# Linked Data Person Search Engine

Aristotelis Charalampous
School of Computing and Communications
Lancaster University
United Kingdom
a.charalampous@lancaster.ac.uk

## ABSTRACT

Discovery of knowledge by accessing the Semantic Web necessitates the efficient utilization of crawling and indexing strategies. The searching depth and relevance of seed URIs can define the spectrum of results output. This paper focuses on exploration of said methodologies through a custom developed application to suggest possible areas of improvement and best practices in an experimentally guided process. Achieving so has been the contribution of SPARQL and Apache Jena technologies for RDF relationship extractions within the Wikipedia empowered DBpedia corpus [1]. Primary ambition is to basically document discrepancies in live and focused crawling designs in an overabundance of analogous research projects.

## Categories and Subject Descriptors

H.3.4 [Information Systems]: Information Storage and Retrieval—Semantic Web; H.3.3 [Information Search and Retrieval]: Search process.

## General Terms

Algorithms, Documentation, Performance, Design

## Keywords

Linked Open Data, Semantic Web, Search Engine, Crawler, Resource Retrieval, Indexing

## 1. INTRODUCTION

Undoubtedly, the widely successfully Linked Data standards and applications have attracted their increasingly high use by data providers, leading to rich content additions to industrial database silos [2]. The key to their popularity mainly lies in the multi-dimensional links between resources in a single, global information gamut consisting of around 11 million unique URIs, as estimated by the SWSE search engine [3]. These resources encompass documents and data, an evolution over the past state of the Web [4]. The notion of Semantic Web exploitation is increasingly becoming adopted by a numerous selection of papers. Studies such as in [5][1], for example, move towards "smartly" directing queries by filtering the unified space in a topic-based manner. Such a suggestion has also been earlier work done in [6] by employing ontological knowledge reinforced by weights attributed to each of its terms. This is a clear counterpart to breadth-first alternatives which promote quantitative implementations.

The de-facto consideration over natural language questions is their conversion into structured queries. Consequently, mappings based merely on keywords and their lexical derivatives are a current trend. A popular solution has already been presented in [7], where each search term serves as an object reference that is subsequently ranked to judge its similarity with the general context of the query and pre-structured class hierarchies. An extension to the above is found in [8], embedding query enrichment with semantically synonymous terms. Given a broader understanding, it can be appreciated that related solutions contribute to accurate evaluation and result delivery. Finally, lexicographic similarity can be expanded to accommodate for not only classical relations such as synonymy and hypernymy, but also antonymy and various other semantic associations [9].

Another challenge lies in the chosen indexing method which can take different forms. The aforementioned SWSE search engine [3] is a loosely-based over the inverse-functional property[2]. In fact, their tailored-made rule-based variant continuously attempts to identify a set of properties that can uniquely identify resources based on previously recorded knowledge. Still not largely abandoned, a reliable and straightforward alternative concerns inverted index structure [10]. Index keys, in one of its variants, are URIs in which appear in document extracted data. Feasible applications demonstrate the power of converting inferred knowledge into such a single structure to support higher scalability and keyword based querying [11]. In fact, this enables the parallel use of vector space models to empower relevancy rankings.

Therefore, it is our intention in this study to incorporate and record qualitative metrics that will help us discern fundamental differences on a direct comparison of mainly breadth-first and topic specific applications that will be presented in Section Four and Five.

The rest of the paper is organised as follows: Section Two will give the details behind current commercial and academic projects that provided inspiration, as well as provide grounds for system design decisions we take afterwards. Section Three outlines the system's logical procedures that reach query fulfilment and

[1] The paper is available here due to it probably being drawn from publication.

[2] Inverse Functional Property (IFP) index can indicate if a property is unique to a given resource, i.e. the serial number of the US dollar.

Section Five a conclusion of the paper with some remarks for future work.

## 2. RELATED WORK

Expanding on previous content included for [5], we are especially intrigued by their choice of fitting searching requirements directly to their system's graphical interface. This has taken the form of a hotel booking information portal that could extremely minimise cascading branch outs to related URIs. In other words, a focused crawler is developed, crawling through domain-specific pages. It is paramount to distinguish the two types here. Focused types, in general, are pre-runtime processes which, during their execution, populate an index file or a locally accessible database. Live (unfocused) explorations, however, are link-traversal based data retrieval methods that do not have any a-priori information about the Linked Open Data (LOD) Cloud[3].

