A Discovery and Analysis Engine for Semantic Web*

Semih Yumusak†
KTO Karatay University, Turkey
semih.yumusak@karatay.edu.tr
AI4BD GmbH, Switzerland
s.yumusak@ai4bd.com

Andreas Kamilaris
Department of Computer Science,
University of Twente The Netherlands
a.kamilaris@utwente.nl

Erdogan Dogdu
Cankaya University
Ankara, Turkey
edogdu@cankaya.edu.tr

Halife Kodaz
Selcuk University
Konya, Turkey
hkodaz@selcuk.edu.tr

Elif Uysal
KTO Karatay University
Konya, Turkey
elif.uysal@ogrenci.karatay.edu.tr

Riza Emre Aras
KTO Karatay University
Konya, Turkey
riza.emre.aras@ogrenci.karatay.edu.tr

ABSTRACT

The Semantic Web promotes common data formats and exchange protocols on the web towards better interoperability among systems and machines. Although Semantic Web technologies are being used to semantically annotate data and resources for easier reuse, the ad hoc discovery of these data sources remains an open issue. Popular Semantic Web endpoint repositories such as SPARQLES, Linking Open Data Project (LOD Cloud), and LODStats do not include recently published datasets and are not updated frequently by the publishers. Hence, there is a need for a web-based dynamic search engine that discovers these endpoints and datasets at frequent intervals. To address this need, a novel web meta-crawling method is proposed for discovering Linked Data sources on the Web. We implemented the method in a prototype system named SPARQL Endpoints Discovery (SpEnD). In this paper, we describe the design and implementation of SpEnD, together with an analysis and evaluation of its operation, in comparison to the aforementioned static endpoint repositories in terms of time performance, availability, and size. Findings indicate that SpEnD outperforms existing Linked Data resource discovery methods.

CCS CONCEPTS

• **Information systems → Web searching and information discovery**;

KEYWORDS

Linked Data, Semantic Web, SPARQL Endpoints, Discovery, Search Engine

ACM Reference Format:
Semih Yumusak, Andreas Kamilaris, Erdogan Dogdu, Halife Kodaz, Elif Uysal, and Riza Emre Aras. 2018. A Discovery and Analysis Engine for Semantic Web. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3184558.3191599

*http://spend.semihyumusak.com.tr, https://github.com/semihyumusak/SpEnD
†Corresponding author.

1 INTRODUCTION

Semantic Web standards and technologies[1] are becoming more and more popular on the web, promoting common data formats and protocols for better interoperability among systems and machines, and seamless data integration. Linked Data is a term referring to large structured data sources that conform to Semantic Web standards, specifically the *Resource Description Framework* (RDF) [2] data model. Linked Data sources are published and served on the web through Linked Data endpoints (specifically named as SPARQL endpoints), which allows the Linked Data sources to be queried by using the SPARQL query language. Linked Data sources or endpoints are used in an increasing number of web applications to semantically annotate data [12], in order to enhance search results and facilitate information retrieval and knowledge extraction. Recent statistics[3] refer to more than one thousand datasets and billions of triples in the LOD Cloud [12]. The quality of a Linked Data source is highly dependent on its availability and the content of the data it contains, and it is being tracked by a number of projects [39]. A common problem is that Linked Data sources are not always available, often not responding due to request overload or maintenance. Moreover, some of them stop being maintained and disappear after a few months. Hence, it is critical to monitor, discover, report, and verify Linked Data sources' availability in frequent intervals (e.g. hours, days), in order to allow use of these endpoints online by web clients and web-connected machines. It is also critical to deliver information about the quality, correctness, integrity, and conformance of these datasets [23], in frequent intervals. To the best of our knowledge, there exist four major Linked Data static repositories that regularly keep track of Linked Data resources on the web. These are the Linking Open Data Community project (LOD Cloud) [12], SPARQL Endpoint Status (SPARQLES) [39], LODStats [15] and DataHub[4]. These projects list available Linked Data endpoints, analyzing their quality and connectivity. Additionally, the LOD

[1]http://www.w3.org/2001/sw
[2]http://www.w3.org/RDF
[3]http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/
[4]http://datahub.io/

Cloud presents an image of all Linked Data endpoints discovered, representing data sources with circles and the references between them as links. The SPARQLES project focuses on monitoring the accessibility of SPARQL endpoints. Similarly, LODStats focuses on statistical monitoring of these data sources. Eventually, Datahub is used as a data sharing platform for all other three static repositories which are explained more in detail in Section 2.1. Although the aforementioned repositories contain hundreds of Linked Data endpoints, they cannot discover new online endpoints fast enough nor keep track of offline endpoints efficiently, relying on the publishers to update the information related to the data sources (see Section 2). More importantly, around half of the data sources currently listed by these projects are offline (see Section 5). Aiming to address this need for a web-based dynamic search engine that discovers Linked Data endpoints in frequent intervals (e.g. hours, days), we propose SPARQL Endpoints Discovery (SpEnD), a novel web crawling approach for discovering, analyzing, and publishing Linked Data sources on the Web.

SpEnD is able to serve the most up-to-date list of Linked Data sources and their metadata. Thus, the main contribution of this paper is a new dynamic discovery and analysis method for Linked Data endpoints, which outperforms all other existing approaches for discovery of data sources on the Semantic Web, in terms of availability, and size of Linked Data endpoints discovered (See Section 5). The rest of the paper is organized as follows: Section 2 presents related work in Linked Data and crawling-related studies. Section 3 describes our general methodology for discovering Linked Data endpoints. Then, Section 4 presents the implementation of the SpEnD system while Section 5 evaluates SpEnD by comparing its performance in relation to the other four Linked Data repositories. Finally, Section 6 concludes the paper by pointing out future work.

2 RELATED WORK

Related work spans two categories: (1) Linked Data, metadata and collections, and (2) web crawling and meta-crawling. The former category deals with the evolution of Linked Data, while the latter presents various crawling-based studies which relate to the Semantic Web.

