SEMDEX: Yet Another Semantic Web Search Engine

Olufemi-Allen Akinkunle (31752066)

***Abstract*—In this paper, we discuss the design and implementation of a linked data person search engine. Similar to traditional search engines, this semantic web search engine crawls, indexes and provides a web-based user interface to retrieve people on the Semantic Web. We review related works, discussing how others have attempted to solve the problem of information retrieval on the Semantic Web. We also go into details about the design and implementation of the individual components of the search engine. Later, we conclude with discussions about our approach and future research directions.**

I. Introduction

The World Wide Web is a large collection of documents and other resources identified through Uniform Resource Locators (URL) accessed through the Internet. These documents, called web pages, mostly contain textual information marked up with Hypertext Markup Language (HTML). Humans can read the information contained in web documents, but this is difficult for machines to do. This is because of noise in natural language and the complexity of the document structure of these web documents. To make machines understand the information contained within web documents and aid them in extracting the information accordingly, explicit instructions must be added to the web documents to instruct the machine what the information denotes.

Addressing this problem of adding semantics to information within web documents, the Semantic Web movement provides technologies for publishing machine-readable data on the web. The core technology is Resource Description Framework (RDF). RDF uses Uniform Resources Identifiers (URI) to identify information and entities within documents as well as relationships between these entities [1]. These entities and the relationships between them are defined in statements comprising of a subject, predicate and an object. This subject-predicate-object statement is called a triple. A collection of RDF statements is called a RDF document. RDF can be embedded into web documents using RDFa [1][2], such that the data can be linked, shared and reused across applications and enterprise boundaries.

There has been a rise in the number of RDF data on the web due to the availability of tools and standards like RDF and OWL (Web Ontology Language) for publishing these semantic data [1][3]. A popular example is DBpedia, which is a collection of RDF documents extracted from Wikipedia. DBpedia uses RDF to describe the entities in Wikipedia articles and their properties. The RDF data can be accessed through SPARQL queries which is a SQL-like language for querying RDF documents. In line with the vision of the Semantic Web of sharing data across applications, DBpedia is linked with external RDF datasets like GeoNames and US Census data.

Given this increase in the amount of RDF datasets on the web, it is imperative to provide a way to find and discover this data through a semantic web search engine. Following the linked data approach that all items should be identified using URI references, the search engine would crawl the Semantic Web, following resource URIs and indexing the found resources [2].

This paper introduces SEMDEX, a linked data person search engine that crawls the Semantic Web, finding resources of type 'Person', indexing these found resources. It indexes these found resources on resource URIs, human-readable labels of the resources and in line with linked data approach of linking semantic data, it keeps a list of resources that are linked to the found Person resource. It also provides a web-based user interface through which human users can find these resources.

In the following sections of this papers, we discuss the related work in developing Semantic Web search engines, present our methodology of our linked data person search engine, detail the evaluation process of our work and conclude with discussions of future directions and current limitations of our work.

II. Related Works

There has been a lot of work around building linked data search engines and in this section, we give an overview of these systems that crawl and index the Semantic Web.

A. WebOWL

WebOWL is a semantic web search engine for OWL data. It crawls the Web, looking for sites that contain ontologies and extracts the entities within these entities. It uses a generic OWL database to store the found entities and uses a ranking algorithm that ranks the classes and individuals in the database differently. WebOWL does not use an indexing solution like Apache Lucene or Solr so query from the database might be slow depending on the complexity of the query [4].

B. Swoogle

Swoogle is a semantic web search engine that crawls and indexes entities on the Semantic Web. It extracts metadata for each discovered entity, and computes the relationships between these entities. It uses a relational database to store the found entities. To allow quick search of these entities, they are indexed. It is uses a ranking algorithm that ranks the documents based on the relationships between them [5].

C. Sindice

Sindice is a Semantic Web search engine that crawls the web to find semantic web resources. It indexes these documents using inverse-functional properties, resource URIs and human-readable labels and provides a user interface and API to allow users and applications to locate documents containing information about a given resource. It uses a distributed architecture to scale the system, Apache Jena to crawl the Semantic Web and Apache Solr to index the documents [6].

D. SWSE

Like the other search engines discussed, SWSE crawls the Semantic Web, indexing found documents and provides a user interface for searching, browsing and retrieval of these documents [7]. Like Sindice, it also uses a distributed architecture (Apache Hadoop) to scale the system.