Apart from applying linear computation of a given keyword set to query from, [3] makes extensive discussion in how the optimal search engine would revolve around distribution of crawling tasks. Given that large-scale analysis of the Semantic Web is a pre-requisite, a master machine in a computer network cluster can arrange for parallel document processing tasks. This has also motivated many early studies, such as Swoogle [12], to replicate Google's HTML document web crawling (PageRank) as a full-fledged system supported by relational databases. Another distinguishable example is UbiCrawler [13], a fully-fledged, de-centralised system that constructs independent crawling agents. Each one scans its own part of the web, ideally increasing throughput (information from pages) as these agents grow in numbers.

Recent advancements have seen use of buckets of triple stores, that vaguely resemble hashing functions, to build overlays based on queries' properties [14]. This layering strategy is especially effective in introducing equivalency classes that impose query redirections to relatively relevant content. Similarly to this, [15] is composed by triple stores that follow specific patterns based on combinations of URIs that are given as an input. A single matching triple can ultimately be found that contains index entries that can further allow data source rankings.

Revisiting live exploration techniques at this point, it can be said that their nature does not necessarily accommodate for indexing strategies. Reuse of previously mined knowledge though can prove invaluable as performance times and query quality results accumulated from data sources would not have been otherwise discovered [16]. Crawling into unknown domains can have its risks though, potentially leading to policy breaches that data providers may have set. To tackle this, an upper limit of URIs crawled per pay-level-domains can maximise time-periods in between subsequent resource requests [17].

Tim Berners-Lee has clearly defined the following expectations from publicly published data in the Web[4]:

1. URIs should be used to identify entities
2. HTTP URIs should be used for discovery of said, publicly available entities.
3. Useful information should be provided when a URI is accessed.
4. Links to other URIs should be included, for enhanced entity discovery.

Defeating the purpose of the 3rd and 4th point though, blank nodes in the LOD Cloud account for a small but not negligible ~6%, existence and confrontation of which has been discussed but a final solution not found [18]. Simply put, they play a detrimental role in the detachment of resources from the LOD Cloud due to insufficient linkage, causing searching strategies to produce incomplete results.

Eventually, there is significant work that must be conducted to safeguard a linked data crawler from the unexpected behaviour that either user inputs or the LOD Cloud itself may exhibit. A mere minority of best practices was presented here that we will appeal to in Section Three.

Semantic Web Search Engines can be classified based on the extent to which they fulfil the following requirements, as [19]:

1. Search Environment
2. Query Types
3. Iterative and Exploratory Search
4. Intrinsic problems

The underlying requirements to fulfil each lie in the depth of concept and parallelisation of tasks, leading to creation of different ontologies from related sources, dynamic term analysis for cognitive interpretation of the desired query results and, finally, evolution of the search engine to infer from past knowledge possible crawling routes for optimisation of query results and autocomplete suggestions for the first three respectively. The last category describes ranking and conciseness of retrieved results and entities. All-encompassing examples are Lexxe[5] and Hakia[6], demonstrating powerful searching capabilities.

A last point we wanted to touch is based on Natural Language Processing (NLP) techniques to institute queries from textual input from users. Early advancements made towards tackling the challenge of adaptation of queries to any domain and user input in [20] have shown resulting accuracy of results validity of over 70%. Brought together with ontology online searching, frameworks such as "FREyA" [21] have seen improved recall and precision, surpassing ambiguity and enriching input text with vocabulary enhancement techniques. This has been paramount in the field as previous implementations required the user to inspect query construction if input was unambiguous enough.

[3] A massive set of RDF triples given in the form of {subject, object, and predicate}.

[4] http://www.w3.org/DesignIssues/LinkedData.html [Last accessed on 14/05/2015]

[5] Supports NLP for search in keywords. Query auto completion and rich snippet creation are also featured. Accessible here.

[6] Hakia is available only for corporations and enterprises currently. It's extremely potent SemanticRank algorithm combines fuzzy logic and computational linguistics.