2.1 Linked Data

The work in [6] explains Linked Data as a way to interconnect data sources on the web, so that data becomes machine-readable, semantically annotated, and linked to other data sources. Guides for Linked Data publishing [6] recommend that data is named or identified using URIs, just like other web content, and linked to each other using the RDF model. Hence, Linked Data sources are either published as RDF documents or SPARQL endpoints on the web [6]. If some Linked Data sources are published on the web by following the linked data publishing principles (Availability, Machine-Readability, Open Format, URI identification, and Linked Data)[5] , then it is called Linked Open Data (LOD). An LOD source is qualified to be included in the LOD Cloud project if it meets certain criteria [12]. The LOD Cloud currently consists of more than 1,000 datasets. The most connected source is the DBpedia[6] dataset, derived from Wikipedia. As mentioned in the introduction, there are four major Linked Data collections available in the field. These are the LOD Cloud [12], SPARQLES [39], LODStats [2], and Datahub. The SPARQLES project utilizes the VoID vocabulary to define Linked Data sources, monitoring datasets according to their availability, performance, interoperability and discoverability [8]. In relation to the SPARQLES project, the Dynamic Linked Data Observatory Project [26] makes weekly crawls of Linked Data sources, which are then published on the project's website[7]. LODStats currently has 9,960 datasets indexed on its website[8]. The project reports statistical information about every individual dataset in the VoID vocabulary while the summarized statistical information is reported using the RDF Datacube vocabulary[9]. Finally, Datahub[10] is a Linked Data sharing website, which is used by SPARQLES, LOD Cloud and LODStats projects to share and store metadata about linked datasets. Although Datahub intends to keep track of all Linked Data datasets available on the web, a large number of datasets are not listed (see Section 5). Table 1 compares the aforementioned repositories in terms of data collection method, resources used, definitions employed, metadata formats used and number of endpoints listed.

Table 1: Current Linked Metadata Collections

	Definition	Format	# of Endpoints
LOD Cloud	VoID	Turtle[1]	149
SPARQLES	VoID, Datacube	JSON	496
LODStats	web	HTML	335
Datahub	CKAN API	CKAN API	527

As the LOD cloud gets larger, the need for Linked Data consumption tools increases as well. In order to browse and query Linked Data sources, many browsers have been developed, such as the Tabulator [4], Openlink Data Explorer[11] and Sig.ma [36], as well as search engines such as Swoogle [17], Falcons [10], Sindice [37] and SWSE [20]. In order to improve data availability and reusability, a common need for all these consumer applications is a metadata reference for Linked Data sources. Meta-analysis in this domain is about standardizing retrieval and interpretation of Linked Data sources' description information. In order to provide domain-independent metadata definition, the VoID vocabulary [1] was created for describing metadata about RDF datasets, such as SPARQL endpoint URLs and various statistics (e.g. number of triples, entities, classes and properties).

2.2 Web Crawling and Meta-crawling

The rapid growth of the web increased the need for content-based search and information retrieval, and this was facilitated by the use of web crawlers [7], defined also as spiders, nomads, worms, wanderers, and robots [28]. Early crawlers were differentiated by their storage, indexing (cataloging), search, and crawling methods. Most of them had performance limitations due to connection problems.

[5]http://www.w3.org/DesignIssues/LinkedData.html
[6]http://dbpedia.org/

[7]http://swse.deri.org/dyldo/
[8]http://stats.lod2.eu/
[9]http://www.w3.org/TR/vocab-data-cube/
[10]http://datahub.io/
[11]http://ode.openlinksw.com/

As a solution for the performance issues in web crawling, multi-threaded and distributed crawling has emerged [22], while many different multi-threaded (e.g. Crawler4j[12], Websphinx [30]), and distributed (e.g. Nutch[13], UbiCrawler [7]) web crawlers have been developed. In today's massive web, even these multi-threaded and distributed crawlers fell short of retrieving adequate information in limited time with limited resources. Therefore, some crawlers focused crawling only in a limited scope or domain [34]. The announcement of the Semantic Web [5] in 2001 had an impact on web crawlers too. In the Semantic Web domain, classical web crawling and indexing techniques are incapable of collecting semantically annotated data. Thus, in order to create appropriate crawling and indexing methodologies for Linked Data, new retrieval techniques were developed, such as BioCrawler [3] and MultiCrawler [21]. Also, OntoCrawler [42] used ontology-based website modeling for crawling and classifying classical web documents. Slug [14] was designed to crawl both the classical and the Semantic Web, for the retrieval of relevant documents. On the client side, many search engines were developed, such as Semplore [40], SemSearch [29], Sindice [9], SWSE [20], Falcons [11], and Watson [13], for searching and indexing semantic Web sources and ontologies, or semantic content from the classical Web. As web crawling for the Semantic Web also fell short of retrieving adequate information in limited time and resources [25], meta-crawling was proposed as an effective way to extend classical crawling methods, by including search engines in the crawling stage [27]. Examples of meta-crawling the classical web include SavvySearch [24], Helios [19], and WebCrawler[14]. In the Semantic Web domain, meta-search is not currently employed in the aforementioned semantic search engines [9, 20, 29, 35].

2.3 Our Contribution

Whereas existing projects/repositories for Linked Data sources are mainly based on community-based crowd-sourcing to collect relevant information, in this paper we propose an automatic, dynamic Linked Data endpoints discovery method based on meta crawling. Meta-crawling is a common approach in the web search domain; however, SpEnD approach is unique in applying meta-crawling to the semantic web domain (i.e. the discovery of SPARQL endpoints). In this study, the complete design of SpEnD on discovering, analyzing, and publishing Linked Data endpoints is explained in detail and we compare SpEnD repository with DataHub, LOD Cloud, SPARQLES and LODStats repositories unlike the previous study [44] which consists of describing SpEnD software and comparing SpEnD repository with only DataHub repository. In this way, we address the existing need of the Semantic Web community for a web-based search engine that is capable of discovering Linked Data endpoints and datasets in frequent intervals (e.g. hours, days) which are published anywhere on the web. Although LOD Cloud also claims to use crawling, its crawling service is very basic and inefficient (see our performance comparison in Section 5). We outperform its service by employing continuous meta-crawling (being able to harness any available search engine offering an open API, through a uniform plug and play approach), as an easy and effective way to keep up-to-date metadata about Linked Data sources on the web.

3 METHODOLOGY

SPARQL Endpoints Discovery (SpEnD) is a novel web crawling approach for discovering Linked Data sources on the Web and it consists of four steps: (1) Web crawling, (2) Web-page analysis, (3) Domain learning, and (4) Repository Creation.

