Personalised Access to Linked Data

Milan Dojchinovski and Tomas Vitvar

Web Intelligence Research Group
Faculty of Information Technology
Czech Technical University in Prague
firstname.lastname@fit.cvut.cz

Abstract. Recent efforts in the Semantic Web community have been primarily focused on developing technical infrastructure and technologies for efficient Linked Data acquisition, publishing and interlinking. Nevertheless, due to the huge and diverse amount of information, the actual access to a piece of information in the LOD cloud still demands significant amount of effort. In this paper, we present a novel configurable method for personalised access to Linked Data. The method recommends resources of interest from users with similar tastes. To measure the similarity between the users we introduce a novel resource semantic similarity metric, which takes into account the *commonalities* and *informativeness* of the resources. We validate and evaluate the method on a real-world dataset from the Web services domain. The results show that our method outperforms the other baseline methods in terms of accuracy, serendipity and diversity.

Keywords: personalisation, recommendation, Linked Data, semantic distance, similarity metric.

1 Introduction

In the past years, the Semantic Web community has been primarily focused on developing technical infrastructure and technologies to make the Web of Data feasible [3]. Consequently, these efforts led the development of various methods for Linked Data acquisition, publishing and interlinking, which gave birth to $1,091$ Linked Datasets (as of April 2014 [11]), which is an overall growth of 271% compared to only 294 datasets published in September 2011. Along with these efforts, many end-user applications that *consume* and *deliver* Linked Data have been developed. Between the most studied applications which leverage Linked Data are recommender systems. In a nutshell, Linked Data based recommender systems produce recommendations of Linked Data resources representing items of interest. To predict the resources of interest, they exploit the relations and interactions of the users with the resources. The problem of recommendation of Linked Data resources has been addressed in several recent methods [9,10,14,1,7,5,8]. However, proposed methods are primarily developed for a specific domain, they require manual pre-processing of the datasets and

K. Janowicz et al. (Eds.): EKAW 2014, LNAI 8876, pp. 121–136, 2014.

they can hardly be adapted to new datasets. Thus, there is a need of new sophisticated methods which will be enough robust to process Linked Data from different domains and provide accurate, while at the same time serendipitous and diverse Linked Data resource recommendations.

In this paper, we present a method for personalised access to Linked Data. The method recommends resources of interest for a user. It relies on the assumption that *if a person A and person B have interest in similar resources, then person A is likely to have interest in similar resources in the future, as person B.* To predict resources of interest, the method first measures the similarity between resources representing users, and then recommends resources from similar users. To measure the similarity between two resources in a Linked Data dataset, we propose a novel similarity measure which primarily relies on the shared information of the resources in an RDF graph. The similarity of the resources we compute based on their shared context (i.e., overlap of the surrounding RDF sub-graphs), which we call *resource context graphs.* When computing the similarity of the resources, our method takes into consideration 1) *the size of shared context*–the amount of common resources, 2) *the connectivity of each shared resource*–how well are the context resources connected with the users' resources, and 3) *the informativeness of each shared resource*–less probable resources are considered to be more specific, and consequently more informative than the more common ones. The resources' informativeness is primarily incorporated to differentiate informative shared resources from non-informative, such as resources of type *owl:Thing* or *skos:Concept.* A prototype of the method was implemented on top of the neo4j[1] graph database and we show a resource recommendation use case in the Web services domain. We evaluate the method on several experiments showing that the method produces highly accurate, serendipitous and diverse recommendations compared to the traditional recommendation techniques. For the evaluation we use a real-world dataset, the Linked Web APIs dataset, which is an RDF representation of the ProgrammableWeb[2], the largest Web service and mashup repository.

The reminder of the paper is as follows. Section 2 describes the method, its definitions and algorithms. Section 3 describes several experiments we run to validate and evaluate the method. Section 4 reports on existing methods that relate to ours. Section 5 provide discussions and future directions. Finally, Section 6 concludes the paper.

2 Personalised Resource Recommendation

We formulate the problem of personalised recommendation of resources as problem of ranking and recommending top-N most relevant resources. We base our method on the collaborative filtering technique: it estimates the similarity between users, and produces resource recommendations from users with similar tastes. To this end, we develop two novel graph-based metrics: 1) for measuring

