PageRank and Generic Entity Summarization for RDF Knowledge Bases

Chapter *in* Lecture Notes in Computer Science · June 2018

DOI: 10.1007/978-3-319-93417-4_10

2 authors, including:

Dennis Diefenbach
University of Lyon
20 PUBLICATIONS **120** CITATIONS

SEE PROFILE

PageRank and Generic Entity Summarization for RDF Knowledge Bases

Dennis Diefenbach[1] and Andreas Thalhammer[2]

[1] Université de Lyon, CNRS UMR 5516 Laboratoire Hubert Curien
dennis.diefenbach@univ-st-etienne.fr
[2] Roche Pharma Research and Early Development Informatics, Roche Innovation Center Basel, andreas.thalhammer@roche.com

Abstract. Ranking and entity summarization are operations that are tightly connected and recurrent in many different domains. Possible application fields include information retrieval, question answering, named entity disambiguation, co-reference resolution, and natural language generation. Still, the use of these techniques is limited because there are few accessible resources. PageRank computations are resource-intensive and entity summarization is a complex research field in itself.
We present two generic and highly re-usable resources for RDF knowledge bases: a component for PageRank-based ranking and a component for entity summarization. The two components, namely PageRankRDF and summaServer, are provided in form of open source code along with example datasets and deployments. In addition, this work outlines the application of the components for PageRank-based RDF ranking and entity summarization in the question answering project WDAqua.

Keywords: RDF, ranking, PageRank, entity summarization, question answering, linked data

1 Introduction

PageRank scores and entity summaries are important tools in many applications that are relying on RDF data. We want to start with concrete examples in the question answering domain:

PageRank scores can be used as a feature to disambiguate between resources. Suppose a user asks just for "River". While there are many different meanings for "River" like a film or a village, the most probable one is the one referring to a natural watercourse. PageRank scores can be used in this context to rank the different meanings of "River" and to present the most probable one to the user. Another possible application of PageRank scores is the ranking of an answer set. Suppose a user asks "Which lakes are located in Italy?" Without any ranking, the resulting list could easily start with an unknown lake like "Lake Reschen". This is probably not very relevant information for the user. By ranking the answer set properly, like in the information retrieval context, the usefulness of the answer for the user is increased.

Entity summarization [14] is the problem of identifying a limited number of ordered triples that summarize an entity in the best way—typically presented in knowledge panels. Those are then presented to the user together with the answer to a question (or search result) to enrich the current search context. Moreover they can also be useful for increasing the discoverability within the dataset (in the sense that the user can explore different aspects relating to the answer). Entity summarization and ranking algorithms (such as PageRank) are tightly related as the relevance of a triple given a specific entity needs to be estimated.

On one side, PageRank-based ranking and entity summaries can be essential tools in many domains like information retrieval, named entity disambiguation [20], entity linking[17], co-reference resolution, and natural language generation. On the other side, PageRank computations are resource-intensive and entity summarization is a research field in its own. So, while there are potentially many application areas the lack of easy access to re-usable resources is limiting the use of these technologies.

We present two highly re-usable resources for **1) PageRank [4] on RDF graphs** (PageRankRDF) that can be combined to a **2) generic framework for entity summarization** (summaServer). Both components are well documented and licensed under the MIT License (see Section 2). This enables extensibility and reusability without any types of restrictions. The framework has matured from earlier contributions [16,18,19] in the context of the WDAqua[3] research project with a focus on re-usable components for question answering [5,6,7].

This paper is organized as follows: In Section 2 we provide an overview of the presented resources. In Section 3 we first analyze the performance of PageRankRDF with respect to scalability in time and memory which are the limiting resources during PageRank computations. Second we compare the PageRank scores when computed over the RDF graph and when computed over the corresponding link structure of Wikipedia. In Section 4 we describe the summaServer component. We also describe how to extend summaServer in order to generate summaries for new knowledge bases and its API. In Section 5 we describe how PageRankRDF and summaServer are used in an existing question answering system called WDAqua-core1. In Section 6 we compare this work to existing ones and we conclude with Section 7.

2 Resources

The main contribution of this work encompasses the following two resources.

R1 A command line tool called PageRankRDF to compute PageRank scores over RDF graphs. The source code of PageRankRDF can be found at `https://github.com/WDAqua/PageRankRDF` with a complete documentation and usage instructions. It is released under the permissive MIT Licence.

