SIREn: Entity Retrieval System for the Web of Data

Renaud Delbru
Digital Enterprise Research Institute
National University of Ireland
Galway, Ireland
renaud.delbru@deri.org

Abstract

We present ongoing work on the Semantic Information Retrieval Engine (SIREn), an "entity retrieval system" specifically designed to meet the requirements of indexing and searching a large amount of semi-structured data, e.g. the entire Web of Data. SIREn supports efficient full text search with semi-structural queries and exhibits a concise index, constant time updates and inherits Information Retrieval features such as top-k queries, efficient caching and scalability via distribution over shards. We demonstrate how SIREn can effectively answer queries over 10 billion triples on single commodity machine. The prototype is currently in use in the Sindice search engine which index at the present time more than 50 million harvested documents containing semi-structured data.

Keywords: Web of Data, Resource Description Framework, Semi-Structured Data, Inverted Index, Scalability

1. INTRODUCTION

The amount of Resource Description Framework[1] (RDF) and Microformats available online has grown tremendously in the past years. RDF and Microformats are specifications that allow web sites to expose semi-structured information for machine reuse, implementing something referred to as the "Web of Data", Semantic Web [13] or even Web 3.0. In a way, the idea behind the Web of Data is to "create a universal medium for the exchange of data where data can be shared and processed by automated tools as well as by people"[2]. Typical uses of Web of Data are automatic interactions with advanced clients, e.g. automatic integration with the user's calendar and contact list by encoding entities using formats such as HCard, FOAF, or VCard, search engine result customization, advanced data mashups, etc.

On the one hand the Linked Open Data[3] community has made available several billion triples equivalent of information, driven by the idea of open access to semi-structured data. On the other hand, an increasing number of relevant Web 2.0 players (LinkedIn, Yahoo Locals, Eventful, Digg, Youtube and Wordpress to name just a few) have also added some form of semi-structured data markups given the support now available in Yahoo with the Searchmonkey project, and in Google with its support for RDFa structured snippets.

Precise measurements of this growth are not available, but partial reports and private communications of which we are aware, estimate it in several billions of RDF triples and approximately half a billion of marked up web pages, likely totalling several billion triples equivalent. Whatever the current size of the Web of Data is today, the trend is clear and so is the requirement for handling semantically structured data with a scalability in the same class of traditional search engines.

In this paper we present the Semantic Information Retrieval Engine, SIREn, a system based on Information Retrieval (IR) techniques designed to search "entities" and exhibiting many characteristics of IR systems such as web like scalability, incremental updates, top-k queries and efficient caching among others.

[1] Resource Description Framework: `http://www.w3.org/RDF/`
[2] Semantic Web Activity Statement: http://www.w3.org/2001/sw/Activity.html
[3] Linked Data: `http://linkeddata.org/`

1.1. Web of Data: Preliminaries

The Web of Data is based on the RDF data model that provides the functionality of making machine understandable statement about any resources. An RDF statement is expressed as a triple (s, p, o) consisting of a subject, a predicate, and an object and asserts that a subject has a property (predicate) with some value (object). Given three infinite sets U, B and L called respectively URI references, blank nodes and literals, an RDF statement (s, p, o) is an element of $(U \cup B) \times U \times (U \cup B \cup L)$. An RDF statement can be interpreted as a labelled edge where both the subject and object are the nodes and the predicate is a labelled edge connecting the two nodes. While RDF triples are a seemingly simple concept, the true power of RDF lies in the fact that these triples are combined to form a labelled, directed multi-graph, as depicted in Fig. 1a.

The Web of Data is composed of many interconnected RDF graphs, or datasets, each one nameable by an URI. These named graphs [5] are composed of quads. A quad q is a statement (s, p, o, c) with a fourth element $c \in U$ called "context" for naming the RDF dataset in order to keep the provenance of the RDF data.

1.2. Web of Data: Requirements for SIREn

SIREn has been conceived to index the entire "Web of Data". The requirements have therefore been:

1. Support for the multiple formats which are used on the Web of Data;
2. Support for entity centric search;
3. Support for context (provenance) of information: entity descriptions are given in the context of a website or dataset;
4. Support for semi-structural full text search, top-k query, scalability via shard over clusters of commodity machines, efficient caching strategy and real-time dynamic index maintenance.

With respect to point 1, the two formats which enable the annotations of entities on web pages are Microformats and RDF. At knowledge representation level, the main difference between Microformats and RDF is that the former can be seen as a frame model while the latter has a graph based data model. While these are major conceptual differences, it is easy to see that the RDF model can be used effectively to map Microformats[4]. Under these conditions, we have developed SIREn to cover the RDF model knowing that this would cover Microformats and likely other forms of web metadata.

