r_i, and then identifying all resources sharing a property (plus its value) (*Property-slicing*);
3. running a first SPARQL query to identify all the property-values related to r_i, and then identifying all resources sharing that property-value pair (*Complete-slicing*).

As Table 4 shows, while up to 135 queries were needed when we initially needed only one, the computation time was up to 75% shorter when using the

Table 4. Comparing strategies to identify indirectly related artists

	Direct-SPARQL		**Property-Slicing**		**Complete-Slicing**	
	queries	time	queries	time	queries	time
Ramones	1	139.97	20	109.51	66	37.84
Johnny Cash	1	257.81	30	152.60	135	75.35
U2	1	155.53	22	122.91	70	44.03
The Clash	1	146.43	20	110.84	79	42.61
Bad Religion	1	104.08	23	86.49	97	47.35
The Aggrolites	1	145.92	13	114.52	28	28.33
Janis Joplin	1	230.88	27	151.00	98	62.81

complete-slicing approach[14]. This means that optimisation must be done by the query authors, as writing extensive queries for a complex graph-matching is not yet the best solution regarding scalability. Further work should probably to be done to optimise complex SPARQL query processing and decomposition of patterns [23], so that developers could write single queries instead of relying on decomposition and recomposition of results through external scripts.

6 Related Work

In the realm of large-scale Semantic Web based recommender systems, the most know approach is probably the FOAFing-the-music project [18], that uses the distributed social networking capabilities of FOAF to provide music recommendations based on users' and friends' tastes. Focusing on a similar idea of cross-social-networking recommendations, [20] presented some ways to use FOAF, SIOC and MOAT to compute recommendations and also discussed some first steps on using Linked Data to build explanatory recommender systems. More recently, [9] followed a similar idea by developing a first prototype applied to cross-sites collaborative filtering using Linked Data. However, as discussed in introduction, our motivation was to rely on Linked Data from a resource-centric point of view, not considering social aspects but only links between resources. In that context, LODations recently focused on LOD-based music recommendations[15], while using a simpler approach not ranking the recommendations nor combining multiple features automatically. Furthermore, [17] also focused on the use of ontologies for recommender systems.

Regarding ontologies and data modelling, extensive work has been done around the Music Ontology [22]. This also includes MuSim[16] — The Music Similarity Ontology [14] — that could be mapped to our LDSD ontology in the future.

[14] While we ran these tests only with our local endpoint, using 4store, we observed that the initial full-query time-outs on the public DBpedia endpoint while other strategies ran properly, albeit the query limit that we mentioned earlier, and similar timing issues.

[15] `http://lodations.heroku.com/`

[16] `http://grasstunes.net/ontology/musim/musim.html`

More recently, a Recommendation Ontology has also been proposed[17], and we may also consider alignment with our model and SCOVO — the Statistical Core Vocabulary [8] — to represent statistic information about the explanations of the recommendations.

7 Future Work

In terms of future work, we first plan to investigate additional criteria to tune the distance measurement. It could include using the transitivity of genres, defined as `skos:Concepts` and hierarchically ordered in DBpedia. We may also investigate link propagation and recursivity, as done by SimRank, in order to recommend artists that are more than one node away of the seed one.

Moreover, feature selection is also an issue that needs to be tackled. Indeed, we identified that geolocation properties are often used for recommendation, but not always relevant. This is especially a problem for bands having a poor description in DBpedia, especially non-international ones where often, the only property besides their genre is their location. Then, it makes recommendation based mostly on the genre and location, which is often not relevant enough. We could have imagined excluding or weighting geolocation properties, but the issue is actually more complex. For instance, for a pop-band, being from Washington or San Francisco is probably not relevant. However, for a punk-hardcore one, this makes a lot of sense since the two scenes are radically different, and someone enjoying east-cost punk-hardcore may not listen to west-coast one. However, this would probably require manual classification of such graph patterns, in order to identify their relevance or not in certain contexts.

In addition, while currently limited to DBpedia, we aim at integration further sources of information (Freebase, MusicBrainz, etc.) to compute the recommendations, making the system targeted towards a wider Linking Open Data perspective.

8 Conclusion

In this paper, we discussed how semantic distance measures can be applied to Linked Data, and how they can be used to build music recommendation systems. We provided an algorithm to enable such measures on any Linked Data dataset, and an ontology to represent the distances and their explanations.