[3] WDAqua (Answering Questions using Web Data) – `http://wdaqua.eu/`

Moreover we deliver some derived resources with the PageRank scores for some known datasets in the LOD cloud, namely:

- R 1.1 DBLP[4], using a dump provided by Jörg Diederich of the 22.07.2017, available under the DOI `https://doi.org/10.6084/m9.figshare.5767008.v1`.
- R 1.2 DBpedia [1][5], using the dump of latest release of English DBpedia[6], available under the DOI `https://doi.org/10.6084/m9.figshare.5769312`.
- R 1.3 Freebase [3][7], using the last Freebase dump before shutdown, available under the DOI `https://doi.org/10.6084/m9.figshare.5767017.v1`.
- R 1.4 MusicBrainz[8], using the dump of December 2016 generated using MusicBrainz-R2RML, available under the DOI `https://doi.org/10.6084/m9.figshare.5769189`. (`https://github.com/LinkedBrainz/MusicBrainz-R2RML`).
- R 1.5 Scigraph[9], using the current release of February 2017 (`http://scigraph.springernature.com/`), available under the DOI `https://doi.org/10.6084/m9.figshare.5769201.v1`.
- R 1.6 Wikidata [21][10], using the dump from the 28 September 2017, available under the DOI `https://doi.org/10.6084/m9.figshare.5766432.v1`.

The datasets are available at `https://figshare.com/projects/PageRank_scores_of_some_RDF_graphs/28119`.

R 2 An easily extensible framework for entity summarization called SUMMASERVER. It allows to generate entity summaries and currently supports the following knowledge bases: DBLP, DBpedia, Freebase, MusicBrainz, Scigraph, and Wikidata. Moreover it can be easily extended to support new knowledge bases. The source code of the SUMMASERVER can be accessed at `https://github.com/WDAqua/SummaServer`. It is released under the permissive MIT Licence. Moreover, we deliver a running service of the SummaServer. It can generate summaries for the above-mentioned knowledge bases that can be accessed at the following service endpoints:

- R 2.1 `https://wdaqua-summa-server.univ-st-etienne.fr/dblp/sum`
- R 2.2 `https://wdaqua-summa-server.univ-st-etienne.fr/dbpedia/sum`

[4] `http://dblp.l3s.de/dblp++.php`

[5] `www.dbpedia.org`

[6] All files retrived by: wget -r -nc -nH –cut-dirs=1 -np -l1 -A '*ttl.bz2' -A '*.owl'-R '*unredirected*'–tries 2 http://downloads.dbpedia.org/2016-10/core-i18n/en/, i.e. all files published in the english DBpedia. We exclude the following files: nif_page_structure_en.ttl, raw_tables_en.ttl and page_links_en.ttl. The first two do not contain useful links, while, the last one contains the link structure of Wikipedia that was already used in previews works [18].

[7] `http://freebase.com`

[8] `https://musicbrainz.org`

[9] `http://scigraph.springernature.com/`

[10] `www.wikidata.org`

R 2.3 https://wdaqua-summa-server.univ-st-etienne.fr/freebase/sum
R 2.4 https://wdaqua-summa-server.univ-st-etienne.fr/musicbrainz/sum
R 2.5 https://wdaqua-summa-server.univ-st-etienne.fr/scigraph/sum
R 2.6 https://wdaqua-summa-server.univ-st-etienne.fr/wikidata/sum

As a side note: From a previous contribution [19] there already exists the summaClient JavaScript component. It is a client of the SUMMASERVER that can be easily embedded in web pages. It is also licensed under the MIT License and can be accessed at https://github.com/athalhammer/summaClient.

3 Computation of PageRank on RDF Graphs

In the following we describe Resource R 1, namely PAGERANKRDF, a command line tool for computing PageRank scores over RDF graphs. In particular we analyze its scalability in terms of time and memory which are the limiting resources for PageRank computation. Then we analyze the quality of PageRank scores of Wikidata by comparing them with PageRank scores computed using untyped links between the corresponding Wikipedia articles.

3.1 Runtime Comparison: Non-HDT version vs. HDT-version

Implementing the Pagerank algorithm is a fairly easy task. The main problem is to make it scalable in terms of time and memory. We present two different ways to compute the PageRank scores over RDF graphs. Both implement the PageRank algorithm as presented by Brin and Page in [4]. The first implementation is a straight-forward implementation of the algorithm that takes as input an RDF dump in one of the current formats (like N-triples, Turtle) and computes the corresponding PageRank scores. The second implementation takes as input an RDF graph in HDT format [8]. HDT is a format for RDF that stores the graph in a very efficient way in terms of space. Generally, a factor $\times 10$ between the space consumption of the original RDF dump in one of the usual formats and the corresponding HDT dump is realistic. Moreover at the same time the RDF graph remains queryable, in the sense that triple patterns can be resolved in milliseconds. An HDT file contains three sections: the **H**eader (which simply contains some metadata), the **D**ictionray (which is a compressed mapping between URIs and integers) and the **T**riples (which are also compressed using the Dictionary and additional compression techniques). The second implementation is based on two observations. First, only the graph structure is important for the computation of the PageRank scores, i.e. the last section of the HDT file. Second, the dictionary section, i.e. the URIs, are occupying most of the space. The implementation basically computes the PageRank scores on the third section of the HDT file and uses the dictionary only at the end to assign the scores to the different URIs. This makes the second implementation much more time and memory efficient.