With respect to point 2 and 3, the main use case for which SIREn is developed is entity search: given a description of an entity, i.e. a star-shaped queries such as the one in Fig. 1b, locate the most suitable entities and datasets. This means that, in terms of granularity, the search needs to move from "page" (as per normal web search) to a "dataset-entity". The Fig. 1a 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.

Finally, we will see in Sect. 2 that the SIREn model enables dataset-entity centric search while leveraging well known IR techniques to address the point 4.

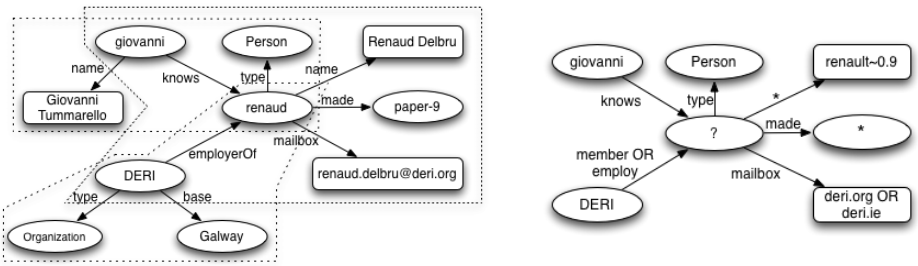
1.3. Approaches for Entity Retrieval

Given an query, an Entity Retrieval System (ERS) helps to locate and retrieve a list of relevant entities. An ERS should allow "imprecise" or "fuzzy" queries and rank the results based on their relevance to the queries. The entity retrieval task is *selection-oriented*, the aim is to select potential relevant entities (e.g. the top-k most relevant ones) from a large entity collection. Two main approaches have been taken for entity retrieval, DBMS based and IR based.

1.3.1. DBMS based approaches

Typically, entities described in RDF data are handled using systems referred to as "triplestores" or "quadstores" and that usually employ techniques coming from the DBMS world. Some of these are

[4] Any23: `http://code.google.com/p/any23/`



(a) Visual representation of an RDF graph. The RDF graph is divided (dashed lines) into three entities identified by the node *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.

built on top of existing DBMS such as Virtuoso[5] or column stores [1] while others are purposely built to handle RDF, e.g. [7, 16, 11].

These triplestores are built to manage large amounts of RDF triples or quads and they do so employing multiples indices (generally b-trees) for covering all kind of access patterns of the form `(s,p,o,c)`. As for DBMS, the main goal of these systems is answering possibly complex queries, e.g. those posed using the SPARQL query language[6]. This task is a superset of entity retrieval as we defined it, and can be seen as *transformation-oriented* since DBMS provide functionality to select entities but also to transform the result set (e.g. create a new RDF graph). This comes at the cost of maintaining complex data structures and index duplication. Also they usually do not support "imprecise" and top-k queries, but instead return all results that precisely match the query (similarly to SQL query in relational databases).

1.3.2. Information Retrieval for Semi-Structured Data and RDF

Information Retrieval (IR) techniques [2, 9] offer tools to overcome such limitations and have been shown by modern search engines to scale to the size of the web. For these reasons, recent systems [17, 3] have started to explore IR also for searching RDF data.

SIREn continues on this trend of research by proposing a variant of a word level inverted index based on a node-labelled tree data structure. Such technique is coming from IR for semi-structured text [10] such as XML document. Node labelling schemes [14] have been developed to optimise retrieval of XML search engines since they provide an efficient way to encode and query the tree structure of an XML document. In this paper, we propose to adapt such technique to the RDF data model.

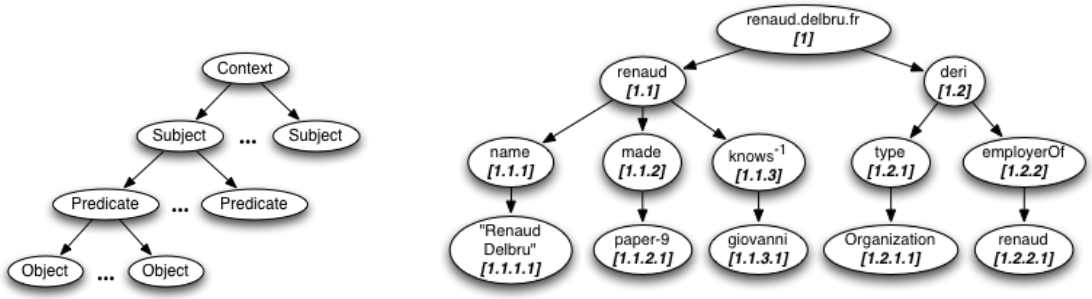
The paper is organized as follows: we first present the index data model in Sect. 2.1 and the associated query model in Sect. 2.2. We report in Sect. 2.3 a short overview of the results of the current scalability evaluation.

