Sindice at SemSearch 2010

Renaud Delbru Nur Aini Rakhmawati Giovanni Tummarello
firstname.lastname@deri.org
Digital Enterprise Research Institute
National University of Ireland, Galway
Galway, Ireland

ABSTRACT

In this report we describe the model which Sindice uses to define, index and rank “entites” found on the web of data. We first give a definition of entity representation and then describe both the query-dependent and query-independent ranking procedures and how they are combined in the Semantic Information Retrieval Engine (SIREn) of Sindice.

1. INTRODUCTION

In this paper, we present a set of technologies built for and used by Sindice [5] and how they can be used to ranked web data, i.e. the Billion Triple Challenge dataset[1]. We first explain how the BTC dataset has been processed and indexed in Sect. 2. We then describe in Sect. 3 how query-dependent ranking is performed, in Sect. 4 how query-independent ranking is performed and in Sect. 5 how these two ranking evidence are combined into a final entity rank.

2. ENTITY INDEXING MODEL

The Semantic Information Retrieval Engine, SIREn [2], is a system based on Information Retrieval (IR) techniques and designed to search “entities” specifically according to the requirements of the Web of Data. The main use case for which SIREn is developed is entity search: given a description of an entity, i.e. a star-shaped query such as the one in Fig. 1(b), locate the most relevant entities and datasets. The Fig. 1(a) shows an RDF graph and how it can be split into three entities *renaud*, *giovanni* and *DERI*. Each entity description forms a sub-graph containing the incoming and outgoing relations of the entity node which is indexed by the system.

Since RDF is semi-structured, SIREN is aimed to support three types of queries: 1. full-text search (keyword based) when the data structure is unknown, 2. semi-structural queries (complex queries specified in a star-shaped structure) when the data schema is known, 3. or a combination of the two (where full-text search can be used on any part of the star-shaped query) when the data structure is partially known.

2.1 Entity Description Extraction

A pre-processing step is required in order to extract the entity descriptions from the dataset. First, the set of n-quads from the dataset is ordered by context and subject. Then, we perform a scan on the sorted n-quads and extract all the n-quads having the same context and subject. The extracted subset of n-quads forms an entity description. The entity description is sent to SIREn for further pre-processing.

Compared to Fig. 1(a), we extract only the outgoing relations of an entity node. Furthermore, we filter all entity descriptions that are composed of only one or two triples in order to reduce the index size and reduce the noise.

[1]BTC Dataset: `http://challenge.semanticweb.org/`

2.2 Entity Description Preprocessing

Each entity description received by SIREn is a list of n-quads. SIREn, before indexing, performs various preprocessing tasks for normalising the data. Literals as well as URIs are tokenised using various rules. Tokenising URIs is useful since it allows to perform keyword search on URIs parts. For example one could just use the local name, e.g. `person`, to match `foaf:person` ignoring the namespace. Then each token is lower-cased. We also filter a few stopwords and words with only one character. No stemming is performed.

3. QUERY-DEPENDENT RANKING

Our scoring functions evaluate individuals (contexts and subjects) by aggregating the values of their partial scores (predicates and objects). Likewise, the score of a predicate or object is an aggregation of the score of the matching terms computed with tf-isf (or term frequency - inverse subject frequency). The tf-isf weighting scheme assigns to the term t a weight in a subject s using the following formulae

$$tf.isf(t,s) = tf_{t,s} \times isf_t$$

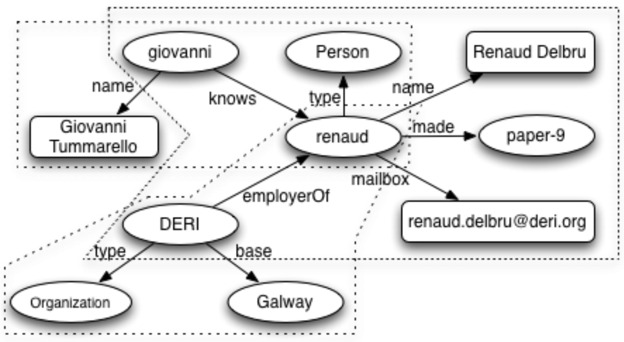
Since URIs are treated as terms, we can apply the tf-isf weighting scheme on URIs (or parts of them since URIs are normalized). The tf-isf scheme gives a low weight to a predicate or an object that occurs in many entities as, for example, for ubiquitous predicates like `rdf:type` or generic classes like `foaf:Agent`.

