IEICE TRANS. INF. & SYST., VOL.E100–D, NO.4 APRIL 2017

PAPER Special Section on Data Engineering and Information Management

SpEnD: Linked Data SPARQL Endpoints Discovery Using Search Engines*

Semih YUMUSAK[†a)], ***Student Member***, **Erdogan DOGDU**[††], **Halife KODAZ**[†††], **Andreas KAMILARIS**[††††], ***and*** **Pierre-Yves VANDENBUSSCHE**[†††††], ***Nonmembers***

SUMMARY Linked data endpoints are online query gateways to semantically annotated linked data sources. In order to query these data sources, SPARQL query language is used as a standard. Although a linked data endpoint (i.e. SPARQL endpoint) is a basic Web service, it provides a platform for federated online querying and data linking methods. For linked data consumers, SPARQL endpoint availability and discovery are crucial for live querying and semantic information retrieval. Current studies show that availability of linked datasets is very low, while the locations of linked data endpoints change frequently. There are linked data respsitories that collect and list the available linked data endpoints or resources. It is observed that around half of the endpoints listed in existing repositories are not accessible (temporarily or permanently offline). These endpoint URLs are shared through repository websites, such as Datahub.io, however, they are weakly maintained and revised only by their publishers. In this study, a novel metacrawling method is proposed for discovering and monitoring linked data sources on the Web. We implemented the method in a prototype system, named SPARQL Endpoints Discovery (SpEnD). SpEnD starts with a "search keyword" discovery process for finding relevant keywords for the linked data domain and specifically SPARQL endpoints. Then, the collected search keywords are utilized to find linked data sources via popular search engines (Google, Bing, Yahoo, Yandex). By using this method, most of the currently listed SPARQL endpoints in existing endpoint repositories, as well as a significant number of new SPARQL endpoints, have been discovered. We analyze our findings in comparison to Datahub collection in detail.

key words: *linked data, semantic Web, SPARQL endpoint, endpoint discovery, metasearch, knowledge graph*

1. Introduction

Semantic Web standards and technologies [1] are being used widely in today's Web. "Linked data" is a term referring to large structured data sources that conform to Semantic Web standards, such as the Resource Description Framework (RDF) data model. Linked data sources are the main Semantic Web data sources on the current Web and they are used in many applications, from enhancing search results to open knowledge extraction. As Semantic Web technologies are getting more popular in use, linked data sources are continuously growing in number and size. Recent statistics [2] list more than one thousand data sets and billions of triples in the cloud.

The quality of a linked data source is highly dependent on its availability and content. The availability and content quality for linked data sources are being tracked by a number of projects [2], [3]. It is clear that linked data sources are not always alive. Sometimes they do not respond due to request overload, maintenance, or they may go offline permanently. Therefore, it is critical to monitor, report and provide verification about these data sources' accessibility continuously, if the future Web aims to utilize these data sources online and in real-time. It is also critical to deliver information about the quality, correctness, integrity, and conformance of these datasets [4]. Although there are existing projects for monitoring, reporting and analysis of these data sources, there is a lack of a methodology on discovering new data sources. In order to fill this gap, we have created an open source linked data crawling and monitoring tool named SpEC v1.0**. With the help of this tool, we have crawled the Web via search engines in order to discover new data sources.

The related work about linked data and crawling is presented in Sect. 2. Our method of discovering unknown SPARQL endpoints is presented in Sect. 3 and our SPARQL Endpoints Discovery (SpEnD) system workflow we developed is described in Sect. 4, together with the SpEC software. Then, SpEnD metadata collection is comparatively analyzed with the other metadata collections in Sect. 5. Finally, Sect. 6 concludes the paper and indicates future work.

2. Related Work

Related work spans into two categories: (1) linked data and (2) Web crawling, linked data source crawling and metacrawling. In the former, the current status of linked data studies are explained. In the latter, classical crawling methods, linked data crawling and metacrawling methods are described, in relation to the Semantic Web.

Manuscript received June 29, 2016.
Manuscript revised November 6, 2016.
Manuscript publicized January 17, 2017.

†The author is with Computer Eng. Dept., KTO Karatay Univ., Turkey.
††The author is with Computer Eng. Dept., Cankaya University, Turkey.
†††The author is with Computer Eng. Dept., Selcuk University, Turkey.
††††The author is with Insight Research Centre for Data Analytics, Turkey.
†††††The author is with the Fujitsu Ireland Limited, Ireland.