Fig. 1 shows the general methodology employed in SpEnD, based on the four steps listed and explained above. In the first step, we propose the use of web crawlers that crawl continuously the web for semantically annotated data and Linked Data sources. A technique to improve the performance of web crawling is by employing meta-crawling (see Section 4.1). An important process here is to identify keywords that can be used to describe Linked Data endpoints, i.e. common patterns that appear in all websites containing Semantic Web-related content (see Section 4.2). A way to achieve this is by manually parsing a large number of available Linked Data sources, trying to identify the most common elements that are evident in all associated web pages. To easily get a complete list of (currently) available Linked Data sources, we can harness existing repositories for Linked Data sources (e.g. LOD Cloud, SPARQLES, LODStats, and DataHub).

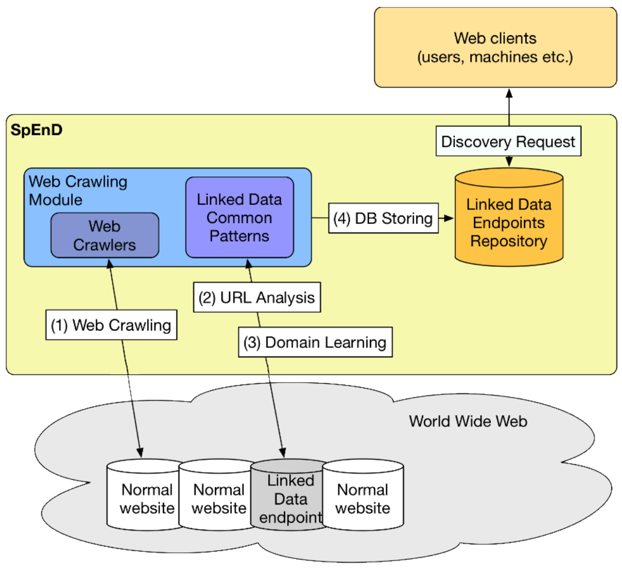


Figure 1: General Methodology

In the second step, web-page analysis is performed to consider whether some of the websites being parsed contain some of the common patterns defined in the first step. If yes, then this is a strong indication that the parsed websites contain Linked Data endpoints and semantically annotated data. There are various techniques to examine whether these endpoints are available and active, with more popular the one of performing some query on the endpoint, expecting to get some response and results (see Section 4.3). In the third step, domain information is harnessed for more targeted

[12]https://code.google.com/p/crawler4j/
[13]http://nutch.apache.org
[14]http://www.webcrawler.com/

and effective web-page analysis. In this case, the current domain is used in various combinations together with the common patterns identified, in order to perform a sophisticated search for Linked Data endpoints and relevant Semantic Web content (see Section 4.4). In the final step, the information about Linked Data endpoints needs to become available to the public so that clients could discover relevant datasets and directly interact with them (or retrieve historical data), and machines could locate relevant information/-knowledge required for their reasoning operations. Thus, a new Linked Data endpoint should be provided by the SpEnD search engine, which could be then used by web clients/machines for discovery of relevant datasets/information. A web and/or desktop application could also be created to serve humans (i.e. through a graphical user interface) as well as machines (i.e. through an API) (see Section 4.5).

4 IMPLEMENTATION

We implemented SpEnD by adopting a meta-crawling approach. Harnessing commercial search engines allowed us to avoid costs for infrastructure and increase system's performance. Our intention is not to promote the use of search engines for the web crawling task (as this would clearly increase the dependency on them for important web-related tasks), but only to demonstrate the possible effectiveness and potential of web crawling for realizing a search engine for the Semantic Web. Moreover, we considered Linked Data sources to be represented as SPARQL endpoints, as SPARQL is currently the most popular way to retrieve and manipulate data stored in RDF format. In the future, we are willing to include other Linked Data endpoints too, if available and widely used. The system's architecture is visualized in Fig. 2. The SpEnD system implementation has five major steps: (1) meta-crawling, (2) creating search keywords for meta-crawling (3) web-page analysis, (4) domain learning and (5) creation of a repository of SPARQL endpoints. These steps are explained in the following sub-sections.

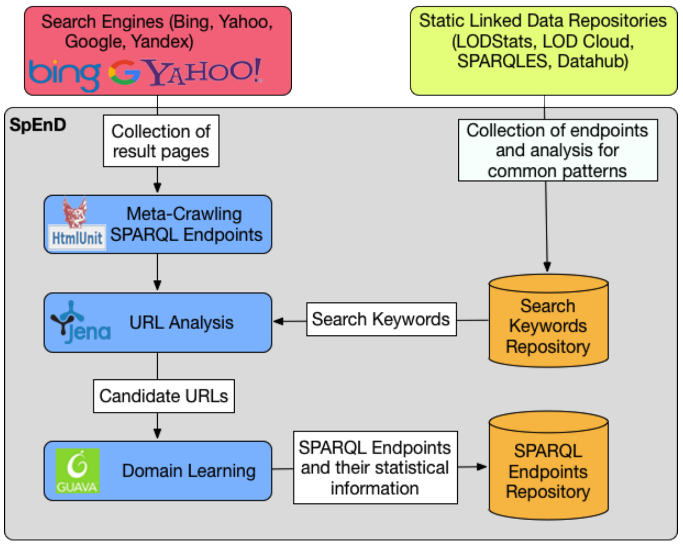

Search Engines (Bing, Yahoo, Google, Yandex)
bing
YAHOO!
Static Linked Data Repositories (LODStats, LOD Cloud, SPARQLES, Datahub)
SpEnD
Collection of result pages
Collection of endpoints and analysis for common patterns
HtmlUnit
Meta-Crawling SPARQL Endpoints
Jena
URL Analysis
Search Keywords
Search Keywords Repository
Candidate URLs
GUAVA
Domain Learning
SPARQL Endpoints and their statistical information
SPARQL Endpoints Repository

Figure 2: System Diagram

4.1 Meta-Crawling SPARQL Endpoints

Search engine APIs[15][16], have various specifications. In the discovery stage, a unified meta-crawling approach (applicable to any search engine) was applied on all four major search engines (Bing, Yahoo, Google and Yandex), based on standardized configuration files. By using this approach, the limitations of the different search API interfaces could be overcome. Besides, a new search engine can be included in the system by simply inserting its XML record in the configuration file. The XML schema is designed to specify the common features and parameters for meta-crawling using any search engine. Some search engines have further limitations to restrict access for common web crawlers. For example, Google search engine results in a 403 HTML error response for (frequent) requests coming from Crawler4J and WebSphinx [30]. In the SpEnD system, a crawling object is created by using parameter values from a configuration file defined by a XML schema, and then a crawling thread is created for each search engine. These threads obey the limitation rules of the corresponding search engines.