III. METHODOLOGY

In this section, we explain our approach and how we went about building a linked data person search engine. Our system had three functional requirements:

- Crawl the Semantic Web, finding resources of type 'Person'
- Index the found resources so as to provide fast look-up using resource labels and Uniform Resource Identifiers (URIs)
- Provide a web-based user interface where users can search, browse and retrieve Semantic Web documents.

To fulfil these requirements, our system is influenced by traditional web search engines and the Sindice Semantic Web search engine [2]. We built the different parts of the systems as modules that can run standalone. The modules are the Crawling Indexing module and the Web interface module.

Because this system is intended to demonstrate the feasibility of building a linked data search engine, we decided to only crawl the DBpedia SPARQL endpoint, indexing resources of type 'Person'. The DBpedia dataset is very large and contains about 1,445,000 person entities [8].

A. CRAWLING AND INDEXING MODULE

The crawling and indexing module is built using Python's rdflib library for manipulating RDF data. The crawler starts by accepting a seed URL, runs a SPARQL query of the type:

$$< URI > ?predicate\ ?resources \quad (1)$$

against the DBpedia SPARQL endpoint to find all the resources that are objects of the resource referenced by the seed URL. All these found resources (each identified by a URI in accordance with linked data approach) that are objects of the seed URL resources are added to a to-crawl list. For each of the found resources, another SPARQL query is run to get all the resources of type 'Person' that are objects of the URL.

For each found person, we index it using its human-readable label (rdfs:label), its URI and all the resources that it is linked to. We use Elasticsearch, which is a search server built on top on Apache Lucene. We decided to use Elasticsearch over Apache Lucene or Solr because it abstracts away all the difficulties of working directly with Apache Lucene's Java API.

B. WEB INTERFACE MODULE

The web interface module is the main entry point of the application to the end user. The user can search for people on the crawled documents in our Elasticsearch index using text labels and URIs. The web interface was built using Flask, which is a micro web framwework written in Python.

Fig. 1. The web frontend to Semdex

Figure 2 shows the documents matched after the user searched using the keyword 'David'. This matches all the documents with people named 'David'

Figure 3 shows the details of a particular resource after it is clicked from the result list. It also shows the documents that are linked to the resource.

IV. EVALUATION

RDF documents that match certain criteria can be found on the Semantic Web using SPARQL endpoints like the one provided by DBpedia. SPARQL queries are run against these endpoints and they return results in a variety of formats. However, these SPARQL queries can be slow as the queries become complex. In traditional databases, one of the ways to increase the speed of information retrieval is through the use of an index. One of the goals of building Semdex is to build an index that will provide fast look-up for the documents available on the Semantic Web, albeit on a much lesser scale.

To test the notion that searching against an index is faster than running native SPARQL queries against SPARQL endpoints, we ran a benchmark experiment to see the difference in the time taken to run both type of queries.

We ran about 100 queries of the type:

$$FIND\ documents\ with\ name\ LIKE\ 'query' \quad (2)$$

against our Elasticsearch index and against the DBpedia SPARQL endpoint. Our results show that, on average, it takes

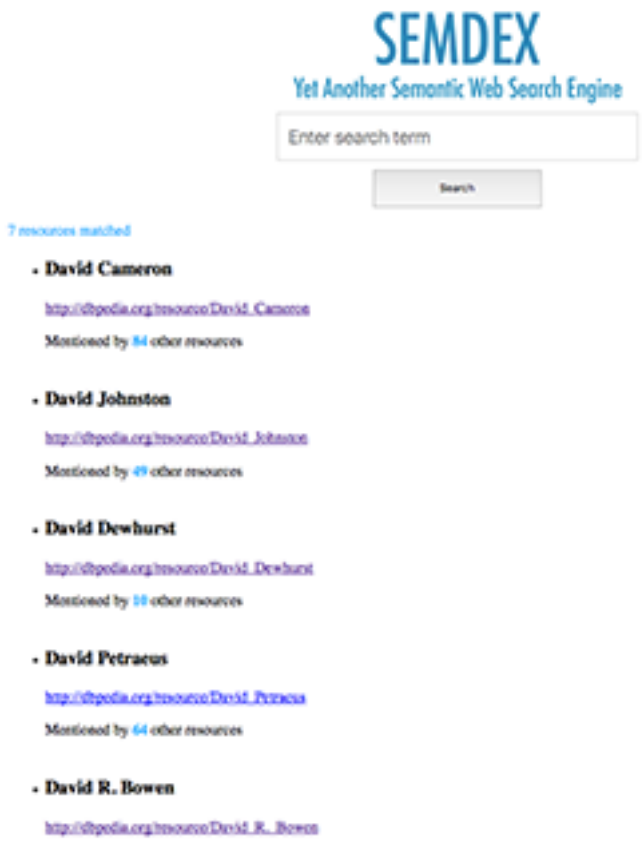

Fig. 2. Results for keyword search 'David'

