Keyword-Based Navigation and Search over the Linked Data Web

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

Aidan Hogan
Dept. of Computer Science
University of Chile
ahogan@dcc.uchile.cl

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

ABSTRACT

Keyword search approaches over RDF graphs have proven intuitive for users. However, these approaches rely on local copies of RDF graphs. In this paper, we present an algorithm that uses RDF keyword search methodologies to find information in the live Linked Data web rather than against local indexes. Users navigate between documents by specifying keywords that are matched against triples. Navigation is performed through a pipeline which streams results to users as soon as they are found. Keyword search is assisted through the resolution of predicate URIs. We evaluate our 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 on average. We think that this approach can help people analyze and explore various Linked Datasets in a *follow your nose* fashion by simply typing keywords.

1. INTRODUCTION

On the traditional "Web of Documents", navigation is characterized by following links (with anchor text) between documents. On the Web of Data (also known as *Linked Data*), documents contain structured resource descriptions, where resources are interlinked with specific properties. This increased granularity has opened the door to new approaches for browsing [1] and querying [2, 3, 4] this information space, navigating links that match user-specified structured patterns. With such methods, we are getting closer to answering complex queries against the live Linked Data web, and not just against a crawled and hardly up-to-date index.

However, users of these link traversal approaches need: (i) to provide a formal description of the portion of the Web they wish to traverse *upfront* and (ii) to know which URIs to follow at each step of the navigation (similar to writing SPARQL queries, for example, where users need to understand the structure of the data). Furthermore, users may often wish to explore different combinations of data sources and URIs to get a more complete picture; they are unlikely to know what navigational steps or what URIs are relevant until they actually encounter them.

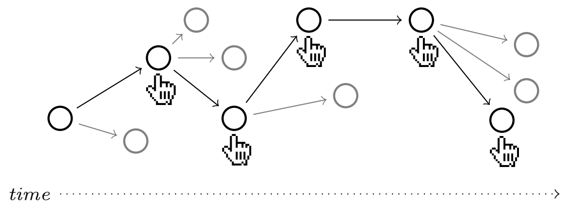


Figure 1: Navigating the Web of Documents involves clicking through various URIs coming from different sources.

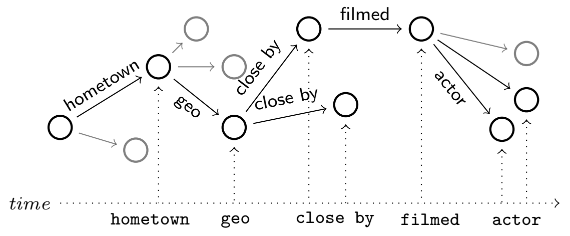


Figure 2: Navigating the Web of Data using our approach involves typing keywords rather than clicking.

In order to avoid the need for users to specify a complex structured query upfront [2, 4] or for local free-text indexes to be built [5, 6], we propose an approach that combines key aspects of (i) Linked Data navigation, such as link traversal techniques [2] and semantic controlled navigation [4]; and (ii) RDF search methodologies [5, 6]. The overall idea is that starting at (a) given location(s) on the Web, the user can interactively specify the next step by typing keywords that are matched against the current data and used to find and traverse links, adding more data. Users can thus use keywords to navigate and answer questions across multiple datasets without needing to know their structure or vocabulary beforehand or without needing to go through a (possibly out-of-date or incomplete) centralized index. All the user needs to start is the seed URL of an initial document.

This paper continues with discussion of related work in Section 2. We introduce our methodology in Section 3 and provide a detailed algorithm in Section 4. In Section 5 we evaluate our approach against similar existing techniques. In Section 6 we conclude and discuss future work.

2. RELATED WORK

Several works propose methods for browsing and navigating Linked Datasets. NautiLOD [4] was our main source of inspiration, providing a language for navigating the Web of Data in a controlled manner. Other approaches, such as Link Traversal Querying [7] and Linked Data Fragments [8], allow for querying Linked Datasets by resolving and traversing URIs automatically. Unlike these approaches, we do not require an upfront query or plan.

