Keyword-Based Navigation and Search over the Linked Data Web

Luca Matteis
Department of Computer Science
Sapienza University of Rome
matteis@di.uniroma1.it

Roberto Navigli
Department of Computer Science
Sapienza University of Rome
navigli@di.uniroma1.it

ABSTRACT

Keyword search approaches over RDF graphs showed that it is possible to answer questions against structured data using keywords. However, these approaches rely on local copies of RDF graphs which limits their potential use. In this paper we present an algorithm that uses RDF keyword search methodologies to answer questions against the live linked data web, rather than against local indexes. To do this we navigate along a path of keywords, and search for triples along the way. The navigation is performed through a pipeline which streams results to users as soon as they are found, while the search is assisted through the resolution of predicate URIs. We evaluated this methodology by converting several natural language questions into lists of keywords and seed URIs. For each question we measured how quickly and how many triples appeared in the output stream of each step of the pipeline. Results show that relevant triples are streamed back to users in less than 5 seconds. We think this approach can help people analyze and explore various linked datasets in a *follow your nose* fashion by simply typing keywords.

1. INTRODUCTION

Browsing and navigating through URIs that come from different sources is the main characteristic of the web of documents. For the web of data (also known as *Linked Data*) we now have more granular access to resources published on web, which, contrary to documents, are better structured and linked to each other with specific type information. This increased granularity has opened the doors to new approaches for browsing [1] and querying [2, 3] this information space. One interesting approach allows users to perform controlled navigation of the web through a formal language called NautiLOD which can be used to follow routes and therefore answer specific queries [4]. With these methods we are getting closer to answering complicated questions against the live linked data web, and not just against a crawled, and hardly up-to-date, index.

Figure 1: Navigating the web of documents involves clicking through various URIs coming from different sources.

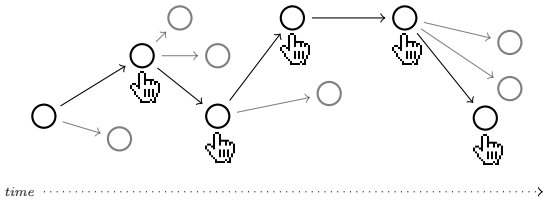
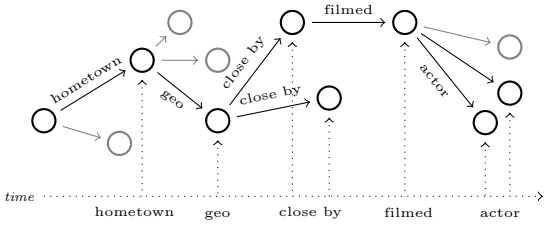


Figure 2: Navigating the web of data using our approach involves typing keywords rather than clicking.



A problem with these link traversal approaches is that users must be aware of the portion of the web they want to query or navigate: (i) they need to provide an *upfront* formal description of the portion of the web they want to traverse and (ii) they need to know which URIs to follow at each step of the navigation. This is also true for SPARQL endpoints: to write queries users must be aware of the structure of the data they want to query. Furthermore a single query is not going to provide a complete answer: one must try different combinations of data sources and URIs in order to obtain a more complete picture.

We call for a new approach which instead of requiring upfront formal definitions of the querying or navigation users want to carry out, we enable users to perform their discoveries "on the go" using keywords. This approach takes advantage of the main property of linked data, where URIs should always *resolve* to some useful information. Compared to simply browsing the web, such as clicking and resolving various links manually, our keyword-based approach finds the links in a resource based on their type, and therefore allows for a more structured navigation. Figure 1 and 2 show a comparison between browsing the web of documents and browsing the web of data using our keyword-based approach.

Example 1. *What are the interests of the people known by Andreas Harth?*

Excerpt of the navigation showing the contents of the stream as the navigation proceeds along the keyword pipeline.