[1] `http://www.neo4j.org/`

[2] `http://www.programmableweb.com/`

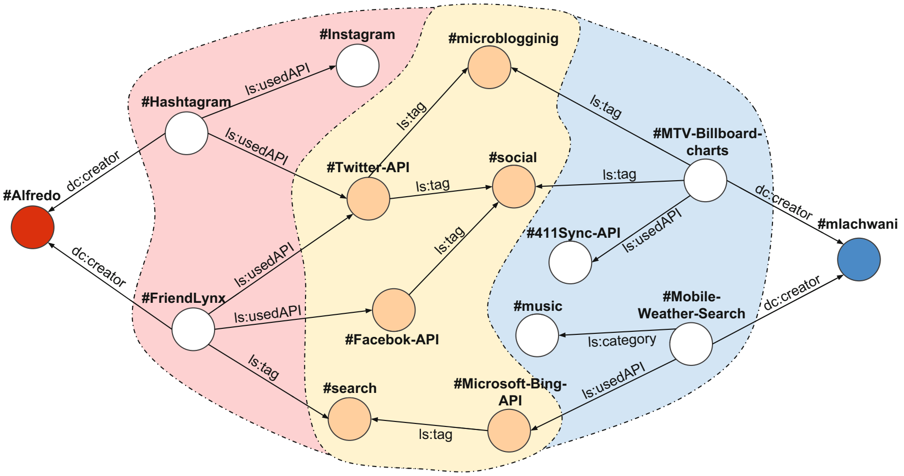


Fig. 1. Excerpt from the Linked Web APIs dataset with resource context graphs with context distance of 3

semantic resource similarity, and 2) for measuring *semantic resource relevance*. The first, we use to compute similarity between users represented as RDF resources. The second, uses the computed user similarities to estimate the relevance for each resource candidate.

The metric for measuring semantic resource similarity we develop based on a set of assumptions. The set of assumptions is as follows.

(i) ***The more information two resources share, the more similar they are.*** The first assumption is that if two resources share some information, then they are similar to each other. Considering an RDF graph, a shared information, as described in [1], might be an object of triples where subjects are the resources in question. In this case, only shared information in distance of one will be taken into consideration, when estimating their similarity. However, depending on the way the RDF data is modelled, similar resources might share information in any distance. Thus, in our method we allow adjustment of the context distance as required. Figure 1 shows an excerpt of the Linked Web APIs dataset (see Section 3.1 for more information about the dataset). In the figure, we present two context graphs with a distance of 3, for the users *Alfredo* and *mlachwani*. Considering the figure, the users have 6 resources in common. Note that if we choose a lower distance values, 1 or 2, no shared information will be evidenced.

(ii) ***Better connected shared resources carry more similarity information.*** According to the assumption, for the user *Alfredo*, the *Twitter-API* carries more similarity information than the *Facebook-API* or the *search* tag, since the *Twitter-API* is better connected to the resource representing

the user *Alfredo*. This can be evaluated by counting the simple paths[3] with a pre-defined maximum length, between the resources. From the *Alfredo*'s node the *Twitter-API* can be reached by two simple paths
$p_1 = \{$*Alfredo*, *Hashtagram*, *Twitter-API*$\}$
$p_2 = \{$*Alfredo*, *FriendLynx*, *Twitter-API*$\}$
, while the *Facebook-API* only by one
$p_3 = \{$*Alfredo*, *FriendLynx*, *Facebook-API*$\}$

(iii) ***Less probable shared resources carry more similarity information than the more common.*** Our assumption is that if two resources have in common more informative resources, then they are more similar. Considering the whole Linked Web APIs dataset, the *Microsoft-Bing-API* carries more information content, since its node is characterised with a low degree value 40 (due to its low usage in mashups, leading to a low number of incident links). On the other side, the *Twitter-API* and *Facebook-API* are popular Web APIs and extensively used in mashups, and their node degree values are 799 and 418, respectively. To conclude, the *Microsoft-Bing-API* is more informative than the *Twitter-API* and *Facebook-API* and will carry more similarity information.

Based on these assumptions, we develop our method for personalised resources recommendation. First, we propose a theoretical definition of Linked Data, followed by several definitions that we use to ground our metrics for computation of resource similarity and relevance. We present the algorithm that we use to compute the semantic similarity between resources, and the algorithm that uses the computed resource similarity to recommend relevant resources from similar users.

2.1 Definitions

Definition 1. *Let $\mathcal{G}$ be a Linked Data dataset defined as a graph $\mathcal{G} = (\mathcal{R}, \mathcal{L})$ in which $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ is a set of resources identified with their URIs, and $\mathcal{L} = \{l_1, l_2, ..., l_m\}$ is a set of links (predicates) between those resources, where $l_i = (r_j, r_k)$ is a concrete link between two resources.*

While this definition describes one dataset, the LOD cloud can be described as union of all G_i datasets. Note that ontologies are not excluded from the definition, and they can be also modelled.

Definition 2. *Let $\mathcal{G}_{r_i,d} = (\mathcal{R}_i, \mathcal{L}_i)$ be a sub-graph of a Linked Data dataset graph $\mathcal{G}$ whose resources ($\mathcal{R}_i$) and links ($\mathcal{L}_i$) sets are subset of those of $\mathcal{G}$ with restriction that only nodes within maximum distance d from the resource r_i are included. We will further refer to this sub-graph as a* ***resource context graph.***

Definition 3. *Let $\mathcal{C}_{r_i,r_j}$ be a set of resources shared by context graphs of the resources r_i and r_j. We will refer to this set of resources as a* ***shared context.***

[3] Note that by *simple path* we mean a path without repeating vertices, as it is defined in the graph theory.