Instead our approach enables users to perform their discoveries "on the go". In this sense it is somewhat similar to Linked Data browsers such as Tabulator [1]. However, Tabulator is more akin to a traditional browsing scenario on the Web of Documents. Figures 1 and 2 draw a comparison. In both scenarios, the user is involved in every step. But rather than clicking to resolve a link manually, our approach enables users to search for *multiple* links using keywords: in our approach, the navigation can *branch*.

Versus keyword search approaches (e.g., see [5, 6]), we do not require a local inverted index. We rather aim to enable live exploration of the Web in search of (up-to-date) answers.

A related technique that combines both navigation and search is Treo [9], which accepts a natural language question that it uses to navigate a Linked Dataset in search for answers. However, Treo again requires an upfront question. Likewise Treo is assisted by third-party corpora, such as Wikipedia, to find specific keyword paths to follow, while our approach (currently) relies entirely on Linked Data.

In terms of streaming results to users on-the-fly, Triple Pattern Fragments [3] was a source of inspiration.

3. KEYWORD-BASED NAVIGATION AND SEARCH

To navigate and search the Web of Data using our approach, users (or agents) initiate their browsing similar to the usual Web of Documents: they provide a list of starting locations (seed URIs).[1] The navigation starts by resolving the seed URIs after which users are invited to introduce a keyword. The keyword is used to search for relevant content against the information retrieved. This process is illustrated in more detail in Figure 3 where we perform a navigation from seed URI `http://harth.org/andreas/foaf#ah` looking for relevant information using the keyword "*known*". URIs found using this keyword are sent to an output stream. In the upper-right corner we see this entire process happening again, using a new keyword and new seed URIs, namely the ones returned from the earlier step.

In the following sub-sections, we will explain in more detail the navigation and search aspects of our approach.

3.1 Navigation

The type of navigation we use to browse the Linked Data Web in a programmatic way is similar to other works such as LDpath[2] and NautiLOD [4]. These approaches rely specifically on following RDF links between resources and servers. However, these methods only work in scenarios where users know upfront which RDF links they wish to follow whereas our approach does not make this assumption. Instead, our navigation is interactive and driven by keywords rather than explicit URIs or regular expressions, allowing for a more flexible and less rigorous browsing experience. Furthermore, in other related approaches, users typically have to construct queries upfront and wait for the entire process to finish before they can act on the results. Instead we rely on returning results to users in a streaming fashion as quickly as possible, enabling the discovery of data as you go.

[1]One could imagine these URIs coming from a search engine or bookmark list, etc.

[2]`http://marmotta.apache.org/ldpath/`

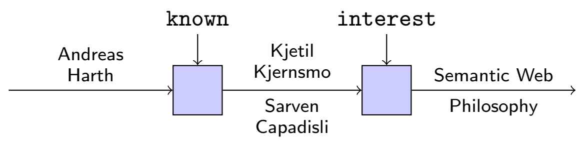
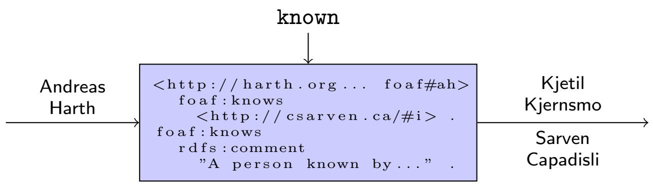


Figure 4: Navigation using a streaming pipeline

To enable the streaming of results, we structure the navigation process in the form of a *pipeline*. Each element of the pipeline is responsible for searching against the data it receives. Figure 4 gives an example. The resource Andreas Harth is fed into the first element of the pipeline, and using the keyword "`known`", we look for relevant triples. These triples are then sent to an output, which becomes the input stream for the next element of the pipeline. Rather than having to wait for the process to finish at each step, we act on results as soon as they are found. Thus the second element can already start processing the resource Kjetil Kjernsmo even though the earlier step has not finished looking for triples. This enables users to continue the navigation using new keywords, even while other elements of the pipeline are still busy looking for relevant results.