In addition, we have build dbrec, a recommendation system using DBpedia and providing open and explanatory recommendations for more than 39,000 bands and solo artists. The system was evaluated with use-centric evaluation, both off-line and on-line. We showed how it competes with last.fm, in addition of providing relevant novel recommendations, while relying only on public and open data, and not of listening behaviours of a large user set.

[17] `http://smiy.sourceforge.net/rec/spec/recommendationontology.html`

Finally, more than the distance measurement and the application, we discussed some set of lessons learnt from building the system, in terms of data quality, architectures for Semantic Web applications and optimisation of SPARQL queries. We hope that such lessons could be useful for implementers and provide some useful insights for anyone building applications consuming Linked Data circa'2010.

References

1. Berners-Lee, T.: Linked Data. Design Issues for the World Wide Web, World Wide Web Consortium (2006), `http://www.w3.org/DesignIssues/LinkedData.html`
2. Bizer, C., Heath, T., Ayers, D., Raimond, Y.: Interlinking Open Data on the Web. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519. Springer, Heidelberg (2007)
3. Budanitsky, E., Hirst, G.: Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In: Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources (2001)
4. Celma, Ò., Herrera, P.: A new approach to evaluating novel recommendations. In: RecSys 2008: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 179–186. ACM, New York (2008)
5. Corlosquet, S., Delbru, R., Clark, T., Polleres, A., Decker, S.: Produce and Consume Linked Data with Drupal! In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 763–778. Springer, Heidelberg (2009)
6. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Berlin (2007)
7. Hausenblas, M.: Exploiting linked data to build web applications. IEEE Internet Computing 13(4), 68–73 (2009)
8. Hausenblas, M., Halb, W., Raimond, Y., Feigenbaum, L., Ayers, D.: SCOVO: Using Statistics on the Web of Data. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 708–722. Springer, Heidelberg (2009)
9. Heitmann, B., Hayes, C.: Using Linked Data to build open, collaborative recommender systems. In: Linked AI: AAAI Spring Symposium "Linked Data Meets Artificial Intelligence", AIII (2010)
10. Heitmann, B., Kinsella, S., Hayes, C., Decker, S.: Implementing Semantic Web applications: reference architecture and challenges. In: Proceedings of the 5th Workshop on Semantic Web Enabled Software Engineering. CEUR Workshop Proceedings, vol. 524, CEUR-ws.org (2009)
11. Hendler, J.A., Golbeck, J.: Metcalfe's law, Web 2.0, and the Semantic Web. Journal of Web Semantics 6(1), 14–20 (2008)
12. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems (TOIS) 22(1), 5–53 (2004)
13. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the Pedantic Web. In: 3rd International Workshop on Linked Data on the Web (LDOW2010) at WWW 2010. CEUR Workshop Proceedings, vol. 628, CEUR-ws.org (2010)
14. Jacobson, K., Raimond, Y., Sandler, M.: An Ecosystem for Transparent Music Similarity in an Open World. In: International Symposium on Music Information Retrieval (2009)

15. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: KDD 2002: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543. ACM, New York (2002)
16. McSherry, D.: Explanation in Recommender Systems. Artificial Intelligence Review 24(2), 179–197 (2005)
17. Middleton, S.E., Alani, H., De Roure, D.: Exploiting Synergy Between Ontologies and Recommender Systems. CoRR, cs.LG/0204012 (2002)
18. Celma, Ò., Ramirez, M., Herrera, P.: Foafing the music: A music recommendation system based on RSS feeds and user preference. In: Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR (2005)
19. Passant, A.: Measuring Semantic Distance on Linking Data and Using it for Resources Recommendations. In: Linked AI: AAAI Spring Symposium "Linked Data Meets Artificial Intelligence", AIII (2010)
20. Passant, A., Raimond, Y.: Combining Social Music and Semantic Web for Music-related Recommender Systems. In: Proceedings of the First Workshop on Social Data on the Web (SDoW2008). CEUR Workshop Proceedings, vol. 405, CEUR-ws.org (2008)
21. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man and Cybernetics 19, 17–30 (1989)
22. Raimond, Y., Abdallah, S., Sandler, M., Giasson, F.: The Music Ontology. In: International Conference on Music Information Retrieval, pp. 417–422 (September 2007)
23. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: WWW 2008: Proceeding of the 17th International Conference on World Wide Web, pp. 595–604. ACM, New York (2008)