According to the assumption (iii), in order to give less impact to the less informative resources, we perform weightening of the resources based on the information content (IC) they convey. In the information theory [12], the information content of a concept is defined as the logarithm (i.e., with base 2) of the inverse of its probability

$$IC(c) = -\log(\pi(c)), \tag{1}$$

where $\pi(c)$ is the probability of occurrence of the concept c. The probability $\pi(c)$ is calculated as the quotient of the frequency of occurrence of c and the total number of concepts in the corpus. In the following definition we adopt the general definition of IC to be applicable in Linked Data.

Definition 4. *Let $RIC(r_i)$ be a function which computes the IC carried by a resource (RIC) defined as*

$$RIC(r_i) = -\log(\frac{deg(r_i)}{max\{deg(r_k) : r_k \in \mathcal{R}\}}) \tag{2}$$

where the probability of occurrence of a resource is computed as the quotient of $deg(r_i)$ – resource degree computed as the total number of incident links, and $max\{deg(r_k) : r_k \in \mathcal{R}\}$ – the degree of the resource with the highest degree. Computed resource information content is within the interval [0,1]. See Section 3.2 for actual computed information content of the resources in the Linked Web APIs dataset.

Definition 5. *Let $gain(p)$ be a function which computes the gain of information from one end to another in a simple path $p = \{r_1, r_2, ..., r_n\}$ where r_i is the i-th resource in the list of resources visited in the path from r_1 to r_n. We define the function for computing the information gain as*

$$gain(p) = \prod_{i=1}^{n} RIC(r_i) \tag{3}$$

Note that the gain function is a multiplicative function of RIC weights with values between 0 and 1, and computed gain for longer paths will be lower than for shorter paths. We use the function to compute the connectivity of a shared resource with a user's resource. For a closer shared resource (shorter path) the computed gain will be higher, than for the more distant shared resource (longer path).

2.2 Algorithm: *"Computing Resource Similarity"*

The similarity between two resources r_i and r_j we compute according to the following algorithm.

Inputs:

- Graph $\mathcal{G}$ representing a Linked Data dataset.

- A context graph $\mathcal{G}_{r_i,d_i} = (\mathcal{R}_i, \mathcal{L}_i)$ for r_i with context distance d_i.
- A context graph $\mathcal{G}_{r_j,d_j} = (\mathcal{R}_j, \mathcal{L}_j)$ for r_j with context distance d_j.
- A shared context set $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ of the context graphs $\mathcal{G}_{r_i,d_i}$ and $\mathcal{G}_{r_j,d_j}$.

Output:

- A computed similarity sim_{ij} for the resources r_i and r_j.

Uses:

- A function $paths(r_i, r_j, d)$ that returns a set of all simple paths (with a maximum length d) between two resources.
- A function $gain(p)$ that computes the gain of information in a path p.

Algorithm:

```
1:  // compute the amount of similarity between the two resources
2:  // as a sum of the similarity carried by each shared resource in $\mathcal{C}$
3:  $sim_{ij} \leftarrow 0$
4:  for all $c_k \in \mathcal{C}$ do
5:      // sum the information gained in all simple paths between
6:      // the resource $r_i$ and the shared context resource $c_k$
7:      $\mathcal{P}_i \leftarrow paths(r_i, c_k, d_i)$, $s_{r_i} \leftarrow 0$
8:      $s_{r_i} \leftarrow s_{r_i} + \sum_{p \in \mathcal{P}_i} gain(p)$
9:      // sum the information gained in all simple paths between
10:     // the resource $r_j$ and the shared context resource $c_k$
11:     $\mathcal{P}_j \leftarrow paths(r_j, c_k, d_j)$, $s_{r_j} \leftarrow 0$
12:     $s_{r_j} \leftarrow s_{r_j} + \sum_{p \in \mathcal{P}_j} gain(p)$
13:     $sim_{ij} \leftarrow sim_{ij} + \frac{s_{r_i} + s_{r_j}}{2}$
14: end for
```

For each shared context resource c_k, the algorithm first retrieves all simple paths (lines 7 and 11) between the shared context resource c_k and the resources we compute similarity for (r_i and r_j). Next, In lines 8 and 12, the algorithm computes the gained information for each simple path taking into account the pre-computed resource informativeness. The algorithm independently computes the semantic similarity of the shared context resource c_k to the both resources (r_i and r_j). Finally, in line 13, the algorithm computes the semantic similarity carried by a single context resource, as an arithmetic mean of the computed similarity to the both resources (r_i and r_j). The final similarity score is computed as sum of the similarity information carried by each context resource.

2.3 Algorithm: *"Computing Resource Relevance"*

The computed resource similarity using the previous algorithm, is then used to compute the relevance of a resource candidate for a given user. The relevance of the resource for a user we compute according to the following algorithm.