3.2 Search

To find relevant triples at each step of the pipeline, several methodologies can be used. For instance, work such as [5] show that keyword search against RDF structures is achievable with high accuracy. We keep the search component modular, so that specific implementations can decide which technique to utilize. In the evaluation section herein we perform search using a string similarity algorithm against the string representation of each triple.



```
<http://harth.org...  foaf#ah>
  foaf:knows
    <http://csarven.ca/#i> .
foaf:knows
  rdfs:comment
    "A person known by..." .
```

Figure 5: Search using predicate resolution

To enable further matches, we extend the search with content acquired by resolving the predicate URIs associated with the current resource. When matching RDF links, we are thus not only limited to matching tokens in the URI string itself, but can match labels, comments, etc. Figure 5 shows an example of how predicate resolution can enrich a search result. For the Andreas Harth resource, no triples are found using the keyword "`known`". However, by resolving predicate URIs we are able to find triples within the foaf:knows predicate that match the keyword. We can therefore use triples with the foaf:knows predicate that are associated with our main resource to continue our navigation.

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

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

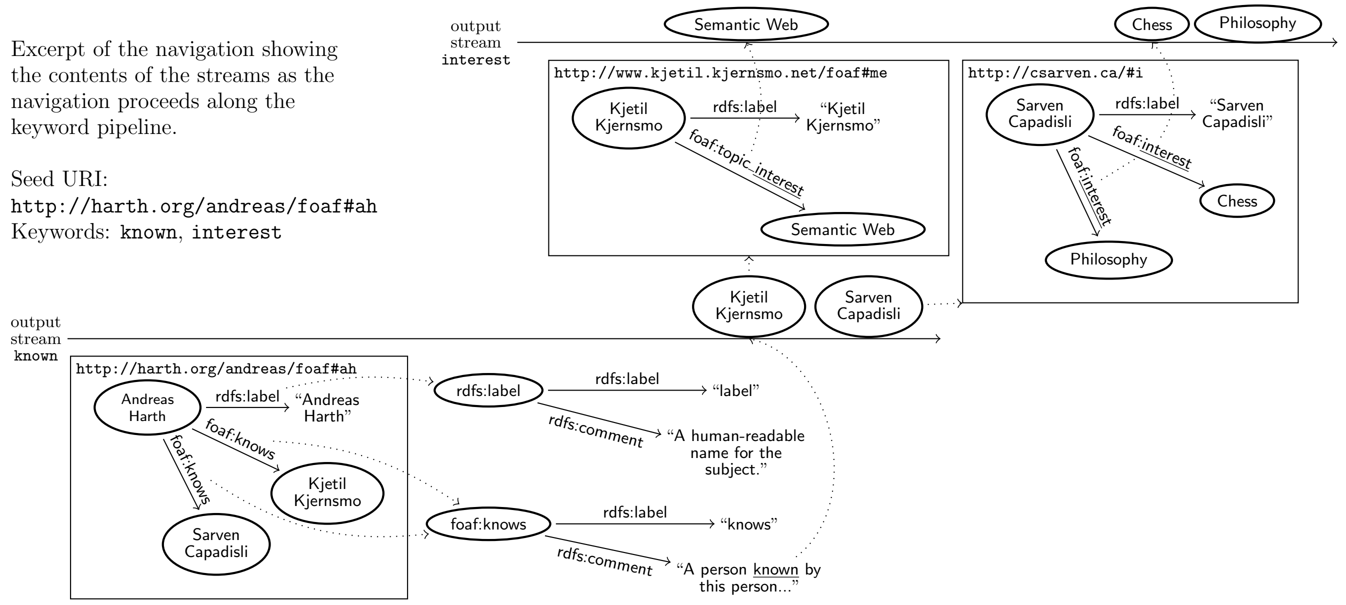


Figure 3: What are the interests of the people known by Andreas Harth?