Inside the search engine objects, a meta-crawling algorithm is called for every meta search task. The algorithm related to this task is displayed in Algorithm 4.1 as the extraction, parsing and filtering processes are shown in pseudo-code.

In this algorithm, a meta-crawling task is performed for each search keyword on every search engine, as follows:

- At the beginning, the search engine parameters specified in the configuration file are initialized for the search engine object.
- Afterwards, a search task is performed for each search keyword (see Table 2). The algorithm visits all search result pages listed under the search task until the end. Through this meta-crawling process, all URLs hidden under the HTML source code are extracted, irrelevant file types (e.g. pdf, gif, jpeg) and the excluded keywords are filtered out.
- All pages containing some of the search keywords are temporarily saved.

Algorithm 4.1: $\text{METACRAWLING}(SearchEngine, SearchPhrases)$

```
$Parameters \leftarrow GetParams(SearchEngine)$
$pages \leftarrow GetPages(Parameters, SearchPhrases)$
for each $page \in pages$
    $URLs \leftarrow ExtractURLs(page)$
    $URLs.FilterFileTypes(Parameters)$
    $URLs.FilterExclusions()$
    $URLs.save()$
```

4.2 Creating Search Keywords for Meta-Crawling

In general, meta-search keywords are selected by domain experts, an operation which is critical for the complete coverage of the potential results. In this study, our meta-crawling approach starts with

[15]http://datamarket.azure.com/dataset/bing/search
[16]https://developers.google.com/custom-search

a keyword set, which is extracted by our preliminary crawler that collects known SPARQL endpoint pages. This keyword extraction process eliminates the need for domain knowledge in meta-crawling by analyzing and identifying common keyword patterns in the web pages Linked Data resource gateways. In total, 275 HTML pages containing SPARQL endpoints were crawled, by scanning available Linked Data repositories (LOD Cloud, SPARQLES, LODStats, and DataHub), and analyzed to find common patterns evident in all cases by using a term frequency scoring. The most commonly used keywords found are the following: sparql, query, rdf, virtuoso, openlink, inference, iri and endpoint. Besides single words, common HTML tags used along with the aforementioned keywords are the following: label, a, span, title, meta, h1, h2, h3, li, dt, p and option. We then combined these results and gathered a list of meta-crawling search keywords and specific search directives. Table 2 lists some examples from the final list of search keywords. By using this method, we collected and experimented all keywords and phrases which exist in known SPARQL endpoint pages. Further, we enriched the keyword list with search operators[17] for meta-crawling purposes.

Table 2: Sample Search Keywords and Directives [43]

Search Text	Description
sparql	Results having sparql
sparql -language	Results having sparql and not language
"sparql endpoint"	Exact phrase
allintitle: sparql query	Results having title sparql and/or query
intitle:sparql	Results having title sparql
"Virtuoso SPARQL"	Exact phrase
inurl:PoolParty inurl: sparql	Results with PoolParty and sparql in URL
"sparql endpoint" site:gov	Results with extension gov

4.3 Web-page (URL) Analysis

In this step, the HTML source codes of the URLs listed in the search engine result pages are extracted and parsed. These URLs are filtered by using web data extraction methods such as pre-defined regular expressions and filtering criteria [16]. In parallel to the meta-search algorithm, every candidate URL identified is checked if it was tested before. If not, then it is tested whether it is a SPARQL endpoint, based on the simple SELECT query by using the Jena Framework[18].

4.4 Domain Learning

After the preliminary search trials by means of search keywords (as those listed in Table 2), a simple learning algorithm makes one more sophisticated search on the candidate URLs. Although the algorithm in Fig. 4 is capable of locating Linked Data-related web sites, the SPARQL endpoint pages may not show up in the search result pages with the common keywords listed in Table 2. To avoid this possibility, the Pay Level Domain (PLD) names are extracted from the candidate URLs and then a new search query is created by using the "site" keyword (e.g. "sparql site:domain.com"). With this search extension, a more complete search is performed for each domain, by means of the Google Guava libraries[19]. The next step here is to perform a statistical meta-analysis of each Linked Data source discovered. To perform this, the VoID vocabulary is used, which includes a set of properties to define a Linked Data source in terms of numbers, names and statistics. For example, the VoID vocabulary implementation project[20] offers several SPARQL queries to collect statistical information about existing data sources. From the available methods, seven queries were selected for our purposes, listed in Table 3. By sending these queries to the SPARQL endpoints, various statistics were collected. The results from these queries provide useful information about the size and range of the Linked Data sources listed at each repository. These statistical results are also used to filter out same URLs during the URL collection procedure, i.e. URLs having the same number of triples and entities are marked as being the same.

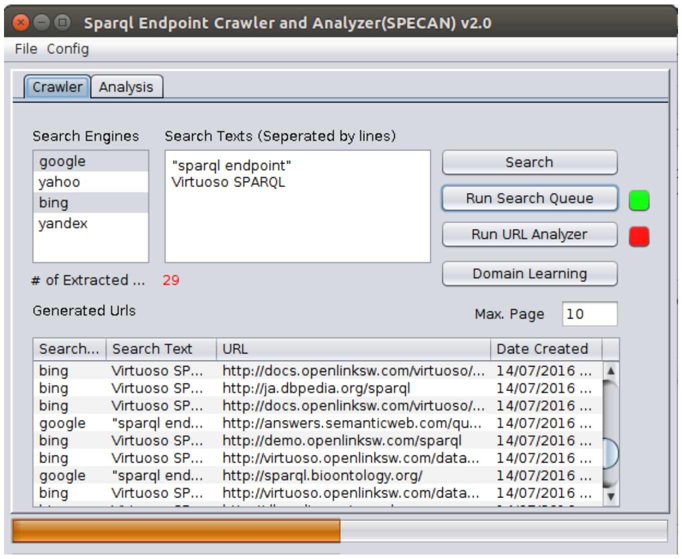
Table 3: SPARQL Queries for Statistical Analysis [43]