2. THE SIREN MODEL

An entity description is a set of triples having one common node and can be depicted as a star-shaped graph. The simplest semi-structured indexing method is to represent an entity as a set of attribute-value pairs using field-based approach [8]. With field-based indexing, index terms are constructed by concatenating the field name with the terms from the content of this field. For example, if a publication has a field *title*, *author* and *abstract*, the index terms for the author field will be represented as *author:renaud* and *author:delbru*.

While this technique is widely used, it has strong limitations when dealing with large amount of heterogeneous semi-structured data:

[5]Virtuoso: `http://virtuoso.openlinksw.com/`
[6]SPARQL: `http://www.w3.org/TR/rdf-sparql-query/`



(a) Conceptual representation of the data tree model of the index

(b) Node-labelled data tree of the example dataset using Dewey's encoding

FIGURE 2: The SIREn data model

1. It is very inefficient to search across all the fields. A single query term will be expanded into a disjunction of n index terms with n the number of different fields.
2. The lexicon becomes prohibitly large, and therefore index term lookups becomes expensive. Many identical terms can appear in various fields, but will be considered as different terms by the system. For example, if m terms appear in n fields, it will produce $m * n$ index terms.
3. Multi-valued fields cannot be handled properly. At query time, we cannot differentiate if an index term belongs to the first or second value of a field. This causes false-positive when using term conjunction. For example, if a publication has two authors, e.g. *Renaud Delbru* and *Giovanni Tumarrello*, then the query *author:renaud AND author:tumarrello* will return a match which is not the expected behaviour.

In order to overcome these limitations, we designed an inverted index based on a tree-structured data model that enables versatile star-shaped querying while enjoying efficient use of disk space, effective compression, fast dynamic updates and sub-linear query processing.

2.1. SIREn Data Model

SIREn, similarly to XML information retrieval engine, adopts a tree data structure and orderings of tree nodes to model datasets, entities and their RDF descriptions. The data tree model is pictured in Fig. 2a. This model has a hierarchical structure with four different kind of nodes: context (dataset), subject (entity), predicate and object. Each node can refer to one or more terms. In case of RDF, a term is not necessarily a word (as in part of an RDF Literal), but can be an URI or a local blank node identifier.

Inverted index based on tree data structure enables to efficiently establish relationships between tree nodes. There are two main types of relations: *Parent-Child* (PC) and *Ancestor-Descendant* (AC). To support this set of relations, the requirement is to assign unique identifiers (node labels) that encodes relationships between the tree nodes. Several node labelling schemes have been developed and the reader can refer to [14] for an overview of them. In the rest of the paper, we will use a simple prefix scheme, the *Dewey Order* encoding [4], but the model is not restricted to the Dewey scheme and another scheme (e.g. interval-based) could be used instead.

Using this labelling scheme, structural relationships between elements can be determined efficiently. An element *u* is an ancestor of an element *v* if label(*u*) is a prefix of label(*v*). Fig. 2b presents a data tree where nodes have been labelled using Dewey's encoding. Given the label $\langle 1.2.1.1 \rangle$ for the term `Organisation`, we can efficiently find that its parent is the predicate `rdf:type`, labelled with $\langle 1.2.1 \rangle$.

The data tree structure with PC and AC relations covers the quad relations CSPO (outgoing relations) and COPS (incoming relations). Incoming relations are symbolised by a predicate node with a $^{-1}$ tag in Fig. 2b. The tree data structure is not limited to quad relations, and could in theory be used to encode longer paths such as 2-hop outgoing and incoming relations.

2.2. SIREn Query Model

Since RDF is semi-structured data, we expect three types of queries: 1. full-text search (keyword based), 2. semi-structural queries (complex queries specified in a star-shaped structure), 3. or a combination of the two (where full-text search can be used on any part of the star-shaped query). We present in this section a set of query operators over the content and structure of the data tree that cover the three types of queries.

2.2.1. SIREn Operators

Content operators The content query operators are the only ones that access the content of a node, and are orthogonal to the structure operators. They include extended boolean operations such as boolean operators (intersection, union, difference), proximity operators (phrase, near, before, after, etc.) and fuzzy or wildcard operators.