Inputs:

- Graph $\mathcal{G}$ representing a Linked Data dataset.

- Resources r_u - a user requester, and r_c - a resource candidate.
- A set of users' resources $\mathcal{R}' = \{r_1, r_2, ..., r_n\}$, where $r_k \in \mathcal{R} \setminus r_u$.
- A set of user similarity scores $S = \{s_{u1}, s_{u2}, ..., s_{un}\}$, where s_{uk} is a semantic similarity computed with Alg 2.2 for the resource r_u and $r_k \in \mathcal{R}'$.

Output:

- A computed relevance score for the resource candidate r_c and the user r_i.

Uses:

- A function C that returns a resource context graph for a given resource.
- A function $paths(r_i, r_j, d)$ that returns a set of all simple paths (with a maximum length d) between two resources.
- A function $gain(p)$ that computes the gain of information in a path p.

Algorithm:

```
 1: $rel \leftarrow 0$
 2: for all $r_k \in \mathcal{R}'$ do
 3:     // create a resource context graph
 4:     $\mathcal{G}_{r_k} \leftarrow C(r_k, d, \mathcal{G})$
 5:     // check presence of the resource $r_c$ in the context graph
 6:     if $r_c \in \mathcal{G}_{r_k}$ then
 7:         // sum the gain of information for all simple paths between
 8:         // the user and the resource candidate
 9:         $\mathcal{P} \leftarrow paths(r_c, r_k, d)$
10:         $rel \leftarrow rel + s_{uk} * \sum_{p_i \in \mathcal{P}} gain(p_i)$
11:     end if
12: end for
```

First, the algorithm creates a context graph for each user similar with the user r_u (line 4). Next, the algorithm checks whether the resource candidate is present in the context graph (line 6). If yes, the algorithm computes the connectivity of the similar user and the resource candidate r_c (lines 9–10). The connectivity is computed as sum of the gained information for all the simple paths between the user and the resource candidate r_c. In line 10, the algorithm also takes into account the pre–computed similarity score s_{uk} between the users. The final score is a sum of the relevance values computed from each similar user.

3 Experimental Evaluation

In this section, we describe the dataset used for validation and evaluation of the method. We present a resource recommendation use case and we report on the results from several experiments. In the experiments we addressed following set of questions:

- *What is the quality of the recommendations provided by our method in comparison with the other traditional methods?*
- *How the resource information content (RIC) influences the quality of the recommendations?*
- *How surprising and diverse recommendations generates the method?*

3.1 Dataset Description

In order to validate and evaluate the method, we opted for the Linked Web APIs dataset [2]. The Linked Web APIs dataset is an RDF dataset representing the ProgrammableWeb service repository. It consists of information about developers, mashups they have developed, Web APIs used for the mashups, tags and categories associated with the mashups and the Web APIs. Moreover, the dataset also contains technical information about the Web APIs, such as supported protocols and data formats.

The Linked Web APIs dataset re-uses several well-known ontologies developed by the community. We use concepts from the FOAF[4] ontology (prefix `foaf`) to represent the mashup developers (*foaf:Person*), concepts from the WSMO-lite[5] [15] ontology (prefix `wl`) to represent the Web APIs (*wl:Service*), and terms such as *dc:title* and *dc:creator* from the Dublin Core[6] vocabulary (prefix `dc`). Further, we create new concepts (*ls:Mashup*, *ls:Tag*, *ls:Category*, *ls:DataFormat*) and properties (*ls:format*, *ls:tag*, *ls:category*, *ls:usedAPI*) for which we use the `ls` prefix.

The dataset represents the information of the ProgrammableWeb repository as of April 24th 2014, and it contains over 170K RDF triples describing 11,339 APIs, 7,415 mashups and 5,907 users. Figure 1 shows an excerpt of the dataset.

3.2 Use-case: Resources Recommendations

In order to validate and demonstrate our method, we developed a resource recommendation use case for the Web services domain: recommendation of resources representing Web APIs. For this purpose we used the Linked Web APIs dataset. After loading the dataset, for each resource we compute its information content (see definition 4). Table 1 shows the top 5 resources with highest and lowest information content.

Table 1. Top 5 resources with highest (left) and lowest RIC (right)

Resource ID	Label	RIC (bits)	Resource ID	Label	RIC (bits)
27766	Paigeadele user	1.00000	7	Service class	0.00000
27871	retouching tag	0.92576	13	Mashup class	0.04550
28017	Pbwiki API	0.88233	34	Person class	0.06985
28015	Usefulbytes API	0.85151	39	Tag class	0.13267
28014	Philly add API	0.82761	12	mapping tag	0.13273

As expected, resources which are more distinctive, have higher RIC, and will have higher influence in the similarity computation, while the most probable resources, are less informative, and will have less influence in the similarity computation. For example, the resources representing ontological classes, such as the

[4] `http://xmlns.com/foaf/0.1/`

[5] `http://www.w3.org/Submission/WSMO-Lite/`

[6] `http://dublincore.org/documents/dcmi-terms/`

wl:Service or *ls:Mashup* class, carry less RIC due to their high degree value in the RDF graph. On the other side, sharing resource representing the tag *retouching* or the *Usefulbytes API*, will carry more RIC due to their low degree value.