ID	SPARQL Query	Definition
1	SELECT COUNT (*) { ?s ?p ?o }	# of triples
2	SELECT COUNT (distinct ?s) { ?s a [] }	# of entities
3	SELECT COUNT (DISTINCT ?s) { ?s ?p ?o } UNION { ?o ?p ?s } FILTER (!isBlank (?s) && !isLiteral (?s))	# of distinct resource URIs
4	SELECT COUNT (distinct ?o) { ?s rdf:type ?o }	# of distinct classes
5	SELECT count (distinct ?p) { ?s ?p ?o }	# of distinct predicates
6	SELECT COUNT (DISTINCT ?s) { ?s ?p ?o }	# of distinct subject nodes
7	SELECT COUNT (DISTINCT ?o) ?s ?p ?o filter (!isLiteral (?o))	# of distinct object nodes

4.5 Repository Creation

In this final step, a repository has been created for storing the SPARQL endpoints discovered. This repository involves a Virtuoso RDF store, for permanently storing the endpoints discovered, and also a Java application that provides a graphical interface for web and desktop clients to run the SpEnD system. This Java application can perform meta-crawling on search engines, analyze the resulting candidate URLs, and perform statistical analysis on the discovered SPARQL endpoints. The main screen of this application (see Figure 3) performs search engine crawling (see Section 4.1) and web-page analysis (see Section 4.3). The screen has a search text input box for search keywords and queries (see Section 4.2), a list of search engines to be selected, and a table grid for displaying the ongoing search results.

[17] http://www.googleguide.com/advanced_operators_reference.html
[18] https://jena.apache.org/
[19] https://github.com/google/guava
[20] https://code.google.com/p/void-impl/wiki/SPARQLQueriesForStatistics



Search Engines: google, yahoo, bing, yandex

Search Texts (Seperated by lines):

```
"sparql endpoint"
Virtuoso SPARQL
```

# of Extracted ... 29

Generated Urls

Max. Page 10

Search...	Search Text	URL	Date Created
bing	Virtuoso SP...	http://docs.openlinksw.com/virtuoso/...	14/07/2016 ...
bing	Virtuoso SP...	http://ja.dbpedia.org/sparql	14/07/2016 ...
bing	Virtuoso SP...	http://docs.openlinksw.com/virtuoso/...	14/07/2016 ...
google	"sparql end...	http://answers.semanticweb.com/qu...	14/07/2016 ...
bing	Virtuoso SP...	http://demo.openlinksw.com/sparql	14/07/2016 ...
bing	Virtuoso SP...	http://virtuoso.openlinksw.com/data...	14/07/2016 ...
google	"sparql end...	http://sparql.bioontology.org/	14/07/2016 ...
bing	Virtuoso SP...	http://virtuoso.openlinksw.com/data...	14/07/2016 ...

Figure 3: Desktop/Web Application: Crawler Tab [43]

Additionally, through the analysis interface, results from the other four projects can be traced in the Dataset Collection container. Additional features (in the Monitoring&Analysis container) include status monitoring to trace the availability of each SPARQL endpoint and statistical analysis of each SPARQL endpoint by using the SPARQL queries listed in Table 3. The software has been designed to run continuously, enabling an on-going crawling and analysis of SPARQL endpoints available on the web.

5 EVALUATION

Evaluation focuses on two different aspects. First, the search results produced by harnessing various search engines (Bing, Yahoo, Google and Yandex) are analyzed and then, the SpEnD SPARQL endpoints repository is compared with the LOD Cloud, SPARQLES, LODStats, and DataHub repositories. In the following experiments, we left the SpEnD system to work for 24 hours and then we collected the results. All the results of this experimental study, including raw search findings, are available on the SpEnD project's website[21].

5.1 Search Engine Results

A total of 117K unique URLs have been extracted from search engines. Fig. 4 shows the number of crawled URLs vs. the total number of (unique and duplicate) SPARQL endpoints found. After about 44K URLs collected, the number of unique endpoints discovered does not increase anymore. Between 18K and 19K collected URLs, there is a significant increase on the number of unique endpoints discovered. The reason is the operation of the domain learning task (see Section 4.4), which starts after the initial querying of search texts (see Table 2). More than 1K unique SPARQL endpoints have been discovered in total during this experiment. After the discovery process, these 1,037 unique endpoints were analyzed to consider their availability and meta-information. Out of them, 211 were not available (listed in the search engine results but not accessible). Of the remaining endpoints, 168 were also eliminated after further analysis[22], as they were listed in the results set more than once. Although there is no one-to-one correspondence between datasets and SPARQL endpoints, SPARQL endpoints with duplicate datasets (one of them) are excluded from further analysis, in order to compare the number of unique Linked Data sources available more precisely. Although these endpoints are excluded from analysis, they are not excluded from the final list of published endpoints, where such endpoints are marked with a **sameAs** connection and not removed from the final repository list. Hence, a total of 658 unique online endpoints were finally stored in our SPARQL endpoints repository for analysis.

[21]http://spend.semihyumusak.com.tr/

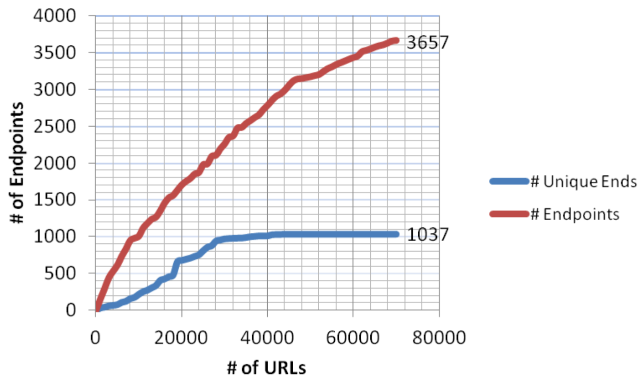


Figure 4: Number of Endpoints vs. Number of Crawled URLs [43]

Table 4 offers some insights over the 658 discovered SPARQL endpoints, in terms of search terms which result in more than 60 endpoints. For example, by searching the "sparql query" phrase in all search engines, 207 unique SPARQL endpoints were discovered. Some endpoints have been apparently discovered by multiple search terms. For example, the search terms "sparql -w3" and "sparql -wiki" returned almost the same SPARQL endpoint URLs.