*This research is supported by The Scientific and Technological research council of Turkey with grant number 1059B141500052.

a) E-mail: semih.yumusak@karatay.edu.tr

DOI: 10.1587/transinf.2016DAP0025

**https://github.com/semihyumusak/SpEnD/releases/tag/v1.0

2.1 Linked Data

Linked data is a term for expressing structured Semantic Web data, which is data on the Web that is linked to each other using URIs and RDF. Bizer et al. [5] explain linked data as a way to interconnect data sources on the Web, so that this data becomes machine-readable, semantically annotated and linked to other data sources. The basic standard for linked data publishing [6] recommends that data is named or identified using URIs, just like other Web content, and linked to each other using the RDF model. Linked data sources are either published as RDF documents or SPARQL endpoints on the Web [5]. If a linked data source is published on the Web by following the linked data publishing principles[†], it is called "Linked Open Data" (LOD). A LOD source is qualified to be included in the "Linking Open Data Project" (LOD Cloud) as long as it meets certain criteria[††]. In the LOD Cloud, all data sources are classified and defined by meta descriptions. In order to provide these meta descriptions, VoID vocabulary is commonly used [7]. VoID vocabulary recommends specific terms and patterns to describe linked data sources. For instance, a SPARQL endpoint URL of a dataset can be expressed by the `void:sparqlEndpoint` property in the VoID vocabulary. Moreover, statistical data about datasets can be expressed by using VoID properties such as the number of triples (`void:triples`), the number of entities (`void:entities`), the number of classes (`void:classes`), and the number of properties (`void:properties`). In this paper, we use these statistics to examine and compare the linked data sources we collected in SpEnD.

2.2 Semantic Web Crawling and Metacrawling

With the announcement of the Semantic Web [8] in 2001, the scope for Web crawlers had changed. In the Semantic Web domain, classical Web crawling and indexing techniques for HTML documents are incapable of collecting knowledge-enhanced (meta) data. Thus, in order to create proper crawling and indexing methodologies for semantically annotated data, new retrieval techniques have been proposed. BioCrawler [9] was created as an intelligent crawler for evaluating semantic contents of Web pages using the BioTope framework [10] as its engine. MultiCrawler [11] was developed as a pipelined crawler and indexer for collecting semi-structured data from the classical Web and the Semantic Web. OntoCrawler [12] used ontology-based website modeling for crawling and classifying classical Web documents.

However, linked data source crawling also fell short of retrieving adequate data in a limited time and resource [13]. Thus, more powerful Web crawlers are needed, which have the capacity of continuously crawling a large portion of the visible Web. A way to achieve this without investing huge amounts in computing infrastructures, is by harnessing the operation of classical existing search engines. This approach could constitute an effective way to extend classical crawling methods by including search engines in the crawling stage [14], creating a metacrawling scenario.

The term metacrawling on the classical Web is usually discussed under the term "meta search engine" [15]–[17]. Retrieving and combining results from search engines in the classical Web domain is not something new, and meta search engines include SavvySearch [18], Helios [19], and WebCrawler. However, on the Semantic Web domain, meta search is not currently employed on any of the existing semantic search engines [20]–[24].

3. Methodology

The following subsections describe the methodology followed in this paper to develop SPARQL Endpoints Discovery (SpEnD), a novel metacrawling method for discovering and monitoring linked data sources on the Web, based on the following two steps:

1. Discovery of SPARQL Endpoints.
2. Meta analysis of linked data sources discovered.

3.1 Discovery of SPARQL Endpoints

SPARQL endpoint discovery is a step by step approach to crawl public search engines based on different search queries. In the crawling stage, a unified novel metacrawling methodology (applicable to any search engine) was applied on major search engines. By using this methodology, the limitations of the search API interfaces provided by existing search engines can be overcome. Moreover, a new search engine can be included in the system by simply inserting its XML record in a configuration file we have designed for this purpose. A sample record of this configuration file is listed in Fig. 1. Its XML schema is designed to specify the common features and parameters to crawl any search engine.