Next, using the Algorithm 2.2 we compute the similarity between the resources representing users. In the Linked Web APIs dataset those are instances of the *foaf:Person* class. When computing the resource similarity it is necessary to set the context distance of resource context graphs (see definition 2). We experimentally set the context distance to 2, thus only resources within distance of 2 will be taken into account when creating the resource context graphs. In this case, only resources representing mashups, Web APIs and assigned tags will be present in the context graphs. See Section 5 for discussion on setting the resource context distance.

Finally, using the Algorithm 2.3 we compute the relevance between each user and each resource candidate. Here, we focused on computation of relevance only for instances of the class *wl:Service*, however, the relevance can be be computed for any other resources, e.g., categories, mashups and even users. Table 2 shows the top 5 most similar users with the user *Alfredo* and the top 5 Web APIs with highest relevance score, also for the user *Alfredo*.

Table 2. Top 5 most similar users with the user *Alfredo* (left) and the top 5 most relevant Web APIs (right)

Resource ID	Username	Similarity score
511	Avishai	2.11250
731	Frogcologne	1.79806
20130	Nobosh	1.69410
2505	Tripsailor	1.64018
1407	Rakf1	1.63710

Resource ID	API Name	Relevance score
245	Twitter API	49.78257
10	Google Maps API	36.32023
129	Facebook API	33.34930
331	Box API	27.85667
165	Flickr API	24.60448

3.3 Evaluation

In order to evaluate the quality of our method in terms of *accuracy* and *usefulness* of the recommendations, we followed standard evaluation protocols for recommender systems. For the evaluation we used the Linked Web APIs dataset and we randomly created training (80%) and testing partition (20%). This led to creation of 3,089 test cases.

For the evaluation of the accuracy we focused on several standard well-known metrics used for evaluation of recommender systems [4]. The metrics used for the evaluation of the accuracy are as follows.

- *Precision and Recall.* A classical evaluation metrics where precision is defined as a fraction of the retrieved items that are relevant, and recall is defined as a fraction of the relevant documents that are retrieved.
- *Area Under the Curve (AUC).* Measures the quality of a list of ranked items. The AUC is equivalent to the probability that the recommender will rank a randomly chosen positive instance higher than a randomly chosen negative

instance. For a random recommender it can be expected to get half the positives and half the negatives correct with an AUC value close to 0.5.

- *Normalized Discounted Cumulative Gain (NDCG).* Measures the quality of a list of ranked items taking into account the position of each item. It gives higher weight to items with higher rank.
- *Mean Average Precision (MAP).* Measures the quality of the list of ranked items as mean of the average precision for a set of test queries.
- *Mean Reciprocal Rank (MRR).* Considers the rank position of the items in the ranking list. A reciprocal rank for a single query is computed as a reciprocal of the rank at which the relevant item is retrieved.

In order to evaluate how the information content influences the accuracy, we evaluated the accuracy on two variants of our method. One which takes into account the informativeness of the resources, and one which does not. For the latter, we experimentally set the informativeness for all the resources at a fixed value of 0.9. Note that choosing any value in the interval between 0 and 1 will have same effect on the final results. We also performed a comparison of our method with the traditional personalised collaborative filtering methods (*User-KNN* and *Item-KNN*)[7] and non-personalised methods (*Random*[8] and *Most popular*[9]), which we consider as baseline. The evaluation was conducted using the evaluation environment MyMediaLite[10] v3.10, which also provides implemen-

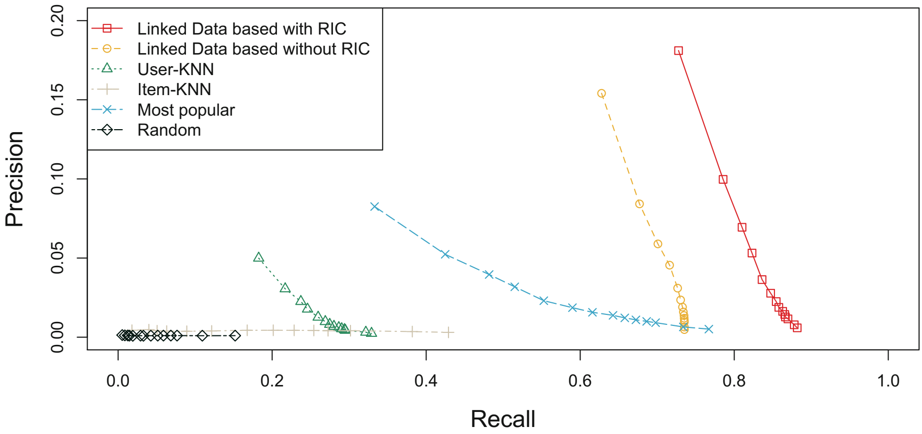


Fig. 2. Precision and Recall curves obtained for different methods.[11]

[7] For the UserKNN and ItemKNN baseline methods, was used a cosine similarity function with the default neighbourhood size experimentally set to k=80.