5.2 Comparison of SpEnD with Existing SPARQL Endpoints Repositories

In this subsection, the results from SpEnD's 24-hour discovery process were compared with those from the other four major endpoint repositories (LOD Cloud, LODStats, SPARQLES, and Datahub). In order to determine the status of every endpoint listed on the four repositories plus SpEnD, a simple SPARQL query was sent daily to each listed SPARQL endpoint during June, 2016, for a total period of one month. Table 5 lists the number of online and offline endpoints at those repositories as well as at the SpEnD dataset. Almost half of the SPARQL endpoints listed in the repositories are offline (passive), which is an indication that the SPARQL endpoint collection is not updated frequently on these repositories. Moreover, there are 211 offline endpoints discovered in the SpEnD dataset.

[22]Each dataset was inspected using the queries listed in Table 3 (number of triples, entities, classes, etc.). If two endpoints had exactly the same results for all the queries, then they were considered as the same dataset (but duplicated in two separate URL endpoints).

Table 4: Search terms resulting in more than 60 endpoints

Search Text	# of Endpoints
"sparql endpoint" site:PLD	518
sparql query	207
"sparql endpoint"	179
sparql -language	171
inurl:sparql	150
allintitle: sparql query	144
sparql -w3	126
sparql -wiki	125
sparql -blog -wiki -w3 -pdf -news	124
sparql -blog -wiki -w3 -pdf	120
sparql -blog	115
allinurl:sparql data	105
"sparql endpoint" -blog -wiki -w3 -pdf	93
"Virtuoso SPARQL Query Editor"	87
"sparql endpoint" -blog	84
"sparql endpoint" -wiki	76
"sparql endpoint"	66
"Virtuoso SPARQL Query Editor"	65
"sparql endpoint" -w3	61

Table 5: Comparison of SPARQL Endpoints Discovered in Terms of their Availability [43]

Dataset	Available Online			Offline	Total
	High	Low	Total Online		
Datahub	210	63	273	254	527
LOD Cloud	69	13	82	67	149
LodStats	136	39	175	160	335
SPARQLES	205	61	266	230	496
SpEnD	**537**	**121**	**658**	**211**	**869**

Fig. 5 visualizes the total number of distinctive and common endpoints found by all projects. The SpEnD repository has 520 distinctive endpoints. However, there are also some SPARQL endpoints that could not be discovered by SpEnD, because they were not linkable with the rest of the web, hence search engines could not discover them. Fig. 6 shows only the active and available endpoints as listed in the four repositories and SpEnD dataset. As this figure shows, 224 out of the 277 active endpoints listed in other collections, were found by SpEnD as well (80.9% Accuracy). SPARQLES and LOD Cloud had no distinctive URLs, while only three distinctive URLs were listed in Datahub and only one distinctive URL listed in LODStats. The Datahub repository mostly covers other lists in terms of active URLs, except SpEnD. Although Fig. 5 shows a significant amount of unique URLs in SpEnD data collection (434 endpoints), some of these URLs are not domain-significant, i.e. there are several SPARQL endpoints under the same domain names. Thus, we also analyzed the SPARQL endpoint URLs based on their PLDs. Fig. 7 shows the number of active PLDs and Table 6 lists those PLDs including more than 10 SPARQL endpoint URLs. In this case, SpEnD discovers 119 out of 130 domains listed in other collections (91% accuracy).

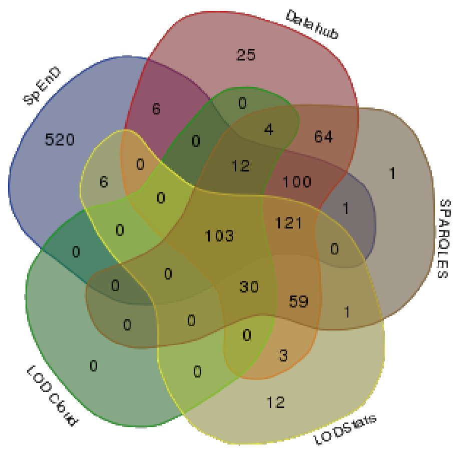


Figure 5: Set of Repositories' Total # of SPARQL Endpoints [43]

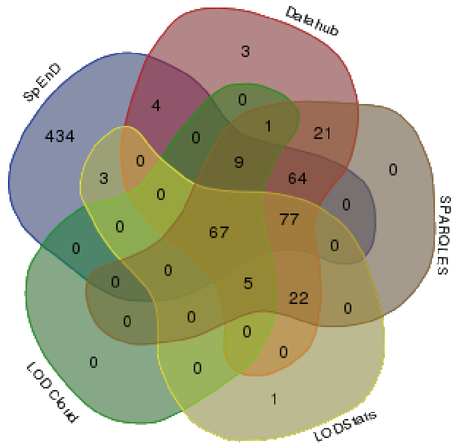


Figure 6: Set of Repositories' # of Available SPARQL Endpoints [43]

We note that the 434 unique endpoint URLs and 146 significant domains discovered by SpEnD project are not validated/qualified by using quality measures (Acosta et al. 2013; Kontokostas and Westphal 2014; Mendes, Mühleisen, and Bizer 2012); however, the authors consider them still valuable as they are not listed and included in any other collections.

In Table 7, the total number of online and offline PLDs and endpoint URLs are listed, comparing SpEnD with the other four repositories. The SpEnD dataset has the highest number of PLDs (265) and the highest number of endpoint URLs (658).

6 CONCLUSION AND FUTURE WORK

Although static repositories such as LOD Cloud, SPARQLES, and LODStats are popular on discovering and monitoring Linked Data sources, this paper shows that they are not capturing the whole

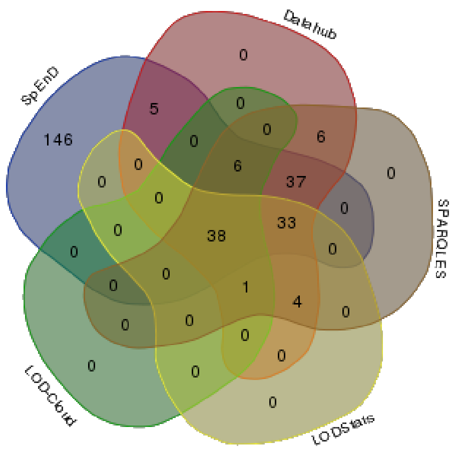


Figure 7: Set of Repositories' Active SPARQL Set Domains (PLD) [43]