In Figure 1 the two implementations are compared by computing the PageRank scores for the Wikidata dump of the 28 September 2017 which has a size of 237 Gb and contains 2.2 billion triples. While the tool supports literals we ignore them in this experiment. It shows that when starting from an HDT dump of the graph the time consumption is reduced by a factor of $\times 19$ and the memory consumption by a factor of $\times 5$. In particular this last point is important since it allows the computation of PageRank scores of bigger datasets on affordable hardware. The time performance is increased for the following reason: When computing PageRank over an RDF file, most of the time is spent parsing and putting the data in a well-suited structure. The computation of the PageRank scores is rather short. With HDT the data is already in an optimal structure for the computation.
Note that HDT dumps of online available datasets can be found in the LOD laundromat [2][11] or under `http://www.rdfhdt.org/datasets/`. Moreover they can be easily created using the corresponding command line tools.[12]

3.2 Input Comparison: RDF relations vs. Wikipedia links

Next to the standard parameters "damping factor" and "number of iterations", PageRank [4] computations naturally depend most strongly on the input graph. Thalhammer and Rettigner showed in their work "PageRank on Wikipedia: Towards General Importance Scores for Entities" [18] that link filtering and weighting can have a strong influence on the output of PageRank calculations. In the same work it was indicated that the output of PageRank computations on the extracted RDF version of Wikipedia (i.e., DBpedia) could correlate less with page-view-based rankings than PageRank computations on the untyped Wikipedia link graph. However, the experiment was not performed and the following question is still open: "How do PageRank computations based on RDF relations compare to those based on Wikipedia links?" In order to answer this question, we start with the assumption that a higher ranking correlation (in our case Spearman's ρ and Kendall's τ)[13] to page-view-based rankings indicates a better ranking outcome.

The input data consists of three different ranking computations: PageRank on the Wikidata RDF graph (via PageRankRDF on a dump from September 28, 2017), PageRank on the Wikipedia link graph (computed with danker v0.1.0[14] on a Wikipedia dump from October 2, 2017 with option ALL[15]), and

[11] http://lodlaundromat.org/
[12] http://www.rdfhdt.org/manual-of-the-c-hdt-library/
[13] Both correlation measures have a codomain of $[-1, 1]$ where -1 means fully anti-correlated and 1 means fully correlated. For computing Spearman's ρ we used the R `cor.test` function and for computing Kendall's τ we used the function `cor.fk` of the R package pcaPP `https://cran.r-project.org/web/packages/pcaPP/`.
[14] danker v0.1.0 – `https://github.com/athalhammer/danker/releases/tag/v0.1.0`
[15] The option ALL uses the different language editions in a voting style using "bag of links semantics": if 200 languages cover the link *USA* $\rightarrow$ *Barack Obama* it is given

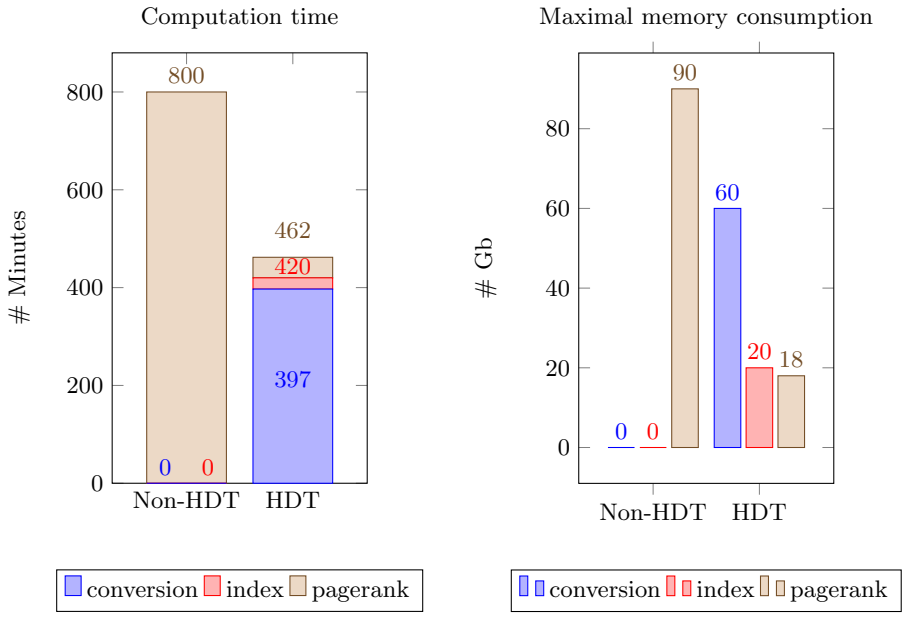

Fig. 1. This figure shows the time consumption and maximal memory consumption for the computation of the PageRank scores for Wikidata. We choose the dump of the 28 September 2017 which has a size of 237 Gb and 2.2 billion triples. The left figures shows the time consumption of the two implementation. The Non-HDT version takes 13 hours. The HDT version takes 42 minutes when the HDT file is already computed and 8.8 hours when the HDT file has to be generated from a different serialization. The right figure shows the memory consumption for the two implementation. The first implementation needs 90 Gb of RAM while the second 18 Gb if the HDT file is already computed and 60 Gb otherwise. The experiments were executed on a Server with Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz and 94Gb of RAM.