[8] Random recommender - randomly recommends items from a given set.

[9] Most popular recommender - recommends items weighted by the number of times they have been seen in the past.

[10] MyMediaLite evaluation environment - `http://mymedialite.net/`

[11] The precision/recall curves were obtained looking @topN, with N set to [5;10;15;20;30;40;50;60;70;80;90;100;150;200].

The results show that our method outperforms the traditional personalised *User-KNN* and *Item-KNN* recommendation methods, as well as the simple *Random* and *Most popular* recommendation methods. The results also show that our method achieves better results when taking into account the resource information content. From all the evaluated methods, the lowest results were achieved for the *Random* recommender. Slightly better results were achieved by the *Item-KNN*, followed by the *User-KNN*. It is interesting the fact that the *Most popular* method achieved better results compared to the other baseline methods. Most likely it is due to the long-tail distribution of the Web API usage in mashups, where small number of Web APIs enjoy significantly greater popularity than the others [13,16].

The results for the *AUC*, *NDCG*, *MAP* and *MRR* metrics are summarized in Table 3.

Table 3. Evaluation results for: Area Under the Curve (AUC), Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), Mean Reciprocal Rank (MRR)

	Random	**Most popular**	**User-KNN**	**Item-KNN**	**Linked Data based without RIC**	**Linked Data based with RIC**
AUC	0.50831	0.89072	0.64023	0.71038	0.89162	**0.95093**
NDCG	0.11608	0.38547	0.22278	0.11273	0.59401	**0.69486**
MAP	0.0064	0.26235	0.14506	0.02114	0.53442	**0.62358**
MRR	0.00742	0.2946	0.17653	0.02355	0.57835	**0.66882**

It can be observed that also for the other metrics our method outperforms the baseline methods. Here we can again see that the variant of our method, which takes into account the informativeness of the resources achieves better results over the variant which does not. An improvement of 6.65% was achieved for AUC, 16.98% for NDCG, 16.68% for MRR, and 15.64% for MRR.

Apart from the accuracy, another important dimension of the recommender system, as argued in [4], is the usefulness of the recommendations in terms of *"how surprising and diverse the recommendations are"*. Since in our case the user requester and the recommended items are represented as nodes in graph, we define the *serendipity* as the length of the shortest path between the requester's resource (r_u) and the recommended resource (r_i). A larger value of the shortest path indicates greater surprise. The overall serendipity of a set of resources (set C) is the average serendipity of the resources in the set.

$$Serendipity(r_u, C) = \frac{\sum_{r_i \in C} shortest\text{-}path(r_u, r_i)}{|C|} \tag{4}$$

The *diversity* of a set of recommended resources we compute as the average dissimilarity among all resource pairs. The formula used for computation of diversity is as follows.

$$Diversity(C) = \frac{\sum_{r_i \in C} \sum_{r_j \in C - r_i} (1 - similiarity(r_i, r_j))}{\frac{|C| * (|C| - 1)}{2}} \tag{5}$$

Here, the similarity between the resources we compute as the Jaccard coefficient of the context graphs of the resources (each r_i and r_j) in the set of recommendations C. Computed diversity score is in the [0, 1] interval, where values close to 0 indicates very similar set of recommended resource, and close to 1 very diverse resource recommendations. In the Web services recommendation use case, diverse recommendations can be considered those recommendations where the Web APIs belong to different category, have assigned different tags, support different protocols or data formats, or have been used by different users.

We evaluate the serendipity and diversity looking at the top 5, 10, 15 and 20 recommendations. The results from the evaluation of the serendipity and diversity are summarised in Table 4.

Table 4. Results from the evaluation of serendipity and diversity

	@top-N	Random	Most Popular	User-KNN	Item-KNN	Linked Data based without RIC	Linked Data based with RIC
Serendipity	@top-5	2.97752	2.66810	2.59197	2.68006	3.18881	3.03271
	@top-10	2.98455	2.67465	2.65514	2.70402	3.54821	3.26700
	@top-15	2.98364	2.65816	2.68101	2.71267	3.73117	3.36509
	@top-20	2.98455	2.65184	2.69780	2.70968	**3.84142**	**3.42444**
Diversity	@top-5	0.65339	0.58347	0.62092	0.63349	0.83417	0.81949
	@top-10	0.65317	0.61354	0.62411	0.64392	0.86044	0.82912
	@top-15	0.65370	0.60374	0.63159	0.64558	0.87511	0.82884
	@top-20	0.65347	0.60719	0.63276	0.64287	**0.88435**	**0.83114**