Some search engines have further limitations to restrict access for common Web crawlers such as Crawler4J[†††] and

```
<searchEngine>
 <name>yahoo</name>
 <excludedWords>yahoo|bing|yimg|zenfs</
     excludedWords>
 <baseUrl>https://www.yahoo.com</baseUrl>
 <queryTextBoxName>p</queryTextBoxName>
 <submitButtonId>search-submit</submitButtonId>
 <submitButtonName></submitButtonName>
 <defaultBrowser>Chrome</defaultBrowser>
 <nextButtonIdentifier>Next</nextButtonIdentifier
     >
 <useUrlRedirection>false</useUrlRedirection>
 <waitIntervalMs>1000</waitIntervalMs>
</searchEngine>
```

Fig. 1 Search engine XML description

[†]http://www.w3.org/DesignIssues/LinkedData.html
[††]http://lod-cloud.net
[†††]https://code.google.com/p/crawler4j/

WebSphinx [25]. For example, Google search engine returns an error response 403[††††] for requests coming from Crawler4J and WebSphinx. Frequent requests from crawlers and unknown browsers are blocked by Google. However, if the search request is made by simulating a Web browser (e.g. Google Chrome, Mozilla, Internet Explorer) using the HtmlUnit[†††††] browser library, all search engines return reliable end-user results. Hence, we developed a metacrawler called SpEnD to crawl search engine results for specific search queries. In SpEnD, search engine browser objects are initialized by using the XML parameters, then a crawling thread is created for each search engine.

Figure 1 shows a sample XML record created for the Yahoo search engine, which has a query box named 'p', submit button named 'search-submit', and next button named 'Next' for paging through results.

3.1.1 Creating Search Keywords for Metacrawling

In order to identify metacrawling keywords that can be used to find SPARQL endpoints, we collected a set of SPARQL endpoints' HTML page sources (enspoints listed in Datahub.io website) and analyzed them to find their common patterns. For this purpose, we created a words list for each document by excluding the English stop words. Then, a simple term endpoint frequency (f) was calculated for each word included in SPARQL endpoint web pages. Based on the calculated f scores, the extracted words were sorted in descending order which gives us a list of most commonly used words at the top of the list. The f scoring function is described as follows:

$$f(t) = n_t/N$$

where
n_t: number of documents where the term exists
N: total number of documents
t: a word or a phrase

In addition to single words, we also calculated frequency scores for key phrases mentioned in specific HTML tags in the same HTML source documents. These HTML tags are: `label, a, span, title, meta, h1, h2, h3, li, dt, p, option` . We finally combined these results and gathered a list of metacrawling search keywords and specific search directives. This step requires some trials and tests by combining different words and phrases together in a single search.

3.1.2 Metacrawling Linked Data Endpoints

In this step, URLs listed in search engine result pages are extracted by parsing the HTML source code. These URLs are then filtered by using Web data extraction methods such as pre-defined regular expressions and filtering criteria [26].

[††††]http://www.w3.org/Protocols/HTTP/HTRESP.html
[†††††]http://htmlunit.sourceforge.net/