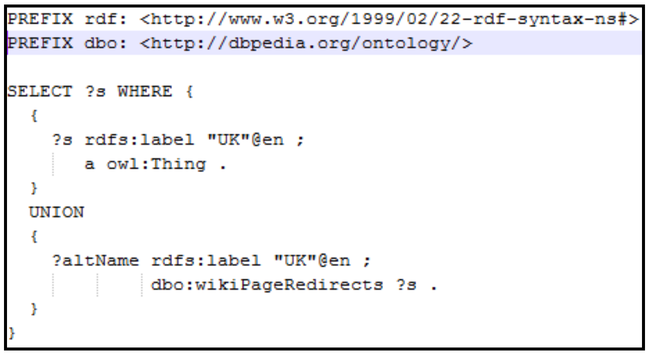
## 3. METHODOLOGY

### 3.1 Crawling Strategies

To try and minimise our problem set, at first, we have decided to find a category set that would give us a rich set of persons' resource URIs. After some investigation in the DBpedia site, it was clear that countries' URIs contained a plethora of people. This has been added as an option for the user in the GUI. We still allow the crawler to branch out from the first country seed URI but this scarcely happened.

An extra layer was added to support a more generic approach. Based on the keywords that a user gave to the application, seed URIs were dynamically constructed and used for Semantic Web crawling. In other words, it is the opposite of what can happen with a country as the keywords provided may be unambiguous or completely unrelated to each other. Of course the URIs may not be assigned to any entity, or, they might be redirected to another resource (see Figure 1). Both of these are handled appropriately by SPARQL queries that reveal any of these occurrences.

Re-crawling schedulers were considered for focused crawling but not for live. Since live crawling's sole purpose is to incrementally index the semantic web, it would be impossible to achieve this by arranging whole index re-structures. This is not the case for country-specific searching though as we intend to re-structure those after a given period of time (given by either the.xml file or for the default value of one day). To re-structure those would not be catastrophic as the search is shallow and limited to a single or extremely few branching out calls. Arguably, the search engine is solemnly targeting "*foaf:Person"* type of resources, and information about individuals to which they correspond to may not change for large time periods.



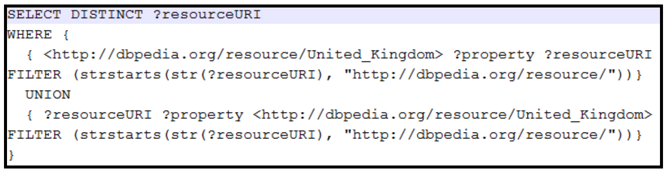
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?s WHERE {
  {
    ?s rdfs:label "UK"@en ;
       a owl:Thing .
  }
  UNION
  {
    ?altName rdfs:label "UK"@en ;
             dbo:wikiPageRedirects ?s .
  }
}
```

**Figure 1:** Redirection Query using "UK" as the input keyword

Conclusively, live crawlers are repeatedly re-crawling without any constraints but, to replicate the instant responses for filtered searches (when re-crawling does not take place), a non-polling approach was developed. This has allowed division of reporting tasks to two paths. Firstly, reports back to the user with available knowledge from the index file are immediate. Secondly allow the user to browse through those results while the live crawling progresses and report back with the complete results afterwards. During this time, the graphical interface is disabled until the process finalises appending extra results to the pre-computed, if any.

The crawling algorithm is the backbone of the whole strategy, supporting recursive calls to the visitation of URIs until either the maximum branching times allowed have been reached or if the inspected URI reaches a point where we cannot branch out anymore. This is almost impossible, as the algorithm is tuned to retrieve all people's URIs before branching out again. This way, we maximise the volume of relevant people in the first few branches. Despite this and given a considerable amount of people to be found, this will backfire in terms of relevancy of results.

Last, but not least, the decision to what resource the crawler should branch out is left at chance. We first execute a query that will return all valid resource with which the current person's resource is linked to it (Figure 2). Then, based on an iteration of the results, we throw a dice to determine if we are going to branch out on a currently visited resource in the iteration.