SubjectiveEye3D[16] by Paul Houle. The latter reflects the aggregated Wikipedia page view counts of the years 2008 to 2013 with different normalization factors (particularly considering the dimensions articles, language, and time). The datasets consist of different numbers of entities:

- Wikidata, PageRankRDF : 38 433 113 Q-IDs (total 80 646 048 resources)
- Wikidata, danker (Wikipedia links): 17 645 575 Q-IDs
- SubjectiveEye3D: 6 211 717 Q-IDs
- PageRankRDF ∩ danker ∩ SubjectiveEye3D: 4 253 903 Q-IDs

danker only includes entities from the Wikipedia namespace 0 (Main/Article), which particularly excludes "File" (namespace 6) and "Category" (namespace 14).

more importance than 15 languages that cover the link *USA* → *Donald Trump*. The PageRank algorithm supports multiple occurrences of the same link by default.

[16] SubjectiveEye3D – `https://github.com/paulhoule/telepath/wiki/SubjectiveEye3D`

Table 1. Spearman's ρ / Kendall's τ correlations of PageRank on RDF relations vs. Wikipedia links (via danker) and the comparison to SubjectiveEye3D.

	PageRankRDF	danker	SubjectiveEye3D
PageRankRDF	1.000 / 1.000	0.427 / 0.328	0.184 / 0.138
danker	0.427 / 0.328	1.000 / 1.000	0.400 / 0.276
SubjectiveEye3D	0.184 / 0.138	0.400 / 0.276	1.000 / 1.000

Both types of entities are included in the SubjectiveEye3D dataset which, in consequence, reduces the number of entities in the intersection set significantly. Another reduction factor were articles that have been deleted since 2013 (the upper limit of the SubjectiveEye3D input).

The result of the mutual correlation computations is outlined in Table 1. Both PageRank-based rankings have a positive correlation with the page-view-based ranking. The results show that danker correlates stronger with SubjectiveEye3D than PageRankRDF for both ranking correlation measures. Note that danker is tailored to the Wikipedia and/or Wikidata setting while PageRankRDF generalizes for all RDF graphs. Although there is no separation of A-Box and T-Box in Wikidata, terms like "Wikipedia Category" (`wd:Q4167836`), "scientific article" (`wd:Q13442814`), and "human" (`wd:Q5`) are prevalent in the top ten terms in the output of PageRankRDF. For specific applications it could make sense to pre-filter the input graph by certain predicates, such as `rdfs:subClassOf`, but this comes at the cost of generality and could impact the ranking output on other ends. Therefore, all datasets presented in R 1.x were computed without such pre-filtering.

The correlation between danker and SubjectiveEye3D is weaker than expected from the more positive results of [18]. In that work, the PageRank experiments are based on page link datasets of English Wikipedia. In contrast, the danker ALL option factors in page links from all Wikipedia language editions and therefore reduces bias towards English Wikipedia. One possibility for the lower correlation could be that SubjectiveEye3D maintains a rather strong bias towards English Wikipedia (despite the mentioned normalization steps).

4 Re-usable API for Serving Summaries of Entities

In this section we present Resource R 2—namely the summaServer—a service implementation that serves summaries of entities contained in RDF graphs. We first recapitulate the SUMMA API design [19] which is implemented by summaServer. Then, we sketch how a typical entity summarization service can be implemented using the summaServer code base.

4.1 The SUMMA API

The SUMMA API [19] is composed of two main components:

– SUMMA Vocabulary.[17]
– RESTful interaction mechanism.

This combination enables seamless integration with other Semantic Web components and a large degree of freedom with respect of the underlying entity summarization algorithm(s). When requesting a summary of an RDF entity only two parameters are mandatory:

entity the URI of the target resource (i.e., the resource to be summarized).
topK the number of triples the summary should contain.

The first interaction with the RESTful server is an HTTP `POST` request for creating a summary (see Listing 1). Note that the identifier of the summary in the summary request is a blank node (Turtle notation). The request basically says: "I would like the server to create a summary that complies with the given parameters." The server then responds with HTTP code 201 (`CREATED`). The `Location` header field denotes where we can find the newly created summary for future reference (i.e., to be accessed via `GET`): `https://wdaqua-summa-server.univ-st-etienne.fr/wikidata/sum?entity=http://www.wikidata.org/entity/Q42&topK=5&maxHops=1&language=en`.