Table 6: Number of SPARQL Endpoint URLs under the same PLD [43]

Pay Level Domains	#Endpoints (>10)
rkbexplorer.com	65
b3kat.de	58
insee.fr	34
dbpedia.org	27
fundacionctic.org	23
data.gov.uk	21
270a.info	15
ign.fr	13
linkeddata.es	13
eagle-i.net	12

Table 7: Number of PLDs and URLs Discovered by SpEnD and Repositories [43]

	# PLDs		# Endpoints	
	Online	Offline	Online	Offline
SpEnD	**265**	59	**658**	211
SPARQLES	125	110	266	230
LodStats	76	78	175	160
LOD Cloud	45	34	82	67
Datahub	130	119	273	254

spectrum of Linked Data endpoints available on the web, nor effectively tracing new endpoints identifying those going persistently offline. This paper addresses this gap by proposing a dynamic approach for discovery of Linked Data endpoints available on the web. We have presented SpEnD, which is a discovery engine for the Semantic Web based on web crawling. Our implementation involved a meta-crawling approach taking advantage of existing commercial search engines, to demonstrate the potential of web crawling in terms of effectively tracing Linked Data resources in less than 24 hours. An evaluation procedure indicated that SpEnD outperforms static repositories in regard to time performance, availability, and size of Linked Data endpoints discovered. A direct comparison revealed that the SpEnD dataset included significantly more SPARQL endpoints, PLDs, triples, entities, resource URIs, classes, predicates, subject nodes and object nodes than all other repositories. Our SPARQL endpoints collection is growing day by day and is available for further analysis and research by the academic community, even though our dataset is not yet rich on semantic annotations and classification. Regular crawling tasks are daily performed, and newly discovered SPARQL endpoints are incrementally added and published on the project's web site. As future work, our next step will be a semantic analysis [38, 41], and ranking [18] of the collected SPARQL endpoints. This will help to get a better understanding about the contents of the discovered Linked Data sources, making it possible to classify SPARQL endpoints according to their domain or context [31–33]. Moreover, since it was observed that a small number of endpoints was not discovered by SpEnD (due to search engines' limitations) but were still available on the static repositories, we will put some effort into improving our overall approach to better include, in an automatic way, endpoints and datasets listed only on those repositories.

7 ACKNOWLEDGEMENT

This research paper is written based on the results accomplished in the PhD thesis prepared by Semih Yumusak and enhanced with the help of additional authors. This research is partly supported by The Scientific and Technological research council of Turkey with grant number 1059B141500052 (Ref.No: B.14.2. TBT.0.06.01-21514107-020-155998).

REFERENCES

[1] Keith Alexander and Michael Hausenblas. 2009. Describing linked datasets-on the design and usage of void, the'vocabulary of interlinked datasets. In *Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09).*
[2] Sören Auer, Ivan Ermilov, Jens Lehmann, and Michael Martin. [n. d.]. LODStats - a statement-stream-based approach for gathering comprehensive statistics about RDF datasets. ([n. d.]). http://lodstats.aksw.org/
[3] Alexandros Batzios, Christos Dimou, Andreas L Symeonidis, and Pericles A Mitkas. 2008. BioCrawler: An intelligent crawler for the semantic web. *Expert Systems with Applications* 35, 1-2 (2008), 524–530. https://doi.org/10.1016/j.eswa.2007.07.054
[4] Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. 2006. Tabulator : Exploring and Analyzing linked data on the Semantic Web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop.*
[5] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The Semantic Web. *Scientific American* May (2001).
[6] Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked data-the story so far. *International Journal on Semantic Web and Information Systems* 5, 3 (2009), 1–22.
[7] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. 2004. UbiCrawler: a scalable fully distributed Web crawler. *Software: Practice and Experience* 34, 8 (jul 2004), 711–726. https://doi.org/10.1002/spe.587
[8] C Buil-Aranda and Aidan Hogan. 2013. SPARQL Web-Querying Infrastructure: Ready for Action?. In *The Semantic Web–ISWC 2013.* Springer Berlin Heidelberg, 277–293.
[9] Stephane Campinas and Diego Ceccarelli. 2011. The Sindice-2011 dataset for entity-oriented search in the web of data. In *Proceedings of the 1st International Workshop on Entity-Oriented Search (EOS).* 26–32. http://research.microsoft.com/en-us/um/beijing/events/eos2011/13.pdf
[10] Gong Cheng, Weiyi Ge, Honghan Wu, and Yuzhong Qu. 2008. Searching Semantic Web Objects Based on Class Hierarchies. *LDOW* (2008). http://www.ambuehler.ethz.ch/CDstore/www2008/events.linkeddata.org/ldow2008/papers/12-cheng-ge-searching-semantic-web-objects.pdf

[11] Gong Cheng and Yuzhong Qu. 2009. Searching Linked Objects with Falcons. *International Journal on Semantic Web and Information Systems* 5, 3 (2009), 49–70. https://doi.org/10.4018/jswis.2009081903

[12] Richard Cyganiak and Anja Jentzsch. 2014. The Linking Open Data cloud diagram. (2014). http://lod-cloud.net/

[13] M D'Aquin, E Motta, Jérôme Euzenat, Inria Grenoble Rhône-alpes, Walton Hall, and Milton Keynes. 2011. Watson , more than a Semantic Web search engine. *Semantic Web* 2, 1 (2011), 55–63.

[14] Leigh Dodds. 2006. Slug: A semantic web crawler. In *Proceedings of Jena User Conference*.

[15] Ivan Ermilov, Jens Lehmann, Michael Martin, and Sören Auer. 2016. *LODStats: The Data Web Census Dataset*. Springer International Publishing, Cham, 38–46. https://doi.org/10.1007/978-3-319-46547-0_5

[16] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. 2014. Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems* 70 (2014), 301–323. https://doi.org/10.1016/j.knosys.2014.07.007

[17] Tim Finin, Li Ding, Tim Finnin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. 2004. Swoogle : A Search and Metadata Engine for the Semantic Web. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*. ACM, 652–659.