```
SELECT DISTINCT ?resourceURI
WHERE {
  { <http://dbpedia.org/resource/United_Kingdom> ?property ?resourceURI
FILTER (strstarts(str(?resourceURI), "http://dbpedia.org/resource/"))}
  UNION
  { ?resourceURI ?property <http://dbpedia.org/resource/United_Kingdom>
FILTER (strstarts(str(?resourceURI), "http://dbpedia.org/resource/"))}
}
```

**Figure 2:** SPARQL query to list of possible resource URIs to branch out

### 3.2 Graphical User Interface (GUI)

We have designed the application's GUI to be as simplistic as possible. First of all, it is based on Java Swing's library with a total of four different screens (two replica report screens) and a few more pop-ups that will be triggered on either wrong user input or for confirmation of an action. Some of the application's variables can also be changed through the available .xml file, namely the maximum allowed number of branches and day limit since next re-crawling update (not for live crawling). The limit is update to milliseconds internally for direct comparisons with *System.nanoTime()* calls (as supported by Java 5+).

### 3.3 Document Inverted - Indexing

A set of indexes constitutes the corpus of knowledge we are trying to build using this application. This set includes a single index for each country (for which filtered crawling was used). A single index file representing live crawling is held at any point of time. To better optimise concatenation functions, as well as insertion and value retrieval from our data structures, we have been using artificially made indexes for each term that might appear in DBpedia resources. A mapping from their respective URIs is held in another data structure and is accessed only for file output purposes.

Indexing in our context takes the form of text searching and processing solely. We retrieve textual content from "*dbpedia-owl:abstract"*, "*rdfs:comment"* and "*dc:description"* properties, which are present in almost any person resource URI we have visited (so far). We then move to remove any stop words that appear in either of their respective text content as well as the search query, using pre-developed lab code. Every term is stored in the index in lower case form to avoid disambiguation when looking up for keys in our data structures. Noteworthy is also that our search engine will not attempt to define search results based on keywords for which no reference was found in the text input. This is to avoid returning no results when it occurs.

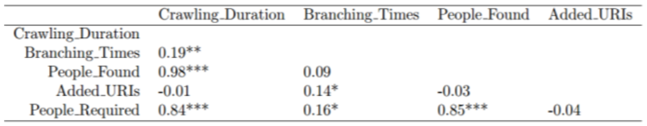
## 4. EVALUATION

During the plethora of runtime executions of our application, we have assembled more than 200 entries with 7 attributes gathered from a 3-day sampling period. We have added 2 extra variables of derived features that we need to satisfy our tests. A description of each can be seen in Table 1. Each dataset entry corresponds to a Semantic Web crawling procedure using a specific keyword from the search query.

| ***Feature Name*** | ***Description*** | ***Type*** |
|---|---|---|
| Date-Time | The date and time of the current dataset entry was recorded. | Date-Time |
| Crawling Duration | The time required to output results after the crawling procedure (it might not take place in focused crawling). | Numeric |
| Branching Times | How many time we have branched out from resource URIs in the duration of the procedure. | Numeric |
| People Found | The number of people found. | Numeric |
| Keyword | The keyword used to derive seed URI to search initially for people and later branch out. | Factor |
| Added URIs | The number of extra URIs added. The results here depend heavily on previous knowledge and if the search query has been previously repeated, as well as the number of people that similar queries were requested to find. | Numeric |