Algorithm 3.1: MetaSearch(*EngineList*[], *QueryList*[])

```
EngineList[] : Set of search engines
QueryList[] : Set of keyword queries
comment: Performs a meta-search for the queries

for each SearchEngine ∈ EngineList[]
    p ← GetXMLParams(SearchEngine)
    for each query ∈ QueryList[]
        currentPage ← GetFirstPage(p, query, SearchEngine)
        while currentPageIsNotEmpty
            URLList ← ExtractURLsFromHTML(currentPage)
            URLList.removeURLsWithIrrelevantFileTypes()
            URLList.removeURLsWithExcludedKeywords()
            URLList.save()
            currentPage ← GetNextPage(currentPage)
```

```
SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Fig. 2 Basic SPARQL query example

In Algorithm 3.1, the metacrawling, extraction, and filtering processes are defined. In this algorithm, a meta-search task is performed for each search keyword at every search engine. At the beginning, the search engine parameters are initialized for the search engine object. Afterwards, a search task is performed for each search keyword. The algorithm visits all search result pages listed under the search task until the end. Through this metacrawling 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.

3.1.3 URL Analysis

After the metacrawling step, every URL is checked for validity. This is a two-step process. First, the URL is checked if it is tested and listed in Datahub repository. If the URL is listed in Datahub repository as a SPARQL endpoint, the URL is then automatically marked as a valid SPARQL endpoint. If it is not listed in Datahub, then it is tested for validity by sending the following simple and generic SPARQL query to the URL endpoint; if the URL returns some results to this query, then the URL is marked as a valid endpoint.

3.1.4 Domain Learning

Even though the method in Algorithm 3.1 is capable of locating linked data-related websites, the SPARQL endpoint pages may not show up in the search result pages with the common keywords like "sparql endpoint". In order to make a more complete search, we have created another step. After the preliminary search trials by using search query texts created for metacrawling (as explained above), a simple learning algorithm makes one more sophisticated search by using the previous results. The PLD (Pay Level Domain) names are extracted from previous 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 ef-

fective search is performed at each domain by specifying a search query for each domain separately.

3.2 Meta Analysis of Linked Data Sources Discovered

In order to create the meta analysis for a linked data source, the VoID[†] vocabulary includes a set of properties to define a linked data source in terms of statistics, numbers, names and descriptions. As a background for this vocabulary, the VoID vocabulary implementation project offers several SPARQL queries to collect statistical information about an existing data source.

By sending these queries to the SPARQL endpoints, the total number of the properties are collected. The results for these queries include information about the size and range of the linked data sources listed in that repository. As the final SPARQL endpoint URL collection refinement, these statistical results are used to filter out the same URLs from the SpEnD URL collection, i.e. endpoint URL's having the same number of triples and entities are considered the same linked data source duplicated and one of them is removed from the list.

4. Implementation