Many predicate URIs will resolve to an entire ontology or vocabulary and not just to the term identified. For instance, the URI `http://xmlns.com/foaf/0.1/knows` resolves to a description of the entire FOAF vocabulary. For this reason our approach relies only on the triples where the resolved URI appears in either the subject or object position.

4. ALGORITHM

We will now describe our approach in more detail by presenting an algorithm and explaining its functionality using a walkthrough example. The question in Figure 3 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 which the navigation initiates, and an ordered set of keywords we need to follow to answer our question (e.g., in the order entered interactively by the user). Each keyword is effectively an element in the pipeline. Algorithm 1 gives an overview of the procedure for a single element of the pipeline.

The algorithm takes as input a stream which we call S, and a keyword k. By following Figure 3, 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 on 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 yet.

Next on line 8 we obtain triples that were not yet found and assign them to the set T. We proceed by resolving the predicate URIs of this set. We loop through each triple of this set on line 9, and we resolve each predicate on line 10. We then apply our search methodology against the set P using the keyword "`known`" with found triples assigned to the set F' (line 12); if F' is non-empty, we extract new

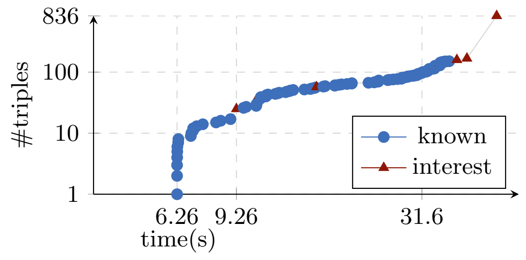
Algorithm 1 Keyword-based navigation and search

```
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:     $N \leftarrow \{$ unvisited URIs of $F$ except predicates $\}$
 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:         $N \leftarrow \{$ unvisited URIs of $t$ except predicates $\}$
14:         $out$.write($N$)
15:     end if
16: end for
```

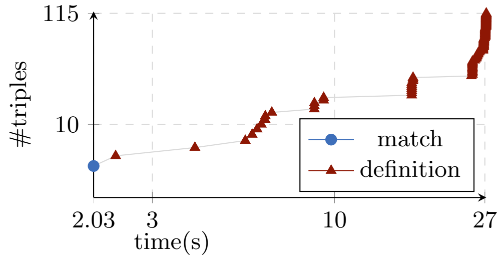
URIs from the current triple t (line 13). On line 14 we write the extracted URIs to the stream *out*. Thanks to predicate resolution, we now match triples with predicate foaf:knows, resulting in 2 output URIs, namely `http://csarven.ca/#i` and `http://www.kjetil.kjernsmo.net/foaf#me`.

Figure 3 continues by listening for the keyword "`interest`". The algorithm is re-executed with an input stream fed from the output stream of the earlier execution. We therefore obtain 2 URIs in the input stream; for each of these URIs the process is repeated. The process terminates when all processing has finished for the given keywords or the user manually terminates the session.

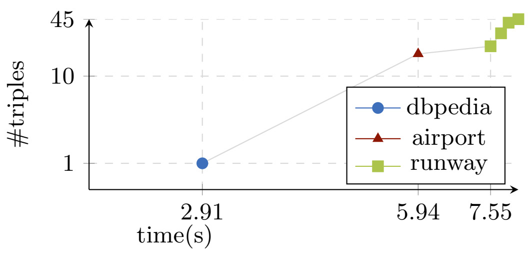
It is important to note that the resolution of URIs can occur in parallel. In the evaluation section we specifically create a new thread for each request to obtain higher throughput. URIs resolved should also be cached to avoid unnecessary re-retrieval of commonly requested documents.