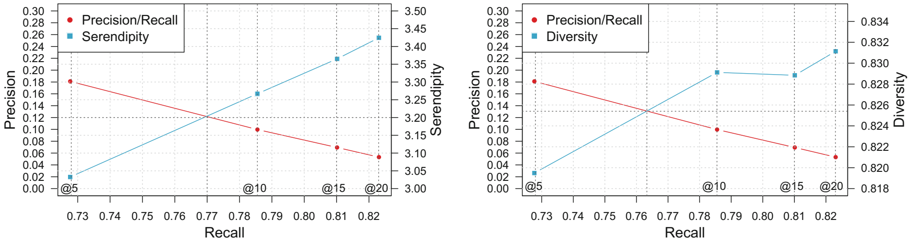
The results from the evaluation of serendipity and diversity show that our method outperforms the other methods. It can be also observed that the variant of our method which does not consider the informativeness of the resources produce more serendipitous and diverse recommendations compared to the variant which considers the informativeness. We can conclude that there is a trade-off between accuracy and serendipity/diversity which is directly influenced by the resource informativeness. In other words, when considering the resource informativeness our method provides more accurate but less serendipitous and diverse recommendations.

We also studied the optimal trade-off between the precision/recall and serendipity/diversity. Figure 3.3 depicts the obtained trade-off curves for our method.

The results show that the optimal values are: i) precision 0.12, recall 0.77 and serendipity 3.2, ii) precision 0.13, recall 0.763 and diversity 0.825. It can be also observed that the optimal precision/recall and serendipity/diversity is achieved when recommending between the top 5 and top 10 most relevant resources.

4 Related Work

A particular method that relates to ours is the *DBrec* presented in [10]. The method is supported by a semantic distance measure for measuring relatedness



(a) Precision/Recall – Serendipity

(b) Precision/Recall – Diversity

Fig. 3. Trade-off between serendipity and accuracy studied @top 5, 10, 15 and 20

between resources. The measure is defined as a function of the direct and indirect links between resources. One limitation of this measure is that similarity between resources can be measured only if the graph distance between the resources is not more than two. Since the smallest distance between the users in the Linked Web APIs dataset is four (i.e., *foaf:Person* → *ls:Mashup* → *wl:Service* → *ls:Mashup* → *foaf:Person*) the measure will fail to produce recommendations for the Linked Web APIs dataset. In comparison, our method can be easily adapted to any dataset by setting the resource context distance parameter. Although the *DBrec* method has been validated on different domains found in DBpedia it demands manual pre-processing of the dataset. The *DBrec* uses only subsets of the DBpedia dump, which needs to be defined in advance for the particular domain. In contrast, our method is not domain or dataset specific and does not require any manual pre-processing of the datasets – it exploits RDF datasets in their original form.

The authors in [1] propose a Linked Data enabled content-based movie recommender, which uses a vector space model to compute similarities between the movies. The approach is not suitable for computation of similarities of resources in datasets (i.e., Linked Web APIs) where the graph distance between the resources is more than two. Moreover, the approach has been developed for the movies domain [9] and its adaptation to other domains requires manual pre-processing. In [14] the authors also use the vector space model to compute similarities between entities for ontology alignment, however, only similarities between directly linked resources can be computed.

In [7] authors propose a *Lookup Explore Discovery* (LED) exploratory search system, which recommends DBpedia resources related to the named entities recognized in the query. The resources are ranked by exploiting i) the co-occurrence of the resources' labels in DBpedia abstracts and ii) the wikilinks information. Moreover, external information sources are queried (Google, Yahoo! and Bing) and their co-occurrence in the returned pages is also evaluated. The system exploits small portion from a single Linked Data dataset, which is DBpedia.

Discovery Hub [5] is an exploratory search engine which recommends resources from the DBpedia namespace. It uses an adapted version of the spreading activation algorithm over typed graphs. The system exploits only small portion of the available information in DBpedia – triples with properties *dcterms:subject* and *rdf:type*, and the DBpedia Pagelinks partition.

Aemoo [8] is another exploratory search system which provides a summary of knowledge about entities. It uses fixed of Encyclopedic Knowledge Patterns (EKPs) to filter out valuable knowledge about the entities. It uses DBpedia as primary source of knowledge

The only existing approach which considers the resource informativeness can be found in [6]. The informativeness of the resources is computed as sum of the information content of its features (directly linked resources). Thus, a resource linked to another resource with high information content, will result also in a high informative resource. In comparison, in our approach we compute the informativeness of the resources based on the number of in-out links incident with the resource. Therefore, the information content of the directly linked resources does not have influence on the resource informativeness.