These operations allow to express complex keyword queries for each node of the tree. Interestingly, it is possibly to apply these operators not only on literals, but also on URIs (subject, predicate and object), if URIs are normalized (i.e. tokenized). For example one could just use an RDF local name, e.g. `name`, to match `foaf:name` ignoring the namespace.

Structure operators In the following, we define a set of operations over the structure of the data tree. Thanks to these operations, we are able to search content to limited tree nodes, to query node relationships and to retrieve paths of nodes matching a given pattern. Joins over paths are possible using set operators, enabling the computation of entities and datasets matching a given star-shaped query.

Ancestor-Descendant: A//D A node A is the ancestor of a node D if it exists a path between A and D. For example, the SPARQL query in Listing 1, line 1, can be interpreted as an Ancestor-Descendant operator, line 2, and will return the path $\langle 1.2.2.1 \rangle$.

Parent-Child: P/C A node P is the parent of a node C if P is an ancestor of C and C is exactly one level above P. For example, the SPARQL query in Listing 1, line 3, can be translated into a Parent-Child operator, line 4, and will return the path $\langle 1.1.1.1 \rangle$.

Set manipulation operators These operators allow to manipulate nodes of the tree (context, subject, predicate and object) as sets, implementing union ($\cup$), difference ($\setminus$) and intersection ($\cap$). For example in Listing 1, the SPARQL query, line 5, can be interpreted as two Parent-Child operators with the intersection operator (AND), line 6.

In addition, operators can be nested to express longer path as shown in Listing 1, line 7 and 9. However, the later is possible only if deeper trees have been indexed, i.e. 2-hop outgoing and incoming relations of an entity.

Listing 1: SPARQL queries and their SIREn interpretation

```
1  SELECT ?g WHERE { GRAPH ?g { deri ?p renaud }}
2  deri // renaud
3  SELECT ?g ?s WHERE { GRAPH ?g { ?s name "Renaud Delbru" }}
4  name / "Renaud Delbru"
5  SELECT ?g ?o WHERE { GRAPH ?g { giovanni knows ?o . deri employerOf ?o . }}
6  knows^-1 / giovanni AND employerOf^-1 / deri
7  SELECT ?s WHERE { GRAPH <renaud.delbru.fr> { ?s knows renaud }}
8  renaud.delbru.fr // knows / renaud
9  SELECT ?g ?s WHERE { GRAPH ?g { ?s employerOf ?o . ?o name "renaud" . }}
10 employerOf // name / "renaud"
```

2.2.2. SPARQL Interpretation

In this section we discuss the extension by which, given the above discussed operators, it is possible to support a subset of the standard SPARQL query language.

By indexing outgoing relations alone, we can show to cover the quad access patterns listed in Table 1. A quad lookup is performed using the tuple operators. Join operations over these patterns

SPOC	POCS	OCSP	CPSO	CSPO	OSPC
(?,*,*,?)	(?,p,*,?)	(?,*,o,?)	(?,*,*,c)	(s,*,*,c)	(s,*,o,?)
	p	*o*	*c*	*c/s*	*s//o*
(s,*,*,?)	(?,p,o,?)	(?,*,o,c)	(?,p,*,c)	(s,p,*,c)	
s	*p/o*	*c//o*	*c//p*	*c/s/p*	
(s,p,*,?)	(?,p,o,c)	(s,*,o,c)			
s/p	*c//p/o*	*c/s//o*			
(s,p,o,?)					
s/p/o					

TABLE 1: Quad patterns covered by outgoing relations and their interpretation with the SIREn operators. The *?* stands for the elements that are retrieved and the * stands for a wildcard element.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Time (s)	0.75	1.3	1.4	0.5	1.5	1.6	4	0.35
Hits	7552	9344	3.5M	57K	448	8.2M	20.7M	672

TABLE 2: Querying time in seconds and number of hits for the 10 billion triples benchmark

are also feasible. Intersection, union and difference between two or more quad patterns can be achieved efficiently using set manipulations over tree nodes.

The covered quad patterns are a subset of the quad patterns covered by conventional RDF data management systems [6]. In fact, they give the ability to retrieve information about variables that are restricted to be at the subject, object or context position.

It is important to underline however that we are restricting the search of an entity inside a dataset, i.e. SIREn does not allow the use of joins over different contexts, and the intersection of quad patterns within an entity, i.e. SIREn does not allow the use of chains of joins among multiples entities. This limits the query expressiveness to a star-shaped query, e.g. in Fig. 1b.