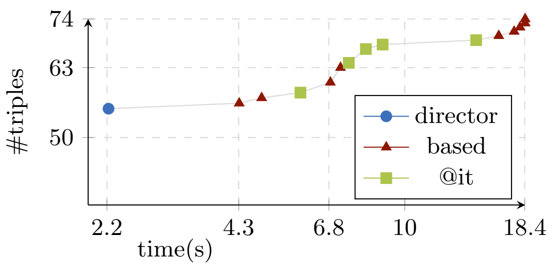
(a) What are the interests of the people known by Andreas Harth? 2 keywords and 1 seed URI: `http://harth.org/andreas/foaf#ah`



(b) What are the available definitions for the English noun "apple"? 2 keywords and 1 seed URI: `http://babelnet.org/rdf/s00005054n`



(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: Execution of questions

5. 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 preliminary evaluation is to check whether our identified method produces relevant results in an acceptable time frame. To this end, we performed two different experiments:

1. First we executed a series of questions, each represented as a set of keywords with a seed URI, using our identified approach. We measured how fast and how many relevant triples appeared in the output streams.

2. Next we compared our methodology against a similar implementation called SWGET [10], a multi-threaded tool that executes NAUTILOD expressions (SWGETM), and we measured the response times at each navigation step for both approaches.

5.1 Experimental setup

To run the experiments, we implemented our navigation pipeline component using an observer pattern and our search component as a multi-threaded Java process that uses the open source APACHE ANY23[3] library to resolve URIs and obtain triples. We instantiated keyword search by utilizing the *dice coefficient*[4] string matching algorithm which works especially well with strings of variable length. The SWGET comparison does not take into account our search methodology since we use explicit 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 (SWGET was also configured with this size). All results for all the experiments are the average of 5 runs.

5.2 Results

Figures 6a, 6b, 6c and 6d show the contents of the various output streams at specific times. Each plotted mark represents data being written to the stream.

We will analyze the results in terms of *response time* (reception of first solution for every keyword) and *navigation hop time* (reception of first solution between each keyword). For the question in Figure 6a, the response time is around 6 seconds for the first keyword and 9 seconds for the second keyword. This is mainly due to the fact that we had to resolve the predicate URIs in order to find triples matching the first "*known*" keyword. The average navigation hop time is 7.7 seconds. For the question in Figure 6b, the response time is 2 seconds for the first keyword and 2.3 seconds for the second. The navigation hop time is 2.2 seconds. For this question we see that the results are returned quicker because we did not have to resort to predicate resolution. For the questions in Figure 6c and 6d, triples relating to the first keyword are found in 2.8 and 2.2 seconds respectively, 5.9 and 4.3 seconds for the second keyword, and 7.5 and 5.8 seconds for the third keyword. The navigation hop time is 5.4 and 4.1 seconds. For these two questions we see that, even with a greater number of keywords, the navigation hop time remains below 10 seconds.

In summary, in these results we can see that response times are all under 10 seconds. Even more important is the navigation hop time, which suggests that users can navigate across resources, and match relevant triples, in around 5 seconds on average. These results, although preliminary, show us that browsing the Web of Data using keywords is possible within the bounds of interactive time-frames, albeit a bit

[3]`http://any23.apache.org/`
[4]`http://www.catalysoft.com/articles/StrikeAMatch.html`

sluggish by modern browsing terms. Methods for improving response times are discussed in the future work section.

SWGET *comparsion:* The speed difference between SWGET and our approach (indicated as KBLD) is shown in Figure 7 where we see SWGET's response time increasing especially for results returned by following `owl:sameAs` links. With our approach the response times are stable without any relevant jumps. The quicker response times of our approach are mainly due to our pipeline algorithm which, instead of waiting for each step to finish, sends results along the pipeline as soon as they are found; our goal is specifically to enable interactivity. SWGET was configured to stream results, however, it still performs a breadth-first search which means each step has to finish before the navigation can continue.