Seed URIs = `http://harth.org/andreas/foaf#ah`
Keywords = known, interest

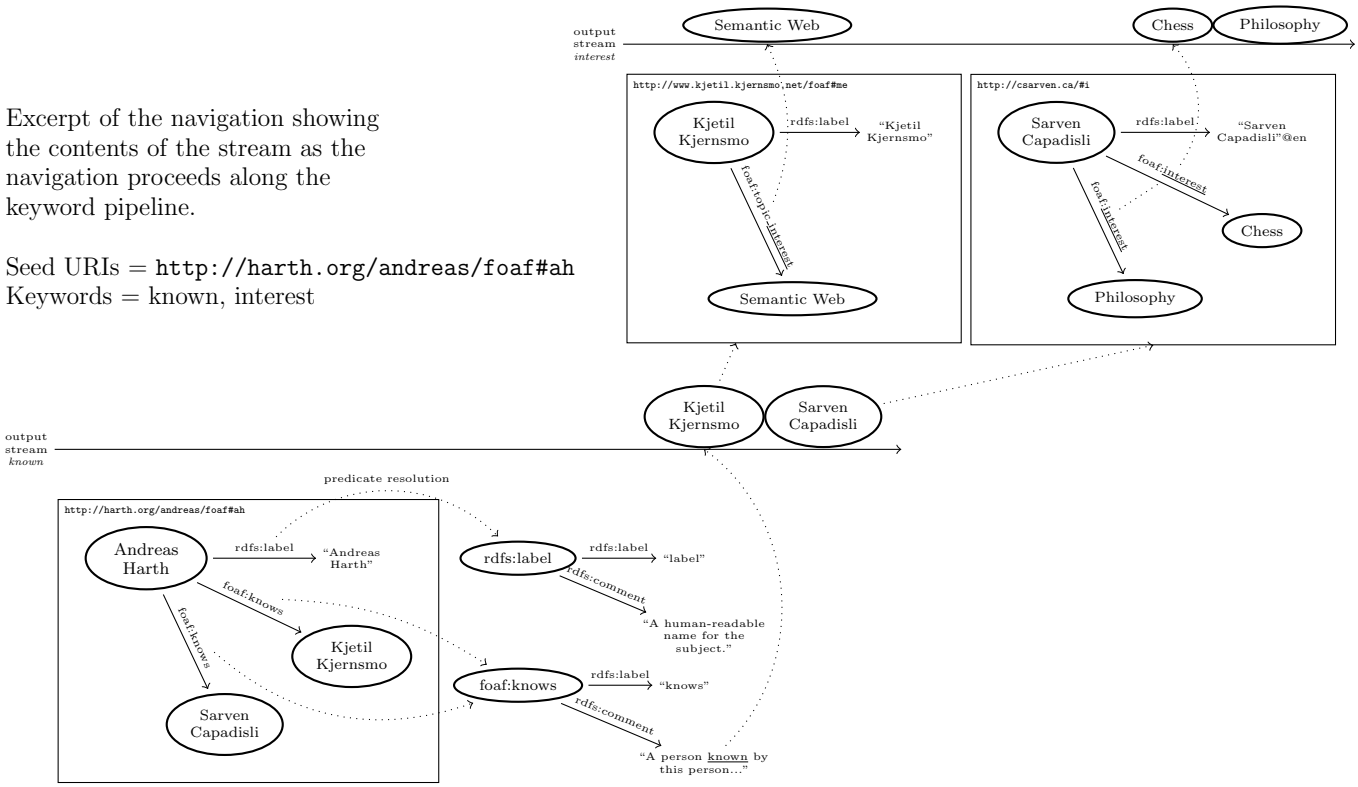
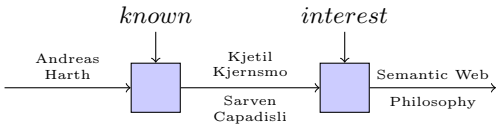


Figure 3: Navigation using a streaming pipeline



2. KEYWORD-BASED NAVIGATION AND SEARCH

Our approach combines the most interesting aspects of (i) Linked Data navigation, such as link traversal techniques [2] and semantic controlled navigation [4], and (ii) RDF graph search methodologies [5]. In fact we think that combining these two approaches of navigating through a set of URIs and searching for relevant information along the way, can lead to answering important questions directly over the linked data web. Although one can still answer questions against the web of data using query languages such as SPARQL, we believe that many questions remain unanswered largely due to the fact that one single upfront query cannot possibly return all the relevant information. At the same time one cannot really download the whole web of data, and although several search engines exist that do this [6], they still require users to understand the underlying data structures in order to let them formulate queries.