The SpEnD project contains a metacrawler and a linked data resource repository. The system's architecture is visualized in Fig. 3. The SpEnD system has three major steps: (a) metacrawling, (b) URL analyzer, and (c) domain learner. *Metacrawling Linked Data Endpoints* step (Sect. 3.1.2) utilizes HtmlUnit[††], which is a Web browser emulator for traversing over Bing, Yahoo, Google, and Yandex search engines. In order to analyze the URLs retrieved at the previous step, Jena Framework[†††] is used in the *URL Analyzer* step (Sect. 3.1.3), by sending SPARQL queries to candidate endpoints. All URLs found in crawling are then processed in the *Domain Learner* step (Sect. 3.1.4), by using Google Guava libraries[††††] for domain name analysis. All these three steps are discussed in detail below. SpEnD is built as a multi-threaded Java application. Above explained steps of the linked data discovery process are implemented as 3 parallel threads: (a) crawler, (b) endpoint extractor, and (c) statistical analyzer. In Fig. 4, the information flow between these threads is described as an activity diagram. A main thread is controlling user interactions and worker threads.

Worker threads are described as follows.

- Crawler: Performs search engine metacrawling jobs and feeds the endpoints extractor threads with the newly found links.
- Endpoint Extractor: Analyzes the candidate links generated by the crawler threads by sending SPARQL queries to every link.
- Statistical Analyzer: Analyzes the SPARQL endpoints by using statistical SPARQL queries listed in VoID descriptions. The queries are sent periodically as long as the thread runs.

Inside SpEnD, we have developed the "SPARQL Endpoint Crawler" (SpEC), in order to metacrawl search engines, analyze resulting URLs, and perform statistical analysis on the SPARQL endpoints discovered. Figure 5 shows the main

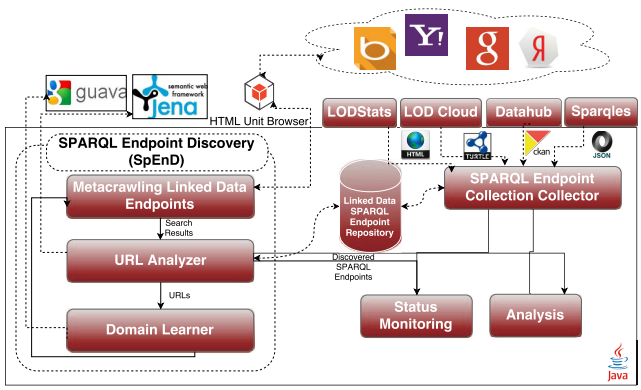

Fig. 3 SpEnD system diagram

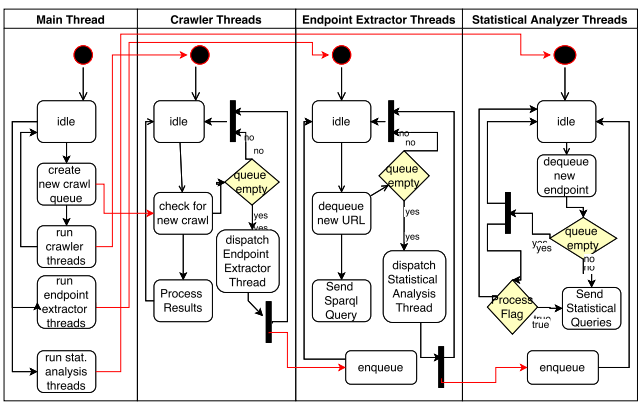

Fig. 4 Threads activity diagram

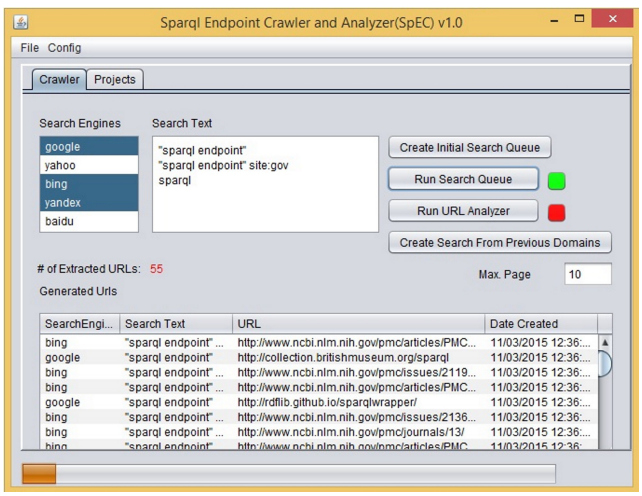

Fig. 5 Crawler management screen

[†]http://www.w3.org/TR/void/
[††]http://htmlunit.sourceforge.net/
[†††]https://jena.apache.org/
[††††]https://github.com/google/guava

screen of the SpEnD desktop application. The screen has two tabs, (a) Crawler, and (b) Projects. In the Crawler section, multi-threaded search engine crawling is performed by using the search queries entered by the user. In the Projects section, the statistical analysis on the discovered SPARQL endpoints is performed. This software is designed to run continuously, enabling an on-going crawling and analysis of all SPARQL endpoints. Currently, the results are updated daily, on the contrary by using classical Web crawling techniques it would take weeks or months, unless vast investments on computing infrastructure were made.

5. Evaluation

We have evaluated SpEnD by running it on four major search engines, namely Google, Yahoo, Bing, and Yandex. The linked data sources discovered are analyzed in terms of search engine results in Sect. 5.1. We then compare our results with another popular linked data metadata repository Datahub.io (Datahub) in terms of endpoint counts and availability in Sect. 5.2, service features in Sect. 5.3, interoperability features in Sect. 5.4, performance features in Sect. 5.5. Finally, the contents and vocabularies of endpoints are evaluated in Sect. 5.6.

5.1 Search Engine Results

Search engines were experimented by using 29 preliminary search queries in the first stage defined in Sect. 3.1.2. Afterwards, for the domain learning step defined in Sect. 3.1.4, totally 3341 domain specific search queries were sent. From the collection of these query results, a total of 295K URLs have been extracted. Out of these 295K URLs, 1,037 of them were marked as SPARQL endpoints based on the method described in Sect. 3.1.3. After the discovery process, these endpoints are analyzed based on their availability and meta information.