Given a boolean query q, an object o, a predicate p and an entity s:

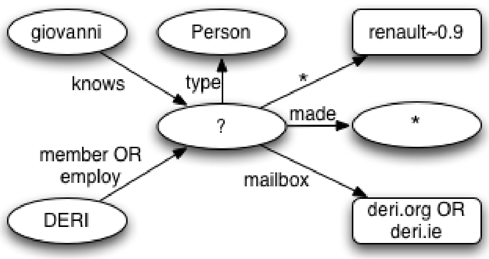
$$score(q,o) = \sum_{t \in q} tf.isf(t,s) \quad (1)$$

$$score(q,p) = \sum_{c \in p} score(q,o) \quad (2)$$

$$score(q,s) = \sum_{p \in s} score(q,p) \times spread(q,s) \quad (3)$$



(a) Visual representation of an RDF graph. The RDF graph is divided (dashed lines) into three entities identified by the nodes *renaud*, *giovanni* and *DERI*



(b) Star-shaped query matching the entity *renaud* where *?* is the bound variable and $\star$ a wildcard

Figure 1: In these graphs, oval nodes represent resources and rectangular ones represent literals. For space consideration, URIs have been replaced by their local names.

with

$$spread(q, s) = \frac{(|q| - |o|) + 1}{|q|}$$

where $|q|$ stands for the number of distinct terms in the query q and $|o|$ the number of objects matching at least one keyword. The spread normalisation factor favours queries that match terms within a single object. More the query terms are spread over multiple objects, smaller the normalisation factor will be. For example, if a query is composed of 3 terms and all the terms occur in a same object, then the normalisation factor will be equal to $\frac{(3-1)+1}{3} = 1$. On the contrary, if each term occurs in a different object, the normalisation factor will be equal to $\frac{(3-3)+1}{3} = 1/3$.

Additionally, we compute the relevance of the subject and context URIs for a given query. We consider keywords matching parts of the subject or context URI as an indication of the relevance of the entity. We combine linearly this additional evidence with the previous score using a weight factor $\alpha = 2$.

4. QUERY-INDEPENDENT RANKING

The query-independent entity ranking is performed using hierarchical link analysis algorithms [3] which are based on a two layer model of the Web of Data, pictured in Fig. 2. The top layer (dataset layer) is composed of a collection of interconnected datasets whereas the lower layer (entity layer) is composed of independent graphs of entities.

The top layer, or dataset graph, can be seen as an approximation of the global entity graph G. Instead of considering entities and links between these entities, we are using higher-level information such as datasets and *linksets*. A linkset is sets of edges having the same label and connecting two datasets. The lower layer, or entity graph, is composed of disjoint graphs D each of them being a collection of entities and intra-dataset links.

According to the hierarchical random walk model, we can apply a two-stage computation. In the first stage, we calculate the importance of the top level dataset nodes. The second stage calculates the importance of entities within a dataset.

We also consider the features of links such as their specificity and cardinality to assign weights. In our experimental setup, we use the Link Frequency - Inverse Dataset Frequency (LF-IDF) weighting scheme [3] on linksets between datasets or on single links between entities.

We compute static entity ranks using the Sindice data repository instead of the original BTC dataset. While the Sindice data repository and the BTC dataset are largely overlapping, the data coming from Sindice are kept up to date and better interlinked. Sindice is more representative subset of the Web of Data.

DatasetRank. The dataset surfing behaviour is the same as in PageRank. We can obtain the importance of dataset nodes by applying a weighted PageRank on the dataset graph. The DatasetRank formula is given in [3].

Weighted EntityRank. The Weighted EntityRank method uses the PageRank algorithm applied on the internal entities and intra-links of a dataset in order to compute the importance of an entity node within a dataset.

Weighted LinkCount. The Weighted LinkCount is a variant of the *in-degree counting links* method [4], an alternative to EntityRank when the dataset can be assumed mostly deduplicated and spam-free. This is often true for very well curated user-input datasets like DBpedia.

Combining DatasetRank and Entity Rank. The local entity ranks are much higher in small datasets than in larger ones, since in the probabilistic model all ranks in a dataset sum to 1. As a consequence any small dataset receiving even a single link is likely to have its top entities score way above many of the top ones from larger datasets. The solution is to normalize the local ranks to a same *average* based on the dataset size. In our experiments we use the formula as described in [3].