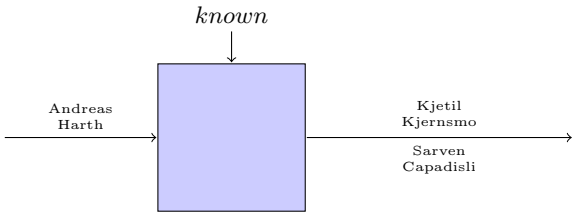
To navigate and search the web of data using our approach, users (or agents) provide a keyword and seed URIs as inputs. The navigation starts by resolving the seed URIs and using the keyword to search for relevant URIs within the triples returned. Users can then at any time provide another keyword to search against the newly found information. This process is visualized in Example 1 where we start the navigation with seed URI `http://harth.org/andreas/foaf#ah` and keyword "known". URIs found are sent to an output stream. In the upper-right corner we see this entire process happening again, but with keyword "interest" and with seed URIs being the ones returned from the earlier step.

2.1 Navigation

The type of navigation we use to browse the linked data web in a programmatic way is the one similar to other work such as LDpath [] and NautiLOD [4]. These approaches rely specifically on following RDF links between resources and servers. The main difference is that our approach returns results to users as quickly as possible in a streaming fashion. Furthermore, the navigation is driven by keywords rather than explicit URIs or regular expressions, allowing for a more flexible and less rigorous understanding of the underlying data structures.

Figure 3 represents the process we had in Example 1 in the form of a *pipeline*, where the navigation path is effectively a pipeline of keywords. Similar to UNIX pipelines [**?**], each element of the pipeline takes an input stream, and writes to its output stream as soon as relevant triples have been found, which enables relevant triples to be streamed back to users as quickly as possible. If users wants to continue the navigation using a new keyword, they can simply add a new element in the pipeline that listens for changes of its input stream, and results will streamed back as soon as they are found.

Figure 4: Search using predicate resolution



2.2 Search

The process of searching for triples in RDF graphs, using keywords, is a well researched topic. Specifically work such as [5] shows that keyword search against RDF structures is achievable with high accuracy. The main difference is that our approach includes in the search also triples returned by *predicate resolution*. This involves the resolution of the predicate URIs (the type of the RDF links) contained in a set of triples. What is interesting about predicate resolution is that we can use the information returned by the predicate URIs to enrich our search: to match RDF links we are not limited to the information contained in their URI. Figure 4 shows an example of how predicate resolution can enrich a search result.

The combination of our navigation and search is what makes our approach unique and enables answering highly specific questions using keywords. Although our main use case scenario is users wanting to answer questions, we believe our approach could also be utilized by computer agents programmed to navigate specific keyword paths to find relevant information.

2.3 Algorithm

We will now describe more in detail our approach by presenting an algorithm and explaining its functionality using a walkthrough example. The question in Example 1 involves accessing data from different sources and following specific paths based on the people *Andreas* knows. To represent this question formally we define a seed URI, from where the navigation initiates, and an ordered set of keywords we need to follow in order to answer our question. Each keyword is effectively an element in the pipeline. Algorithm 1 details the procedure of a single element of the pipeline.

The algorithm takes as input a stream which we call S, and a keyword k. By following Example 1, once the URI `http://harth.org/andreas/foaf#ah` appears in the stream S (line 1), we resolve this URI and assign the triples returned to the set R (line 2). Next at line 3 we apply our search methodology against the set R using the keyword k (which is "known") and the found triples are assigned to the set F. If triples are found we extract the URIs from the found triples (line 5) and write the resulting URIs to our output stream *out*. For the keyword "known" no triples were found, therefore nothing is written to the output stream.

Next at line 8 we obtain triples that were not yet found, assign them to the set T and we proceed with resolving the predicate URIs of this set. We loop through each triple of this set at line 9, and we resolve each predicate at line 10. We then apply our search methodology against the set P using the keyword "known" and the found triples are assigned to the set F (line 12). If triples are found we extract the URIs from the triple t we are currently iterating (line 13). At line 14 we finally write to the stream *out* the extracted URIs. Thanks to predicate resolution, we are capable of matching the triples with predicate `foaf:knows`, resulting in 2 URIs written to output stream, namely `http://www.kjetil.kjernsmo.net/foaf#me` and `http://csarven.ca/#i`.