Out of these 1,037 endpoints, 211 of them are not available (not accessible but listed in the search engine results), so they are eliminated. The remaining 826 endpoints are then checked for duplications, whether the same dataset is listed more than once at different URL endpoints. To this end, the contents of the datasets are checked by evaluating their VoID statistics, that is the number of triples, entities, classes, and so on. If two endpoints return the same VoID statistics, they are considered the same dataset but somehow listed in seperate endpoints. And then, we eliminate all duplicate datasets from the collection. With this evaluation we eliminated a further 168 endpoints from the remaining datasets and ended up with only 658 unique dataset endpoints. Table 1 shows the search phrases we used that return the most results. For example, keywords "sparl query" return 207 unique SPARQL endpoints. We also utilize search directives such as "allinurl" for searching keywords in the URL of sites, or "allintitle" for searching in the title of sites. The rest of the 3370 search phrases and the results are available in the project Web site[†]. Table 2 shows the number of SPARQL endpoint links discovered by each search engine, which is also illustrated graphically in Fig. 6. Exclusive endpoint counts line in Table 2 shows the number of endpoints found only in each specific search engine. For example, 219 unique endpoints have been found only in Google. Figure 6 shows the overlaps of the results in detail for the four search engines. 107 endpoints have been found by all four search engines. The results also show that Google returns a significantly higher number of SPARQL endpoints (454 endpoints) than any other search engine. Although Google dominates the result set, the other search engines also contribute to enlarging the final SPARQL endpoint collection. Hence, we propose for such initiatives not to rely only on Google.

Table 1 Top 5 search phrases and the number of endpoints returned

Search Text	# of Endpoints
sparql query	207
"sparql endpoint"	179
inurl:sparql	150
allintitle: sparql query	144
allinurl: sparql data	105

Table 2 The number of unique endpoints for each search engine (Exclusive ones are found in that specific engine only)

	Google	Bing	Yahoo	Yandex	Unique Total
Exclusive	219	10	30	125	
Total	454	168	223	358	**658**

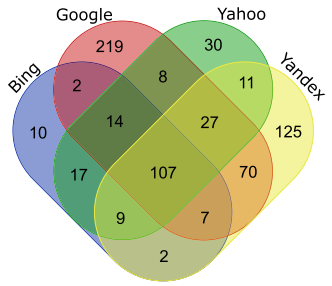

Fig. 6 Search engine SPARQL endpoint discovery results and overlaps

5.2 Datahub vs SpEnD in SPARQL Endpoint Counts and Availability

Here we compare the results of SpEnD with the popular and widely used Datahub repository results. The SPARQL endpoints listed in Datahub are collected by using the CKAN API provided by the website. Those endpoint records are compared both on their URLs and the corresponding PLDs. Finally, a status monitoring was applied in June 2016 and September 2016.

Table 3 compares the results retrieved from Datahub and those discovered by SpEnD. We also checked their availability, i.e. if the endpoint is responding or not. In order to make this availability check, we performed a one month

[†]http://wis.etu.edu.tr/spend/

Table 3 Comparison of SPARQL endpoints discovered

	High Avail. (>90%)	Low Avail. (<90%)	Offline	Total Online
Datahub	210	63	254	273
SpEnD	**537**	**121**	**211**	**658**

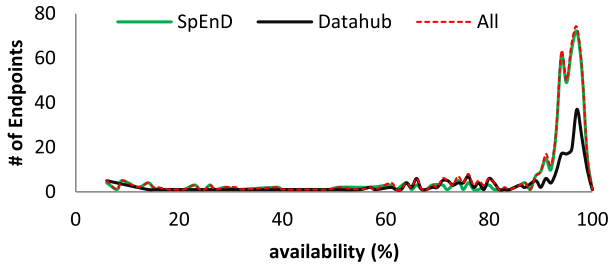

Fig. 7 Endpoint availabilities between June 2016–September 2016

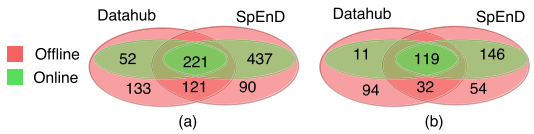

Fig. 8 Intersection set of (a) linked data SPARQL endpoint URLs, (b) active linked data SPARQL set domains (PLD)