Listing 1. Example `POST` request for creating a new summary.

```
curl -d "[ a <http://purl.org/voc/summa/Summary> ; \
<http://purl.org/voc/summa/entity> \
<http://www.wikidata.org/entity/Q6414> ; \
<http://purl.org/voc/summa/topK> 5 ] ." \
-H "Content-type: text/turtle" \
-H "Accept: application/ld+json" \
https://wdaqua-summa-server.univ-st-etienne.fr/wikidata/sum
```

Different client applications can request summaries and interpret the returned content. For this, SUMMASERVER can parse and create output in all standard RDF serializations (in accordance to the provided `Content-type` and `Accept` header parameters). As a matter of fact, summaries do not necessarily need to be requested via `POST` requests but can also be directly accessed via `GET` (SUMMASERVER keeps the URL layout). However, the interaction mechanism could also return summaries identified by non-speaking URIs like `https://wdaqua-summa-server.univ-st-etienne.fr/wikidata/sum/xyz`. An example implementation of a client—the summaClient JavaScript component (`https://github.com/athalhammer/summaClient`)—can interact with any server that implements the SUMMA API layout (see for example Section 5.2).

For more details on SUMMA the reader is kindly referred to [19].

4.2 Implementation Guide

We briefly want to describe the idea used by the SUMMASERVER to generate summaries. Imagine one wants to generate the summary for an entity, like the

[17] Available at `http://purl.org/voc/summa`

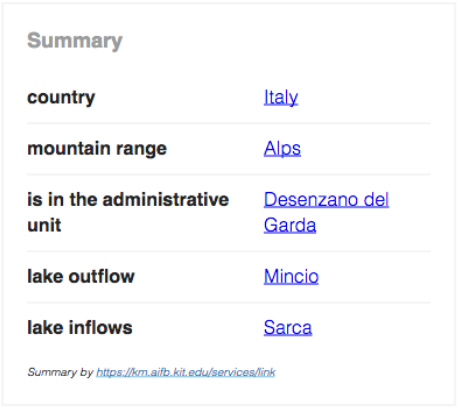

Fig. 2. Example of a summary for "Lake Garda".

Wikidata entity Q6414 corresponding to "Lake Garda", one of the biggest Italian lakes. The objective is to present to the user, between all facts that are known about this entity, the ones that best summarize it. An example of a summary for the "Lake Garda" is given in Figure 2. The idea presented in [16] generates the summary for a target entity X using the following straight-forward strategy. First the knowledge base is explored around X in a breadth-first traversal up to a certain depth (typically only 1, i.e., the next neighbours). For all reached entities the PageRank scores are considered and ranked in decreasing order. The entities corresponding to the first *topK* scores are shown in the summary. In the concrete example of "Lake Garda" the first 5 entities would be "Italy", "Alps", "Desenzano del Garda", "Mincio" and "Sarca". Note, during the breadth-first search the knowledge base can be either traversed in a directed or in an undirected way. In the following, we assume that the PageRank scores for all entities in the knowledge base were computed (for example using the command line tool in Section 3) and stored using the vRank vocabulary [13]. Moreover the PageRank scores are loaded in a SPARQL endpoint together with the original knowledge base. Setting up the SUMMASERVER to generate summaries for entities reduces to: indicate the address of the SPARQL endpoint and writing three SPARQL queries. We want to describe the three queries using as a concrete example the Wikidata knowledge base.

Listing 2. QUERY 1: This query retrives for an ENTITY the corresponding label in the language LANG. For Wikidata the query is

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?l
WHERE {
  <ENTITY> rdfs:label ?l .
  FILTER regex(lang(?l), "LANG", "i") .
}
```

(note that this information must be given since there are multiple ways to express the label of an entity. For example in MusicBrainz it is indicated with properties like `<http://xmlns.com/foaf/0.1/name>` and `<http://purl.org/dc/elements/1.1/title>`)

Listing 3. QUERY 2: This query must retrieve the resources connected to the resource ENTITY, order them according to the PageRank score and take the first TOPK. Moreover it retrieves the labels of the founded resources in the language LANG.

```
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX vrank: <http://purl.org/voc/vrank#>
PREFIX wdd: <http://www.wikidata.org/prop/direct/>
SELECT DISTINCT ?o ?l ?pageRank
WHERE {
   <ENTITY> ?p ?o .
      FILTER (?p != rdf:type && ?p != wdd:P31
      && ?p != wdd:P735 && wdd:P21
      && ?p != wdd:P972 && wdd:P421
      && ?p != wdd:P1343 )
   ?o rdfs:label ?l .
   regex(lang(?l), "LANG", "i") .
   graph <http://wikidata.com/pageRank> {
      ?o vrank:pagerank ?pageRank .
   }
}
ORDER BY DESC (?pageRank) LIMIT TOPK
```

(note that we do not traverse the edges with some labels like rdf:type and wdd:P31).

Listing 4. QUERY 3: This query must retrieve given two resource, ENTITY and OBJECT, the label of the property between them in the language LANG. For Wikidata we use the following query:

```
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX vrank:<http://purl.org/voc/vrank#>
SELECT ?p ?l
WHERE {
<ENTITY> ?p <OBJECT> .
OPTIONAL {
    ?o <http://wikiba.se/ontology-beta#directClaim> ?p .
    ?o rdfs:label ?l .
    FILTER regex(lang(?l), "LANG", "i")
}}
ORDER BY asc(?p) LIMIT 1
```

(note that in Wikidata the label of a direct property is not directly attached to it .)

We have implemented such queries for the following knowledge bases: Wikidata, DBpedia, DBLP, MusicBrainz, Freebase and the Scigraph. The imple-

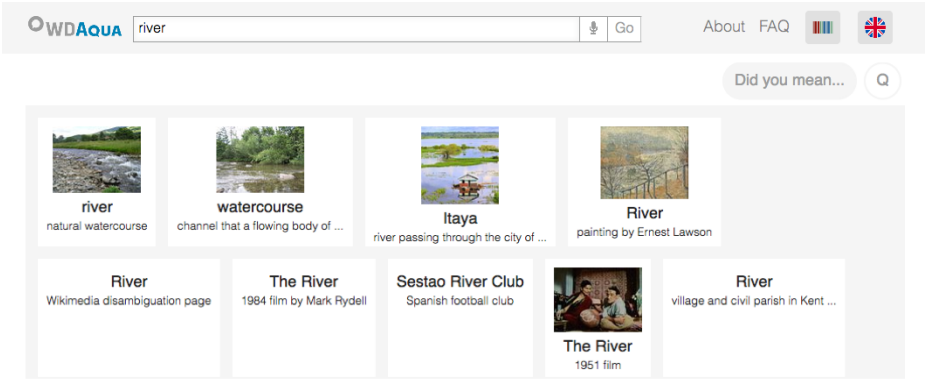

Fig. 3. Screenshot of WDAqua-Core1 for the question "River". The "did you mean" functionality shows other possible meanings of "River" that the user could have intended. The ranking, from left to right, from top to bottom, is based on PageRank scores.

mentations can be found at `https://github.com/WDAqua/SummaServer/tree/master/src/main/java/edu/kit/aifb/summarizer/implemented`. After indicating the endpoint and writing the three above queries the SUMMASERVER provides a summarization service for the corresponding knowledge base. For a more detailed instruction we refer to `https://github.com/WDAqua/SummaServer#extending-to-a-new-knowledge-base-kb`.

5 Use case: Question Answering

In this section we show how the PageRank scores and the entity summarization services are used in the Quesiton Answering system WDAqua-Core1 [6,7].

5.1 PageRank for Question Answering

PageRank scores are used by WDAqua-core1 at two places. The first is for disambiguating entities. Suppose a user just asks for "River" and the question answering system uses Wikidata as an underlying knowledge bases. Multiple entities could be meant like `Q4022` (a natural watercourse), `Q2784912` (a village and civil parish in Kent) or `Q7337056)` (a studio album by Izzy Stradlin). The question is ambiguous but one still wants to present the most probable interpretation to the user. PageRanks are used here to identify the most probable intended interpretation by the user, i.e. the one with the highest PageRank between the possible candidates. A concrete usage is shown in Figure 3.
A second application of PageRank scores relates to result set ordering. Imagine the following scenario. A user asks "Give me lakes in Italy." There are hundreds of lakes in Italy and currently there are 499 in Wikidata. Returning just a list will not be very useful for the user. Since the order is random the first presented

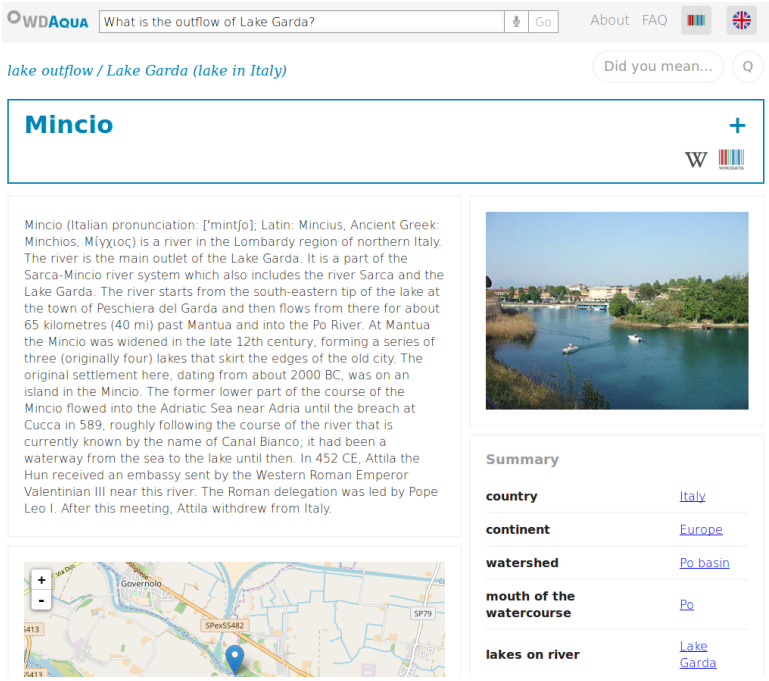

Fig. 4. Screenshot of WDAqua-Core1 for the question "What is the outflow of Lake Garda?". The entity summary is on the right-bottom part. Note that the links are discoverable, i.e. by clicking on "Po" information of "Po" are displayed (in the same way if the user asked directly for "Po").

lakes can be some unknown lake like the "Lago di Posta Fibreno". Ranking the answers according to PageRank will provide "Lago di Garda" and "Lago di Como" in the top ranks which is probably more relevant information for the user.
The PageRank scores used in WDAqua-Core1 correspond to R 1.1, R 1.2, R 1.3, R 1.4, R 1.5, R 1.6 and are computed using the tool presented in Section 3.

5.2 Entity Summarization for Question Answering

Entity summaries are used in Trill [5], the front-end used by WDAqua-Core1. An example is given in Figure 4. The summarization service is used mainly for two reasons: First, to add context to the retrieved answer. An expected result of this is that the confidence of the user in the answer is increased. Second, to increase discoverability within the dataset, i.e., offering a number of facts related to the answer entity the user. The facts are browse-able in the sense that the summary facts are clickable links that allow to easily explore other information in the graph that are connected to the original entities. WDAqua-Core1 currently uses the summarization services offered by SUMMASERVER corresponding to R 2.1,

R 2.2, R 2.3, R 2.4, R 2.5, R 2.6. As explained in Section 3.2 the PageRank scores computed over the linked structure of Wikipedia express better the page views of the user. Since in DBpedia every entity corresponds to a Wikipedia article, for DBpedia we use the PageRank scores computed over the linked structure of Wikipedia.