2.3. Evaluation

We finally report some performance and scalability benchmarks. Indexing time for a synthetic dataset (120GB, 1 billion triples) took only 31 minutes while keeping constant time update and a relatively concise index (15GB) due to efficient compression using word-aligned binary codes. On a term-interleaved inverted index, we achieved a compression average of less than one byte per integer using Simple-9. Query time execution of various star-shaped queries performed at the same order of magnitude than the state-of-the-art triple store RDF-3X [11].

We evaluate SIREn scalability by indexing a dataset composed by 1 billion entities described in approximately 10 billion triples. The dataset is derived from the billion triple challenge dataset[7]. The machine that served for the experiment was equipped with 8GB ram, 2 quad core Intel processors running at 2.23 Ghz, 7200 RPM SATA disks, linux 2.6.24-19, Java Version 1.6.0.06 and GCC 4.2.4.

The following benchmark was performed with cold-cache by using the `/proc/sys/vm/drop_caches` interface to flush the kernel cache and by reloading the application after each query to bypass the application cache. The set of queries is provided at `http://siren.sindice.com`. The performance is given in the Table 2.

3. CONCLUSION

In this paper, we presented the current state of SIREn, an Entity Retrieval System for the Web of Data. The main challenge that has been discussed in this paper is the design of an efficient inverted index especially designed for answering semi-structured (star-shaped) queries while preserving desirable IR features, such as web like scalability, incremental updates, top-k queries and efficient caching among others.

This work has been undergoing for approximately 18 months now and is at different stages of experimentation, validation and ultimately deployment. For example, SIREn presented in this paper

[7] Semantic Web Challenge: `http://challenge.semanticweb.org/`

is currently in use in the Sindice search engine [12] and will be released as an open source project[8]. The system has been evaluated against other existing systems such as RDF-3X, and the results will be published soon. The current work on SIREn is going towards the design of a new labelling scheme extending the basic tree model to a directed acyclic graph, hence enabling more flexible semi-structured queries. Due to space constraint, we limited the discussion to the SIREn model and omitted another aspect of the research work: improving the search quality using link analysis on the Web of Data. An early paper [15] about dataset ranking has been published and the complete ranking system has been finalised and qualitatively evaluated. A paper presenting the results will be published soon.

REFERENCES

[1] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 411–422. VLDB Endowment, 2007.

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[3] H. Bast, A. Chitea, F. Suchanek, and I. Weber. ESTER: efficient search on text, entities, and relations. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference*, pages 671–678, New York, NY, USA, 2007. ACM.

[4] K. Beyer, S. D. Viglas, I. Tatarinov, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and querying ordered xml using a relational database system. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of Data*, pages 204–215, New York, NY, USA, 2002. ACM.

[5] J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.

[6] A. Harth and S. Decker. Optimized index structures for querying rdf from the web. In *LA-WEB*, pages 71–80, 2005.

[7] A. Harth, J. Umbrich, A. Hogan, and S. Decker. YARS2: A Federated Repository for Querying Graph Structured Data from the Web. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, volume 4825 of *Lecture Notes in Computer Science*, pages 211–224. Springer Verlag, November 2007.

[8] R. W. Luk. A survey in indexing and searching XML documents. *Journal of the American Society for Information Science and Technology*, 53(6):415, 2002.

[9] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[10] G. Navarro and R. Baeza-Yates. A language for queries on structure and contents of textual databases. In *SIGIR '95: Proceedings of the 18th international conference on Research and Development in Information Retrieval*, page 93, New York, NY, USA, 1995. ACM.

[11] T. Neumann and G. Weikum. RDF-3X - a RISC-style Engine for RDF. *Proceedings of the VLDB Endowment*, 1(1):647–659, 2008.

[12] 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.

[13] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[14] H. Su-Cheng and L. Chien-Sing. Node Labeling Schemes in XML Query Optimization: A Survey and Trends. *IETE Technical Review*, 26(2):88, 2009.

[15] N. Toupikov, J. Umbrich, R. Delbru, M. Hausenblas, and G. Tummarello. DING! Dataset Ranking using Formal Descriptions. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009.

[16] C. Weiss, P. Karras, and A. Bernstein. Hexastore - sextuple indexing for semantic web data management. *Proceedings of the VLDB Endowment*, 1(1):1008–1019, 2008.

[17] L. Zhang, Q. Liu, J. Zhang, H. Wang, Y. Pan, and Y. Yu. Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, volume 4825 of *Lecture Notes in Computer Science*, pages 652–665. Springer Verlag, November 2007.

[8] SIREn: `http://siren.sindice.com/`