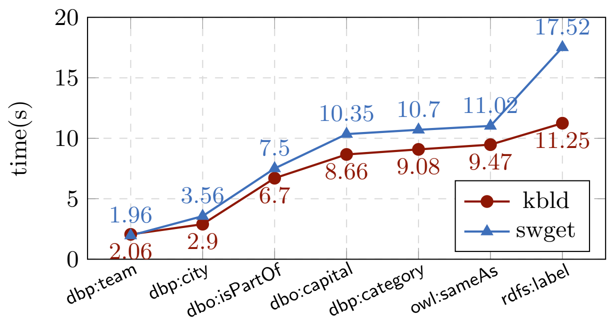


Figure 7: Response times comparison. Seed URI: `http://dbpedia.org/resource/Michael_Jordan`

6. CONCLUSION AND FUTURE WORK

In this short paper we combined RDF keyword search methodologies with Linked Data traversal techniques. Preliminary results showed that combining these two methodologies – with pipeline-based navigation and predicate resolution search – enables clients to explore the Web of Data using keywords. 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.

For future work we would like to further develop and evaluate the techniques presented. Reasoning methods could be used to induce and find other relevant information. BABELNET [11] may be useful to match synonyms and translations. Disambiguation methods such as BABELFY [12], using the context acquired during navigation, may increase the accuracy of the search. For traversal and navigation techniques, presenting a summary of the most used keywords available at each resource could assist users with navigation. Several issues arise when URIs are unavailable or when limits are imposed by data providers; better approaches are also needed for concurrent resolution of URIs so as to provide quick response times while not straining servers. Future work could therefore provide better methods for effectively crawling Linked Datasets at runtime.

We would also like to look at the potential of using our keyword-based approach inside an application such as Tabulator [1]. Furthermore we would like to implement a standalone application to showcase the possibilities of this approach, namely a browser that navigates Linked Data resources using keywords rather than clicking links.

Acknowledgments

Sapienza affiliated authors gratefully acknowledge support from the LIDER project (№ 610782), a Coordination and Support Action funded by the European Commission under FP7. This work was also supported by the Millennium Nucleus Center for Semantic Web Research under Grant № NC120004, and Fondecyt under Grant № 11140900.

7. 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 *ISWC'09*, 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 *ISWC'14*, pp. 180–196, 2014.

[4] V. Fionda, C. Gutierrez, and G. Pirró, "Semantic Navigation on the Web of Data: Specification of Routes, Web Fragments and Actions," in *WWW'12*, pp. 281–290, 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," *Journal of Web Semantics*, vol. 9, no. 4, pp. 365–401, 2011.

[7] O. Hartig and J.-C. Freytag, "Foundations of traversal based query execution over Linked Data," in *HYPERTEXT*, pp. 43–52, ACM, 2012.

[8] R. Verborgh, M. Vander Sande, P. Colpaert, S. Coppens, E. Mannens, and R. Van de Walle, "Web-Scale Querying through Linked Data Fragments," in *Proceedings of the 7th Workshop on Linked Data on the Web*, 2014.

[9] A. Freitas, J. G. Oliveira, E. Curry, S. O'Riain, and J. C. P. da Silva, "Treo: Combining Entity-Search, Spreading Activation and Semantic Relatedness for Querying Linked Data," in *Proc. of 1st Workshop on Question Answering over Linked Data (QALD-1) at the 8th Extended Semantic Web Conference (ESWC 2011)*, 2011.

[10] V. Fionda, C. Gutierrez, and G. Pirro, "The swget portal: Navigating and acting on the web of linked data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 26, pp. 29–35, 2014.

[11] R. Navigli and S. P. Ponzetto, "BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network," *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.

[12] A. Moro, A. Raganato, and R. Navigli, "Entity Linking meets Word Sense Disambiguation: a Unified Approach," *TACL*, vol. 2, pp. 231–244, 2014.