| People Required | The people required from the crawler to be found. | Numeric |
| From Country | Did this query originate from a country focused crawl or a live one? | Factor |
| Branching Type | We have produced 5 categories that define act as levels to define shallow (1) and in-depth crawling (5). | Factor |

**Table 1:** Dataset attributes descriptions

The first step in our dataset exploration takes place with the correlations in for all our numerical metrics. As seen in Figure 3, there are three pairs which demonstrate a high positive relationship. Of course it is sensible to say, even without much evidence, that a search engine will yield more results based on the time it is allowed to crawl the semantic web but it is interesting that our search engine displays good behaviour in retrieving almost every time, as much people as are required. It is only natural that, given more data, the number of people found in comparison with added resources to our index files shows a much higher negative correlation metric. Time constraints are, therefore, reflected in this study from this as the main endeavour was building a concise and seamlessly operated search engine.



| | Crawling_Duration | Branching_Times | People_Found | Added_URIs |
|---|---|---|---|---|
| Crawling_Duration | | | | |
| Branching_Times | 0.19** | | | |
| People_Found | 0.98*** | 0.09 | | |
| Added_URIs | -0.01 | 0.14* | -0.03 | |
| People_Required | 0.84*** | 0.16* | 0.85*** | -0.04 |

**Figure 3:** Correlation values for dataset metrics

Notable is also that our live crawling index file has reached high dimensionality for a variety of keywords. This is provided in the working documents of this study. To investigate though how our factors define the various data distributions, we visualised how the number of people found against the crawling duration when categorised by the branching level and the query type (Figure 4).

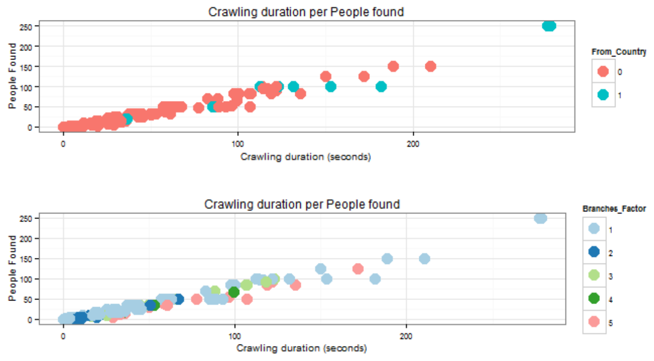


**Figure 4:** Crawling Duration per People Found by Query Type (above) and Branching Level (below)

We can observe that for queries that originate from country specific queries are more concentrated to the axis. This suggests that the crawling duration is not high in most occasions but the content is not as reach as generic crawling. Country resource URIs are interlinked with a lot of persons URIs and their selection was done purposefully here to provide a proof of concept of the power of live crawling. This claim is biased though because the dataset contains, in its majority, live crawling entries. Additionally, the dataset is split evenly between entries for queries below and above 25 people to be found. When it comes to branching levels, it seems quite surprising that highly branching out crawlers do not necessarily produce the best results in crawling times and the content's quality. We can safely say that, with the right selection of seed URIs, need for multiple branch outs is redundant. This was, after all, contribution of current research by optimising search queries through user input and past knowledge analysis.

It is paramount here to visit the search engines accuracy-centric metrics to finalise our qualitative exploration. Figure 6 is intuitive in terms that the greater population of keyword based-searching does not exceed the limit of 100 seconds in their median value for any branching level.

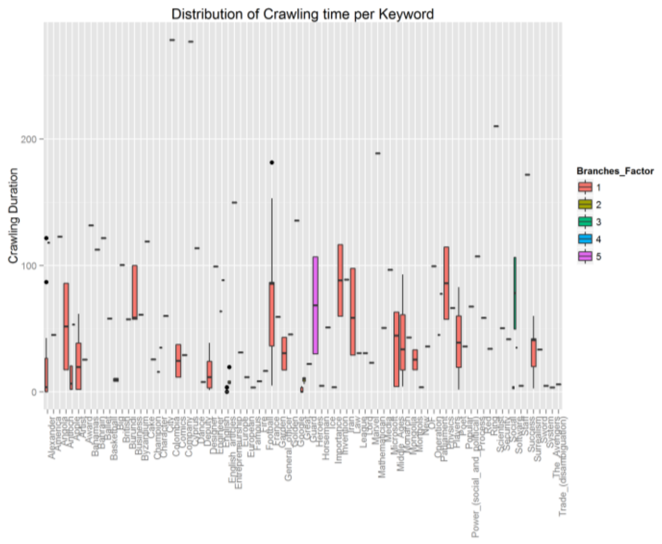


**Figure 5:** Distribution of crawling time required per term in given query

Nevertheless, there is a change in pattern for how many people are found for each. This is hard to quantify at this point because there is a clear disadvantage in lack of understanding on exactly how DBpedia is structured and how that impacts such kind of experiments. Figure 7 is capturing this, listing a total of 83 keywords, same as the previous graph.