Fig. 3. Page showing details of document. It shows the documents that mentions the source

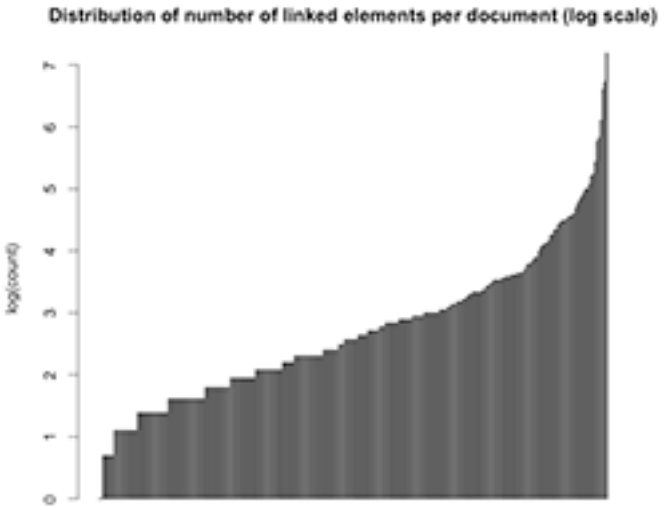

Fig. 4. Distribution of number of linked elements per document (log scale)

37.2 milliseconds to run such queries against our index while it takes about 11000.3 milliseconds to run the queries against the DBpedia endpoint. It is worth pointing out that the index is running on our local machine so there is no network latency that will delay the queries. Nonetheless, there is a huge difference in the times the queries take.

V. FINDINGS

One of the principles of the linked data initiative is that resources on the Semantic web should be linked to other resources to enable sharing and linking. We plotted the distribution of the number of resources each resource in our index is linked to. Figure 4 shows the plot of this distribution on the log-scale.

We find that all the resources are linked to other resources with each one linked to an average of about 40 other resources. We also find from our evaluation of the system that it is faster to run queries against an index of resources than against a SPARQL endpoint.

VI. CONCLUSIONS

In this paper, we presented a linked data person search engine, detailing our methodology and evaluation. Our work was influenced by existing work in the space [2]. Our approach worked but it has a number of limitations. It does not use caching in any way in the system. The use of caching could considerably reduce the look-up time from our Elasticsearch index.

There is a lot of work going on in the Semantic Web space and for the future, we would like to build a distributed version of this system using frameworks like Apache Hadoop. We would also like to expand this system to crawl the Semantic web for every type of resource, not limited to the type of Person as we have currently done it.

REFERENCES

[1] Hogan, A., Harth, A., Umbrich, J., Kinsella, S., Polleres, A. Decker, S. 2011, "Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 9, no. 4, pp. 365-401.

[2] Tummarello, G., Delbru, R. Oren, E. 2008, "Sindice.com: Weaving the Open Linked Data" in Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 552-565.

[3] Delbru, R., Campinas, S. Tummarello, G. 2012, "Searching web data: An entity retrieval and high-performance indexing model", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 10, pp. 33-58.

[4] Batzios, A. Mitkas, P.A. 2012, "WebOWL: A Semantic Web search engine development experiment", Expert Systems with Applications, vol. 39, no. 5, pp. 5052-5060.

[5] Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R., Peng, Y., Reddivari, P., Doshi, V. Sachs, J. 2004, "Swoogle: a search and metadata engine for the semantic web", ACM, , pp. 652.

[6] Tummarello, G., Delbru, R. Oren, E. 2008, "Sindice.com: Weaving the Open Linked Data" in Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 552-565.

[7] Hogan, A., Harth, A., Umbrich, J., Kinsella, S., Polleres, A. Decker, S. 2011, "Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 9, no. 4, pp. 365-401.

[8] Dbpedia.org. (2016). DBpedia. [online] Available at: http://dbpedia.org [Accessed 21 May 2016]