status monitoring task by sending simple SPARQL queries to every endpoint regularly (scheduled to send a query every 10 minutes). In this table, the endpoints responding to more than 90% of the requests are counted as high available and less are counted as low available. Besides, the endpoints that did not respond to any of the queries are listed as offline. From June 2016 to September 2016, we performed multiple status monitoring tasks to record the average availability percentages of the endpoints. The availability percentages of these endpoints are illustrated in Fig. 7. Although, the overall percentages of highly available endpoints are higher in SpEnD dataset than Datahub dataset, there are several endpoints going offline temporarily or permanently. Figure 8-a visualizes the number of exclusive and common endpoints found by SpEnD and Datahub along with their online/offline status.

SpEnD dominates the results with 437 exclusive endpoints. There are some SPARQL endpoints not discovered by SpEnD. This is mainly due to the current unavailability of many of the endpoints in Datahub (obsolete records).

The green shaded areas in Fig. 8-a shows the active and available endpoint numbers only, at the time we tested the endpoints. In this inner set, 221 out of 273 endpoints listed in Datahub collection are found also by SpEnD (80.9% precision).

Although the number of online URLs shown in Fig. 8-a is mostly dominated by SpEnD data collection (437), some of these URLs are not domain-significant, meaning there are several SPARQL endpoints under the same domain names. Therefore, we also analyzed the SPARQL endpoint URLs based on their PLDs (Pay-Level-Domains). Figure 8-b shows the number of unique PLDs, 119 out of 130 domains listed in the other collections were discovered (91.5% precision). There are 11 PLDs, which are not discovered at all. After the discovery process, the search results for these 11 PLDs are manually reviewed. It has been observed that these PLDs are restricted for search engine crawling tasks and are not listed in any search engine.

Table 4 Number of PLDs and URLs

	#PLD		#Endpoint	
	Online	Offline	Online	Offline
SpEnD	**265**	86	**658**	211
Datahub	130	126	273	254

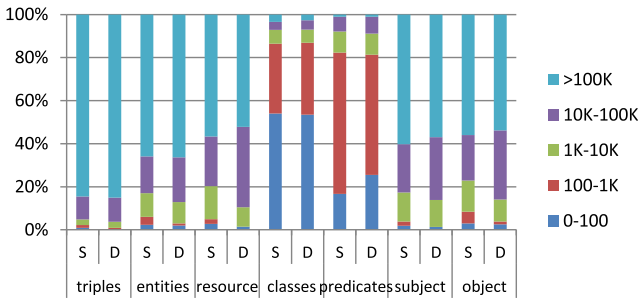

Fig. 9 Statistical comparative analysis S:SpEnD, D:Datahub

In Table 4, the total number of online/offline PLDs and endpoint URLs are listed. SpEnD project has the highest number of PLDs (265) and the highest number of endpoint URLs (658). Although the URL lists are not examined by any quality assessment method (yet), these are available for further research (listed in the project website). The 437 significant endpoint URLs and 146 significant domains discovered by SpEnD project are not qualified by using the quality measures [27]–[29]; however it is still valuable because they are not listed and included in any other collections.

5.2.1 Comparative Statistics

In this section, the statistical queries defined by the VoID vocabulary (explained in Sect. 2.1) were applied to each of the endpoints listed in the Datahub and SpEnD projects. By using these statistical features, we are able to identify the number of triples, entities, resources, classes, predicates, subjects, and objects used in SPARQL endpoints. In Fig. 9, the percentages of endpoints containing similar number of features are grouped and illustrated as a stacked column diagram.

According to Fig. 9, SpEnD endpoint repository contains similar percentages of all statistical results compared to Datahub repository. For every property, percentage of endpoints containing more than 100K results is more than the Datahub repository, which means SpEnD repository contains more high volume endpoints in percentage than the Datahub in terms of triples, resources, classes, subjects, and objects.

5.3 Service Features