5 Discussion and Future Work

Setting the Resource Context Distance. The resource context distance allows us to control the amount of context used when computing the resource similarity. The larger context distance we set, the more context is considered. For example, with distance set to 1, only directly linked resources will be used as context. In datasets, where users in an RDF graph are close to each other, will require setting lower distance, while in datasets, where users are far, will require higher distance. Choosing small context distance in datasets where the users are far from each other, can lead to possibly no overlap of the resource context graphs, and consequently no similarity computed. In our experiments, we set the distance to two, and thus, the context of the user resources will contain resources representing the mashups the user created, the Web APIs used in the mashups and the assigned tags. Also, it is obvious that the size of the context directly influences the time required for computation of the resource similarity. In our future work, we would like to explore methods for automatic determination of optimal context distance for a given dataset.

Resource Similarity Computation in Multi-Domain Datasets. When computing resource similarity our method uses the shared resource context. While the Linked Web APIs is a single-domain dataset, in a multi-domain datasets, such as DBpedia, the shared contexts might contain resources from various domains, which might have direct influence on the recommendations. For illustration, two users being similar in the music domain, does not mean they are similar also in the Web service domain. In our future work, we would like to explore such situations, assess their impact on the quality of the recommendations and appropriately adapt our method. Last but not least we want to evaluate the method on other benchmark datasets with different characteristics and from other domains. This includes, for example, the MovieLens, the DBLP dataset, and the ACM DL dataset.

6 Conclusions

A growing number of published datasets in the LOD cloud require new methods that can provide more efficient access to Linked Data. In this paper, we have presented a novel configurable method for personalised access to Linked Data. The method can be easily adapted to a dataset from any domain and make use of it. It relies on the collaborative filtering approach and it recommends resources from users with similar resource interests. The method is supported with two novel metrics for computing resource similarity and relevance. When computing the similarity between the users the method primarily takes into account the *commonalities*, the *informativeness* and the *connectiviteness* of the shared resources. We validated the method on a resource recommendation use case from the Web services domain and we presented its capabilities. We also evaluated the method on a real-world dataset and the results show that the method outperforms the traditional personalised collaborative filtering and non-personalised methods in terms of accuracy, serendipity and diversity. The results also show that considering the informativeness of the resources improves the quality of the recommendations.

Acknowledgement. This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS14/104/OHK3/1T/18. We also thank to Programmableweb.com for supporting this research.

References

1. Di Noia, T., et al.: Linked open data to support content-based recommender systems. In: Proceedings of the 8th International Conference on Semantic Systems, I-SEMANTICS 2012, pp. 1–8. ACM, New York (2012)
2. Dojchinovski, M., Kuchar, J., Vitvar, T., Zaremba, M.: Personalised graph-based selection of web aPIs. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 34–48. Springer, Heidelberg (2012)
3. Heath, T.: How will we interact with the web of data? IEEE Internet Computing 12(5), 88–91 (2008)
4. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22(1), 5–53 (2004)
5. Marie, N., Gandon, F., Ribière, M., Rodio, F.: Discovery hub: On-the-fly linked data exploratory search. In: Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS 2013, pp. 17–24. ACM, New York (2013)
6. Meymandpour, R., Davis, J.G.: Linked data informativeness. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 629–637. Springer, Heidelberg (2013)
7. Mirizzi, R., Di Noia, T.: From exploratory search to web search and back. In: Proceedings of the 3rd Workshop on Ph.D. Students in Information and Knowledge Management, PIKM 2010, pp. 39–46. ACM, New York (2010)
8. Musetti, A., et al.: Aemoo: Exploratory search based on knowledge patterns over the semantic web. Semantic Web Challenge (2012)

9. Ostuni, V.C., et al.: Cinemappy: a context-aware mobile app for movie recommendations boosted by dbpedia. In: de Gemmis, M., et al. (eds.) SeRSy. CEUR Workshop Proceedings, vol. 919, pp. 37–48. CEUR-WS.org (2012)
10. Passant, A.: dbrec — music recommendations using dBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010)
11. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Janowicz, K., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014)
12. Sheldon, R.: A First Course in Probability. Macmillan, New York (1976)
13. Tapia, B., Torres, R., Astudillo, H.: Simplifying mashup component selection with a combined similarity- and social-based technique. In: Proceedings of the 5th International Workshop on Web APIs and Service Mashups, Mashups 2011, pp. 8–14. ACM, New York (2011)
14. Tous, R., Delgado, J.: A vector space model for semantic similarity calculation and OWL ontology alignment. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 307–316. Springer, Heidelberg (2006)
15. Vitvar, T., Kopecký, J., Viskova, J., Fensel, D.: WSMO-lite annotations for web services. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 674–689. Springer, Heidelberg (2008)
16. Weiss, M., Gangadharan, G.R.: Modeling the mashup ecosystem: structure and growth. R&D Management 40(1), 40–49 (2010), http://dx.doi.org/10.1111/j.1467-9310.2009.00582.x, doi:10.1111/j.1467-9310.2009.00582.x