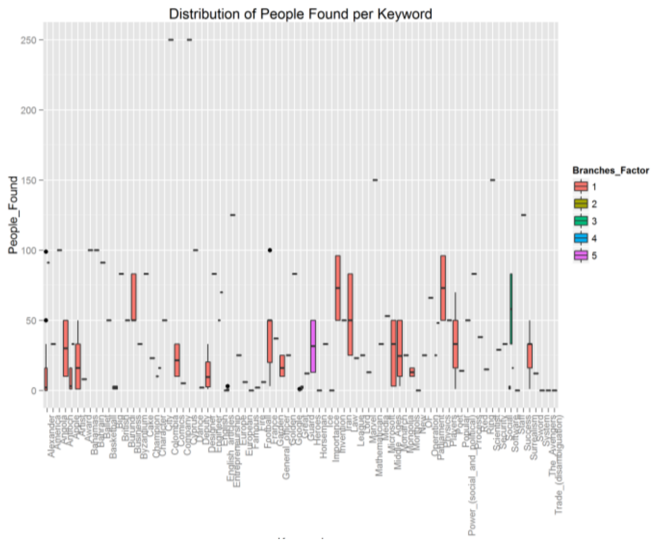


**Figure 6:** Distribution of people found per term in given query

Further discussion will follow in Section Five will uncover pragmatic discoveries that will lead to recommendations for research considerations in the state of the art.

## 5. FINDINGS

Live and focused crawling strategies have been the struggle of contemporary research studies. This paper has had from the beginning a scope of identifying reasons for developers and scientists alike to sway with one or other (or even a hybrid variant). Experience built in this area has led to the conclusions summarised in Table 2. These conclusions can be very likely complement exhaustive comparison of crawling methodologies, such as in the work of G. Pant et al [22].

| *Live Crawling* | *Focused Crawling* |
|---|---|
| ▪ Poses serious time constraints.<br>▪ Rich in content delivery.<br>▪ Highly dependable in various cues from crawled pages.<br>▪ Requires accurate user input and dynamic inference of entry points. | ▪ Not very scalable and limited to an assigned domain.<br>▪ Requires collaboration of similar crawlers to help in preventing querying in irrelevant domains.<br>▪ Good entry points resulting highly accurate first results. |

**Table 2:** Summary of advantages and disadvantages based on crawling types' characteristics

It would not prove feasible if this study has expanded in different domains other than DBpedia and, because of that, we sought to exploit the availability of correctly labelled and linked data points. Judging from success in intelligent tracking of content linkage at runtime without additional manual setup [23], there is little we can say about the complexity of our product in that regard. Indeed, our live crawling discipline could be very well overshadowed by Naïve Best-First Crawlers. These were described in various contexts, such as the thesis in [24]. Briefly, they target cosine similarity computations on given Web pages in the first iteration, letting subsequent ones know which to crawl.

Best described as opportunistic, it cannot be ignored that our random crawling approach defies the requirement of lengthy crawling requirements. Despite the acceptable responses in smaller queries, because we rush into filling our data structures with people from resources on the early crawling stages (that yield more relevant content than later stages), this should generally be avoided. The recursive method adopted is also unsafe for extended crawling as there is risk for stack overflow errors, after further testing.

## 6. CONCLUSION

Certainly, this study has been an enlightening experience in how affluent the semantic web crawlers can be in contrast with conventional ones. Tapping into a structured world-wide web is truly an inspiring direction to the evolution of the Internet. It is also understood that such a solution is way more lightweight as parsing and transforming whole HTML documents are resource intensive and pose a dangerous environmental threat in the form of large data centres and, therefore, carbon emissions.

Future plans in the area and applications such as this one are largely within the radius of machine learning and genetic neural networks algorithms. Systems can be trained to engineer features on documents as they pass by, increasingly evolving their prediction of “positively” and “negatively” related to the class of interest Web pages, as is suggested by [25].

Given the above, there is an encouraging tendency towards reflected in the vivid activities of the research community and potential for maturation of higher-end semantic searching services.