A demo of WDAqua-Core1 can be found at `www.wdaqua.eu/qa`.

6 Related Work

We touch on two fields in this work, namely ranking for RDF knowledge bases and entity summarization. For a good survey on ranking for RDF knowledge bases we refer the reader to Roa-Valverde and Sicilia [12]. Recent work on this topic includes Ngomo et al. [10] which gives an alternative to traditional PageRank computation.[18] Also some vendors have included PageRank functionality in their products.[19] We presented an efficient implementation of PageRank that, when data is already provided in HDT format (as often already done; see LOD laundromat [2]), has a very high time and memory efficiency. For an overview on the field of entity summarization we kindly refer the reader to Section 2.2 of [14]. Recent work includes Pouriyeh et al. [11].

The presented work is intended to provide findable, accessible, interoperable, and re-usable (FAIR) baselines for ranking and entity summarization in RDF knowledge bases. It stands in the light of the FAIR guiding principles [22] that every modern researcher should try to adhere to. We build on [19] where the SUMMA API was originally presented. Next to the service endpoints presented in this work, this API definition has been implemented by [15] with the show case of DBpedia and is online available. We encourage other researchers in the field also to publish their research prototypes along the FAIR guiding principles by adhering to the SUMMA API definition. To the best of our knowledge, DBpedia/Wikidata PageRank[20] [18] is currently the only public source for precomputed datasets of knowledge bases that can easily be loaded into triplestores. PageRankRDF builds on this work and provides general, affordable PageRank computation for RDF knowledge bases. An initial implementation of PageRankRDF was used for experiments by Andreas Harth that are documented at `http://harth.org/andreas/2016/datenintelligenz/`.

7 Summary

We have presented two important and tightly connected resources: a command line tool for computing PageRank scores called PageRankRDF R 1, and a

[18] We tried to use the provided library to carry out the same experiments presented in this paper. The corresponding discussion with the authors can be found here: `https://github.com/dice-group/HARE/issues/1`.
[19] See for example `https://wiki.blazegraph.com/wiki/index.php/RDF_GAS_API#PageRank` or `http://graphdb.ontotext.com/documentation/free/rdf-rank.html`.
[20] DBpedia/Wikidata PageRank – `http://people.aifb.kit.edu/ath/`

framework for computing entity summaries called SUMMASERVER R2. The code is open source and available under an open licence. We have demonstrated that PAGERANKRDF can scale up to large datasets and we are publishing the computed scores for example knowledge bases. Moreover, we have described SUMMASERVER and shown how it can be extended to new knowledge bases. Finally, we have shown how the existing resources are used in a concrete scenario, namely in the existing question answering system WDAqua-Core1.
The presented resources will be maintained and used within the WDAqua ITN Project[21] [9]. Due to the popularity of the previously published PageRank scores [18] for DBpedia/Wikidata and the number of possible applications in different research areas, we believe that the presented resources are an important contribution for the Semantic Web community.
Acknowledgments Parts of this work received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 642795, project: Answering Questions using Web Data (WDAqua).

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: The Semantic Web, Lecture Notes in Computer Science, vol. 4825, pp. 722–735. Springer Berlin Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-76298-0_52
2. Beek, W., Rietveld, L., Bazoobandi, H.R., Wielemaker, J., Schlobach, S.: Lod laundromat: a uniform way of publishing other peoples dirty data. In: International Semantic Web Conference. pp. 213–228. Springer (2014)
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250. SIGMOD '08, ACM, New York, NY, USA (2008), http://dx.doi.org/10.1145/1376616.1376746