[18] Jose Maria Garcia, Martin Junghans, David Ruiz, Sudhir Agarwal, and Antonio Ruiz-Cortes. 2013. Integrating semantic Web services ranking mechanisms using a common preference model. *Knowledge-Based Systems* 49 (2013), 22–36. https://doi.org/10.1016/j.knosys.2013.04.007

[19] a. Gulli and A. Signorini. 2005. Building an open source meta-search engine. *Special interest tracks and posters of the 14th international conference on World Wide Web - WWW '05* (2005), 1004. https://doi.org/10.1145/1062745.1062840

[20] Andreas Harth, Aidan Hogan, Renaud Delbru, Sean O Riain, and Stefan Decker. 2007. SWSE : Answers Before Links !. In *Semantic Web Challenge*.

[21] Andreas Harth, J Umbrich, and Stefan Decker. 2006. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *ISWC*. 258–271. http://link.springer.com/chapter/10.1007/11926078

[22] Allan Heydon and Marc Najork. 1999. Mercator: A scalable, extensible Web crawler. *World Wide Web* 2, 4 (1999), 219–229. https://doi.org/10.1023/A:1019213109274

[23] Aidan Hogan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, and Stefan Decker. 2012. An empirical survey of Linked Data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web* 14 (jul 2012), 14–44. https://doi.org/10.1016/j.websem.2012.02.001

[24] Adele E Howe and Daniel Dreilinger. 1997. SavvySearch: A Meta-Search Engine that Learns which Search Engines to Query. *AI Magazine* 18, 2 (1997), 12–25. https://doi.org/10.1.1.43.8157

[25] Robert Isele, Christian Bizer, and Andreas Harth. 2010. LDSpider An open-source crawling framework for the Web of Linked Data. In *International Semantic Web Conference*. 6–9.

[26] Tobias Käfer and J Umbrich. 2012. Towards a Dynamic Linked Data Observatory. In *Linked Data on the Web at WWW Conference*, Vol. 1380. http://events.linkeddata.org/ldow2012/slides/ldow2012-slides-14.pdf

[27] Andrew Kenneth, John Mcmahon, and C A Us. 2012. United States Patent US7805432 B2 Meta Search Engine. (2012). https://doi.org/10.1197/jamia.M1139.Adar

[28] Jon P. Knight. 1996. Resource discovery on the internet. *New Review of Information Networking* 2 (1996), 3–14. https://doi.org/10.1080/13614579609516865

[29] Yuangui Lei, Victoria Uren, and Enrico Motta. 2006. SemSearch : A Search Engine for the Semantic Web. In *Managing Knowledge in a World of Networks*. Springer Berlin Heidelberg, 238–245.

[30] Robert C. Miller and Krishna Bharat. 1998. SPHINX: a framework for creating personal, site-specific Web crawlers. *Computer Networks and ISDN Systems* 30 (1998), 119–130. https://doi.org/10.1016/S0169-7552(98)00064-6

[31] B. Mutlu, M. Mutlu, K. Oztoprak, and E. Dogdu. 2016. Identifying trolls and determining terror awareness level in social networks using a scalable framework. In *2016 IEEE International Conference on Big Data (Big Data)*. 1792–1798. https://doi.org/10.1109/BigData.2016.7840796

[32] Kasim Oztoprak. 2015. Profiling subscribers according to their internet usage characteristics and behaviors. *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015* (2015), 1492–1499. https://doi.org/10.1109/BigData.2015.7363912

[33] Kasim Oztoprak. 2016. Subscriber Profiling for Connection Service Providers by Considering Individuals and Different Timeframes. *IEICE Transactions on Communications* E99.B, 6 (2016), 1353–1361. https://doi.org/10.1587/transcom.2015EBP3467

[34] V Shah, Riya Patni, Vivek Patani, and Rhythm Shah. 2014. Understanding Focused Crawler. *International Journal of Computer Science & Information Technologies* 5, 5 (2014), 6849–6852. http://ijcsit.com/docs/Volume5/vol5issue05/ijcsit20140505183.pdf

[35] Thanh Tran, Haofen Wang, and Peter Haase. 2009. Hermes: Data Web search on a pay-as-you-go integration infrastructure. *Web Semantics: Science, Services and Agents on the World Wide Web* 7, 3 (sep 2009), 189–203. https://doi.org/10.1016/j.websem.2009.07.001

[36] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. 2010. Sig.ma: Live views on the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web* 8, 4 (nov 2010), 355–364. https://doi.org/10.1016/j.websem.2010.08.003

[37] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. 2007. Sindice. com: Weaving the open linked data. In *The Semantic Web*. Springer Berlin Heidelberg, 552–565.

[38] Elif Uysal, Semih Yumusak, Kasim Oztoprak, and Erdogan Dogdu. 2017. Sentiment Analysis for the Social Media : A Case Study for Turkish General Elections Categories and Subject Descriptors. *Proceedings of the SouthEast Conference. ACM* (2017), 215–218.

[39] PY Vandenbussche, CB Aranda, Aidan Hogan, and J Umbrich. 2013. Monitoring the Status of SPARQL Endpoints. In *International Semantic Web Conference (Posters & Demos)*, Vol. 1380. 3–6.

[40] Haofen Wang, Qiaoling Liu, Thomas Penin, Linyun Fu, Lei Zhang, Thanh Tran, Yong Yu, and Yue Pan. 2009. Semplore: A scalable IR approach to search the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web* 7, 3 (sep 2009), 177–188. https://doi.org/10.1016/j.websem.2009.08.001

[41] Zhichun Wang, Juanzi Li, Yue Zhao, Rossi Setchi, and Jie Tang. 2013. A unified approach to matching semantic data on the web. *Knowledge-Based Systems* 39 (2013), 173–184. https://doi.org/10.1016/j.knosys.2012.10.015

[42] Sheng-Yuan Yang. 2010. OntoCrawler: A focused crawler with ontology-supported website models for information agents. *Expert Systems with Applications* 37, 7 (jul 2010), 5381–5389. https://doi.org/10.1016/j.eswa.2010.01.018

[43] Semih Yumusak. 2017. *A Novel Method to Discover and Analyze Linked Data Sources*. Ph.D. Dissertation.

[44] Semih Yumusak, Erdogan Dogdu, Halife Kodaz, Andreas Kamilaris, and Pierre-Yves Vandenbussche. 2017. SpEnD: Linked Data SPARQL Endpoints Discovery Using Search Engines. *IEICE Transactions on Information and Systems*, E100-D, 4 (2017), 758–767. https://doi.org/10.1587/transinf.2016DAP0025 arXiv:1608.02761