Algorithm 1 Keyword-based navigation and search

Input: input stream S and keyword k
Output: output stream *out* of URIs found
1: **upon** URI **in stream** S **do**
2: $R \leftarrow$ triples from resolving URI
3: $F \leftarrow$ search for k in R and return triples
4: **if** $F.length > 0$ **then**
5: $N \leftarrow$ { URIs of F except predicates } $- URI$
6: *out*.write(N)
7: **end if**
8: $T \leftarrow R - F$
9: **for each** triple $t \in T$ **do**
10: $P \leftarrow$ triples from resolving predicate of t
11: $F \leftarrow$ search for k in P and return triples
12: **if** $F.length > 0$ **then**
13: $N \leftarrow$ { URIs of t except predicates } $- URI$
14: *out*.write(N)
15: **end if**
16: **end for**

Example 1 then continues by listening for the keyword "interest". The algorithm explained above is re-executed, this time with an input stream fed from the output stream of the earlier execution. We will therefore obtain a total of 2 URIs in the input stream and for each of them the process is repeated. It is important to note that the process of resolving URIs can occur in parallel. In the evaluation section we specifically create a new thread for each request to obtain higher throughput.

3. EVALUATION

The main characteristic of our keyword-based navigation and search is that it allows to answer questions "as you go" by simply typing keywords. The primary purpose of this evaluation is to verify whether our identified method produces relevant results in an acceptable time frame. To this end, we will perform two different experiments:

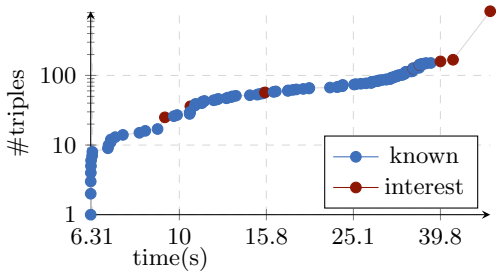
1. First we will execute a series of questions, each represented as a set of keywords with a seed URI, using our identified approach. We will measure (i) how fast and how many relevant triples appear in the output streams, and (ii) how many different web servers were accessed during the navigation.

2. Next we will compare our methodology against a similar implementation called swgetM [], a multi-threaded tool that executes NautiLOD expressions, and we will measure the total execution times and response times at each navigation step for both approaches.

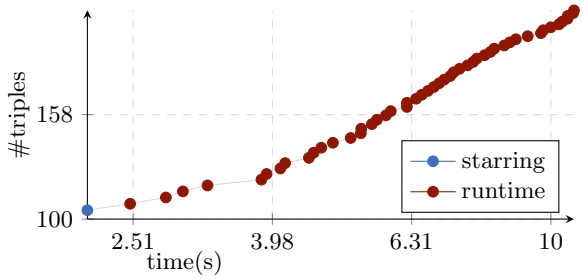
For the first experiment we measured a total of 4 different questions (Fig. **??** and **??**). The question in Fig. **??** and **??** are executed using 2 keywords. While the question in Fig. **??** and **??** are executed using 3 keywords. We chose highly specific questions, that involved accessing resources hosted

Figure 5: Execution of questions

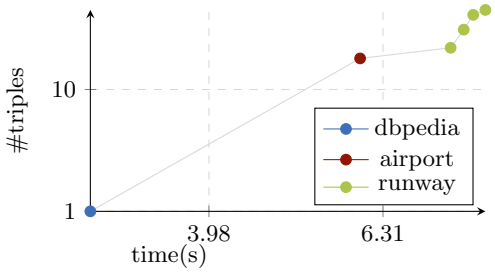
(a) What are the interests of the people know by Andreas Harth? 2 keywords and 1 seed URI: http://harth.org/andreas/foaf#ah



(b) What is the runtime of the movies Keanu Reeves starred in? 2 keywords and 1 seed URI: http://dbpedia.org/resource/Keanu_Reeves



(c) What are the lengths of the runways of the airports in Rome, Italy? 3 keywords and 1 seed URI: http://sws.geonames.org/3169070/



(d) Give me the locations of the movies directed by Ridley Scott, in Italian. 3 keywords and 1 seed URI: http://data.linkedmdb.org/resource/director/8472

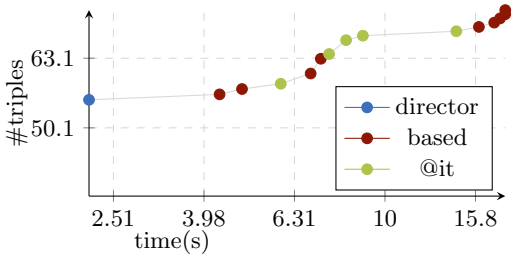


Figure 6: Comparison with swgetM

foo

bar

on different domains and servers, as to also measure the main quality of the linked data web: its distributed nature.