4. Brin, S., Page, L.: The Anatomy of a Large-scale Hypertextual Web Search Engine. In: Proceedings of the Seventh International Conference on World Wide Web 7, pp. 107–117. WWW7, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands (1998), http://dx.doi.org/10.1016/S0169-7552(98)00110-X
5. Diefenbach, D., Amjad, S., Both, A., Singh, K., Maret, P.: Trill: A reusable front-end for qa systems. In: ESWC P&D (2017)
6. Diefenbach, D., Singh, K., Maret, P.: Wdaqua-core0: A question answering component for the research community. In: ESWC, 7th Open Challenge on Question Answering over Linked Data (QALD-7) (2017)
7. Diefenbach, D., Both, A., Singh, K., Maret, P.: Towards a question answering system over the semantic web (2018), arXiv:1803.00832
8. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF representation for publication and exchange (HDT). Web Semantics: Science, Services and Agents on the World Wide Web 19, 22–41 (2013), http://dx.doi.org/10.1016/j.websem.2013.01.002
9. Lange, C., Shekarpour, S., Auer, S.: The WDAqua ITN: Answering questions using web data. In: EU project networking session at ESWC (2015)

[21] http://wdaqua.eu

10. Ngomo, A.C.N., Hoffmann, M., Usbeck, R., Jha, K.: Holistic and Scalable Ranking of RDF Data. In: 2017 IEEE International Conference on Big Data. to appear (2017)
11. Pouriyeh, S., Allahyari, M., Kochut, K., Cheng, G., Arabnia, H.R.: ES-LDA: Entity Summarization using Knowledge-based Topic Modeling. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 316–325. Asian Federation of Natural Language Processing (2017), http://aclweb.org/anthology/I17-1032
12. Roa-Valverde, A.J., Sicilia, M.A.: A survey of approaches for ranking on the web of data. Information Retrieval 17(4), 295–325 (2014), http://dx.doi.org/10.1007/s10791-014-9240-0
13. Roa-Valverde, A.J., Thalhammer, A., Toma, I., Sicilia, M.A.: Towards a formal model for sharing and reusing ranking computations. In: Proceedings of the 6th International Workshop on Ranking in Databases (DBRank 2012) held in conjunction with the 38th Conference on Very Large Databases (VLDB 2012) (2012), http://www.aifb.kit.edu/web/Inproceedings3537
14. Thalhammer, A.: Linked Data Entity Summarization. Phdthesis, KIT, Fakultät für Wirtschaftswissenschaften, Karlsruhe (2016), http://dx.doi.org/10.5445/IR/1000065395
15. Thalhammer, A., Lasierra, N., Rettinger, A.: LinkSUM: Using Link Analysis to Summarize Entity Data. In: Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings, Lecture Notes in Computer Science, vol. 9671, pp. 244–261. Springer International Publishing, Cham (2016), http://dx.doi.org/10.1007/978-3-319-38791-8_14
16. Thalhammer, A., Rettinger, A.: Browsing DBpedia Entities with Summaries. In: The Semantic Web: ESWC 2014 Satellite Events, pp. 511–515. Lecture Notes in Computer Science, Springer International Publishing, Cham (2014), http://dx.doi.org/10.1007/978-3-319-11955-7_76
17. Thalhammer, A., Rettinger, A.: ELES: combining entity linking and entity summarization. In: International Conference on Web Engineering. pp. 547–550. Springer (2016)
18. Thalhammer, A., Rettinger, A.: PageRank on Wikipedia: Towards General Importance Scores for Entities. In: The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016, Revised Selected Papers, pp. 227–240. Springer International Publishing, Cham (Oct 2016), http://dx.doi.org/10.1007/978-3-319-47602-5_41
19. Thalhammer, A., Stadtmüller, S.: SUMMA: A Common API for Linked Data Entity Summaries. In: Engineering the Web in the Big Data Era, Lecture Notes in Computer Science, vol. 9114, pp. 430–446. Springer International Publishing, Cham (2015), http://dx.doi.org/10.1007/978-3-319-19890-3_28
20. Tristram, F., Walter, S., Cimiano, P., Unger, C.: Weasel: a machine learning based approach to entity linking combining different features. In: Proceedings of 3th International Workshop on NLP and DBpedia, co-located with the 14th International Semantic Web Conference (ISWC 2015), October 11-15, USA (2015), http://nbn-resolving.de/urn:nbn:de:0070-pub-27755352
21. Vrandečić, D., Krötzsch, M.: Wikidata: A Free Collaborative Knowledgebase. Communications of the ACM 57(10), 78–85 (2014), http://dx.doi.org/10.1145/2629489
22. Wilkinson, M. D. *et al.*: The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3 (2016), http://dx.doi.org/10.1038/sdata.2016.18