Global Weighted PageRank. The baseline algorithm that we use for comparison is a global version of EntityRank (GER). This algorithm is similar to the *Weighted EntityRank*

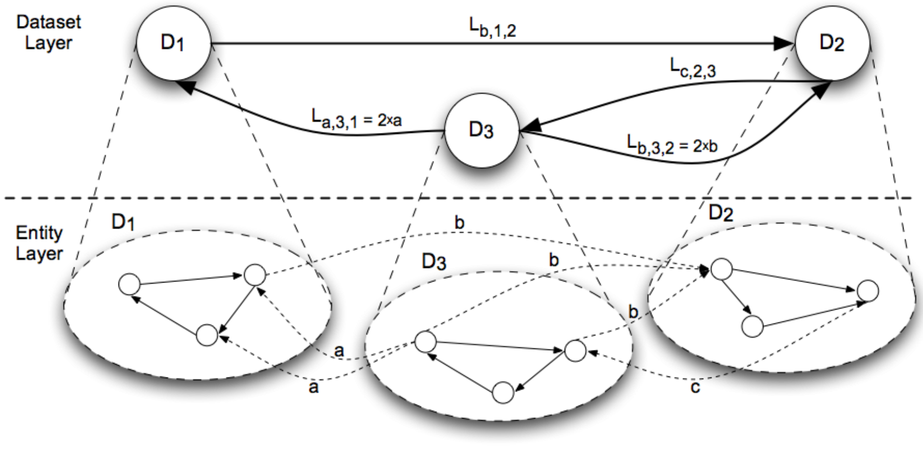


Figure 2: The two-layer model of the Web of Data. Dashed edges on the entity layer represent inter-dataset links.

with the only difference that it operates on the full Web of Data graph G.

Previous evaluations [3] have shown that Dataset LinkCount (DLC) and Dataset EntityRank (DER) could be superior to the Global EntityRank.

5. COMBINING RANK EVIDENCE

We combine the query-dependent and query-independent evidence in order to compute the final entity score. We adopt one of the method presented in [1].

In our experiment, we retrieve the full ordered list of entities matching for each query. We integrate the static score S_s with the query score S_q in order to obtain the final entity score S_f. We finally rerank the entity list based on the final score.

The query score is normalized using its logarithm $\log(S_q)$ and the static score using the sigmoid function [1] with parameters $w = 1.8$, $k = 1$ and $a = 0.6$. The final formula is $S_f = \log(S_q) + w * \frac{S_s^a}{k^a + S_s^a}$.

6. CONCLUSION AND FUTURE WORK

We described how the BTC dataset can be processed and ranked using Sindice technologies. We explained the query-dependent ranking strategy which is being currently implemented in SIREn[2], the entity retrieval system at the core of the Sindice semantic search engine. We outlined the query-independent ranking strategy of Sindice and how its result can be combined with the query score in order to get a final entity rank.

Future work will focus on improving the SIREn query-dependent ranking strategy. Currently, there is no document length or field (predicate) length normalization. A strategy like BM25 with field weighting could be adapted for the SIREn and RDF use cases. Also, there is a need for automatically assigning weights to predicates at query time in order to improve effectiveness.

7. ACKNOWLEDGMENTS

This material is based upon works supported by the European FP7 project *Okkam - Enabling a Web of Entities* (contract no. ICT-215032), and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

8. REFERENCES

[1] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 416–423, New York, NY, USA, 2005. ACM.

[2] R. Delbru, N. Toupikov, M. Catasta, and G. Tummarello. A Node Indexing Scheme for Web Entity Retrieval. In *Proceedings of the Extended Semantic Web Conference (ESWC 2010)*, 2010.

[3] R. Delbru, N. Toupikov, M. Catasta, G. Tummarello, and S. Decker. Hierarchical Link Analysis for Ranking Web Data. In *Proceedings of the Extended Semantic Web Conference (ESWC 2010)*, 2010.

[4] M. A. Najork, H. Zaragoza, and M. J. Taylor. Hits on the web: how does it compare? In *Proceedings of the 30th annual international Annual ACM Conference on Research and Development in Information Retrieval*, 2007.

[5] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1), 2008.

[2]SIREn: `http://siren.sindice.com/download.html`