For the second comparison experiment, since NautiLOD works explicitly with URIs, we compared a single question containing 3 different URIs Fig. **??** (in our system a URI is considered as a keyword). It is important to note that swgetM works differently than our methodology, because it blocks at each step of the navigation until it finds all the results at that step. Our approach instead streams the result along the path. For this reason we will measure the total execution time needed to complete each step of the navigation and also the response times needed to retrieve the first result at each step of the navigation. All results are the average of 5 runs.

3.1 Experimental setup

To run the experiments we implemented our navigation pipeline component using the observer pattern. Whereas the search component is a multi-threaded Java process that uses the open source Apache Any23 library to resolve URIs and obtain triples. We did not implement the keyword RDF graph search algorithm proposed here [5], instead we utilized existing substring string similarity implementations to perform full-text keyword search against an RDF graph. For the swgetM comparison we did not take into account our search methodology, simply because we explicitly used URIs. As HTTP requests were run in parallel in different threads, to avoid many simultaneous connections that may overload the servers and generate errors, we used a thread pool size of 5.

3.2 Results

Figures **??** and **??** plot the contents of the stream for each question. Each question is represented using a different color. The smallest circles refer to the first keyword in the path. The largest circles refer to the last keyword in the path. For the first question in Figure **??** we can see that the response time is around 6 seconds. This is mainly due to the fact that it had to resolve the predicate URIs in order to match triples based on the keyword "known". For the second and third questions in Figure **??** however we see that triples are returned relatively quickly in less than a second for the first keywords and around 2 seconds for the last keyword.

We can see that most questions are answered *completely* after an average of 15.4 seconds. However, the response time, which is the time needed to retrieve a first result, averages around 2 seconds. Furthermore we see that the *navigational hop time*, which is the time needed to retrieve results at each step of the navigation, averages around 5.5 seconds. These results show us that navigating step by step the linked data web using keywords is achievable in acceptable time frames. Navigating with many keywords all at once is obviously slow, but that is not the intended use of our methodology which rather wants to allow users and agents to perform their discoveries on the go. Increasing the keywords, and therefore the complexity of the question, still maintains quick response times, while obviously increasing the number of servers accessed and the number of triples returned.

4. CONCLUSION AND FUTURE WORK

In this short paper we wanted to combine RDF keyword search methodologies with linked data link traversal techniques. Results show that combing these two methodologies using our pipeline-based navigation and predicate resolution search, enables users and agents to explore the web of data using keywords. Furthermore, we can answer questions that were previously impractical to answer, without any knowledge of the underlying linked datasets. Our approach of streaming results back to the user as quickly as possible, shows that applications can be developed to make use of our keyword-based approach.

5. REFERENCES

[1] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets, "Tabulator: Exploring and analyzing linked data on the semantic web," in *In Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[2] O. Hartig, C. Bizer, and J. C. Freytag, "Executing SPARQL Queries over the Web of Linked Data," in *Proceedings of the 8th International Semantic Web Conference*, vol. 5823 of *ISWC '09*, (Berlin, Heidelberg), pp. 293–309, Springer-Verlag, 2009.

[3] R. Verborgh, O. Hartig, B. De Meester, G. Haesendonck, L. De Vocht, M. Vander Sande, R. Cyganiak, P. Colpaert, E. Mannens, and R. Van de Walle, "Querying Datasets on the Web with High Availability," in *Proceedings of the 13th International Semantic Web Conference*, vol. 8796 of *Lecture Notes in Computer Science*, pp. 180–196, Springer, Oct. 2014.

[4] V. Fionda, C. Gutierrez, and G. Pirró, "Semantic navigation on the web of data: specification of routes, web fragments and actions," in *Proceedings of the 21st international conference on World Wide Web*, pp. 281–290, ACM, 2012.

[5] S. Elbassuoni and R. Blanco, "Keyword search over rdf graphs," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 237–242, ACM, 2011.

[6] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker, "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, pp. 365–401, Dec. 2011.