## 7. REFERENCES

[1] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, “DBpedia - A crystallization point for the Web of Data,” *J. Web Semant.*, vol. 7, no. 3, pp. 154–165, 2009.

[2] C. Bizer, T. Heath, and T. Berners-Lee, “Linked data-the story so far,” *Int. J. Semant. Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[3] 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,” *J. Web Semant.*, vol. 9, no. 4, pp. 365–401, 2011.

[4] C. Bizer, “The emerging web of linked data,” *IEEE Intell. Syst.*, vol. 24, no. 5, pp. 87–92, 2009.

[5] M. Bansal, “ONTOLOGY BASED SEARCH USING SEMANTIC WEB,” vol. 3, no. 3, pp. 8–13, 2014.

[6] D. Mukhopadhyay, a. Biswas, and S. Sinha, “A New Approach to Design Domain Specific Ontology Based Web Crawler,” *10th Int. Conf. Inf. Technol. (ICIT 2007)*, pp. 289–291, 2007.

[7] G. Cheng and Y. Qu, “Searching Linked Objects with Falcons,” *Int. J. Semant. Web Inf. Syst.*, vol. 5, no. 3, pp. 49–70, 2009.

[8] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, “Sindice.com: a document-oriented lookup index for open linked data,” *Int. J. Metadata, Semant. Ontol.*, vol. 3, no. 1, p. 37, 2008.

[9] A. Budanitsky and G. Hirst, “Evaluating WordNet-based Measures of Lexical Semantic Relatedness,” *Comput. Linguist.*, vol. 32, no. 1, pp. 13–47, 2006.

[10] J. Umbrich, K. Hose, M. Karnstedt, A. Harth, and A. Polleres, “Comparing data summaries for processing live queries over Linked Data,” *World Wide Web*, vol. 14, no. 5, pp. 495–544, 2011.

[11] A. Bialecki, R. Muri, and G. Ingersoll, “Apache Lucene 4,” *Proc. SIGIR 2012 Work. Open Source Inf. Retr.*, pp. 17–24, 2012.

[12] L. Ding, T. Finin, a Joshi, and R. Pan, “Swoogle: a search and metadata engine for the semantic web,” *Cikm*, pp. 652–659, 2004.

[13] P. Boldi, B. Codenotti, M. Santini, and S. Vigna, “UbiCrawler: A scalable fully distributed Web crawler,” *Softw. - Pract. Exp.*, vol. 34, no. 8, pp. 711–726, 2004.

[14] M. Konrath, T. Gottron, and A. Scherp, “SchemEX—Web-Scale Indexed Schema Extraction of Linked Open Data,” *Proc. 11th Interational Semant. Web Conf. ISWC2011*, pp. 1–8, 2011.

[15] G. Ladwig and T. Tran, “Linked Data Query Processing Strategies,” *Semant. Web ISWC 2010*, vol. 6496, pp. 453–469, 2010.

[16] O. Hartig, “How caching improves efficiency and result completeness for querying linked data,” *CEUR Workshop Proc.*, vol. 813, 2011.

[17] H. Lee, D. Leonard, X. Wang, and D. Loguinov, “IRLbot : Scaling to 6 Billion Pages and Beyond,” *Computer (Long. Beach. Calif).*, vol. 3, no. 3, pp. 427–436, 2008.

[18] A. Mallea, M. Arenas, A. Hogan, and A. Polleres, “On blank nodes,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7031 LNCS, no. PART 1, pp. 421–437, 2011.

[19] M. Majid Qureshi, “Comparative Analysis of Semantic Search Engines Based on Requirement Space Pyramid,” *Int. J. Futur. Comput. Commun.*, vol. 2, no. 6, pp. 562–566, 2013.

[20] E. Kaufmann, A. Bernstein, and L. Fischer, “NLP-Reduce: A ‘naïve’ but Domain-independent Natural Language Interface for Querying Ontologies,” *4th Eur. Semant. Web Conf. ESWC 2007*, pp. 1–2, 2007.

[21] D. Damljanovic, M. Agatonovic, and H. Cunningham, “FREyA: An interactive way of querying linked data using natural language,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7117 LNCS, pp. 125–138, 2012.

[22] G. Pant, P. Srinivasan, and F. Menczer, “Crawling the Web,” *Springer*, pp. 153–177, 2004.

[23] L. Barbosa and J. Freire, “An Adaptive Crawler for Locating Hidden-Web Entry Points,” *Www*, pp. 441–450, 2007.

[24] G. Pant, “Learning to crawl Classifier-guided topical crawlers,” no. July, p. 160, 2004.

[25] S. A. Özel, “A Web page classification system based on a genetic algorithm using tagged-terms as features,” *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3407–3415, 2011.