In order to analyze service features of an endpoint, a simple

Table 5 Server names in HTTP get responses

	Datahub		SpEnD	
	PLD	Endpoint	PLD	Endpoint
Apache	32	108	51	285
Virtuoso	39	71	63	119
nginx	18	16	31	58
Jetty	3	0	3	3
AllegroServe	0	0	2	2

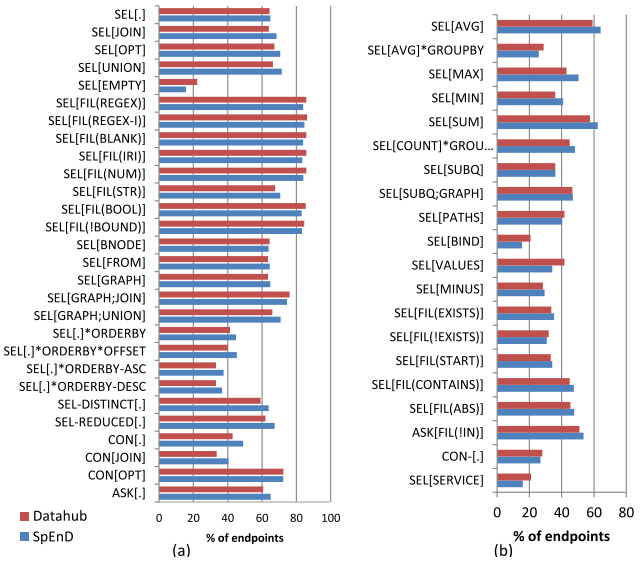

Fig. 10 SPARQL compliance results for (a) SPARQL 1.0 (b) SPARQL 1.1

GET request is sent to all endpoints in our repository by requesting data in RDF format. The `server` header of each response is extracted.

Overall, 527 endpoints returned `200 OK` response to the lookups. The responses were classified in Table 5 in terms of endpoint URLs and PLDs for Datahub and SpEnD. In these results, most of the endpoints returning RDF data use Apache, Virtuoso, and nginx servers.

5.4 Interoperability (SPARQL 1.0 and 1.1 Support)

For all endpoints listed in Datahub and SpEnD, a list of test SPARQL queries for testing SPARQL 1.0[†], and SPARQL 1.1[††] were sent to every SPARQL endpoint. The test queries were the same as the ones used in [30]. As it is illustrated in Fig. 10-a and Fig. 10-b, a comparison of Datahub and SpEnD in terms of the percentage of compliant endpoints for every single feature, is performed. Whereas SpEnD lists two times more endpoints which supports SPARQL 1.0 standards, the percentage of endpoints listed in each repository are similar. Overall, SPARQL 1.1 support is lower than SPARQL 1.0 support. Based on the percentages listed in Fig. 10-b, both repositories show similar distributions.

[†]http://www.w3.org/2001/sw/DataAccess/tests/r2 (l.a.: 2016-07-10)

[††]https://www.w3.org/2009/sparql/docs/tests/data-sparql11/ (l.a.: 2016-07-10)

Table 6 Result-size thresholds

	# of Endpoints		
# of Results	Union	Datahub	SpEnD
500	5	1	5
1000	11	6	7
1500	1	1	1
10000	59	32	56
20000	3	1	3
50000	8	4	8
100000	39	20	38
TOTAL:	126	65	118

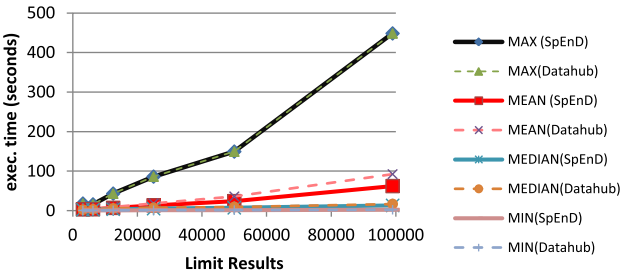

Fig. 11 Comparing different limit sizes

5.5 Performance

5.5.1 Result Streaming Performance

The result size thresholds for endpoints are examined by sending a `SELECT` query by limiting the result size to more than 100.000 records. 398 of the endpoints returned non-empty results. 199 endpoints returned more than 100.000 results, which means there is no result size threshold. 126 of the endpoints returned a "round number" as explained in [30]. Table 6 lists the number of endpoints with result-size thresholds comparatively by listing SpEnD, Datahub, and combined number of endpoints. In terms of the run-times for the endpoints returning more than the 99% of the limits specified, the execution times are illustrated in Fig. 11. Although max, min, and median execution times are the same for both SpEnD and Datahub endpoints, the average execution time for the SpEnD endpoints are lower than Datahub endpoints.

5.5.2 Atomic Lookup and Join Performance

In this experiment, the atomic level run times for simple `ASK` queries specified in [30]. The queries were prepared to request every combination of subject, predicate, and object properties. For instance, if a subject (s) is queried, the object and predicate conditions are written as <a> and <b> (which does not exist in any repository). By way of this, an endpoint needs to trace every single triple, or go through the relevant indexes, and this represents the maximum run time for such queries. In order to examine the caching effect on the query results, query submissions are performed twice for each property by labeling the first query as cold and the second query as warm. `ASK` queries were sent to all

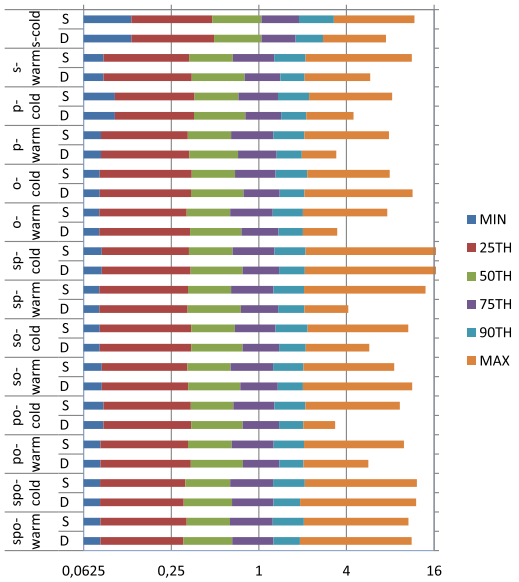

Fig. 12 Runtimes for ASK queries (%iles). S:SpEnD, D:Datahub

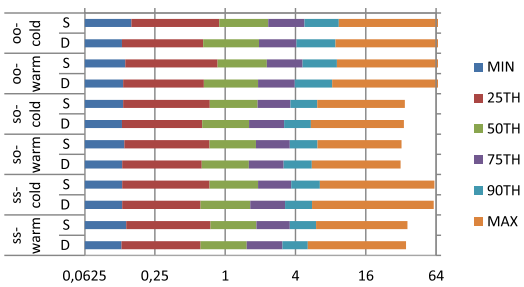

Fig. 13 Runtimes for JOIN queries (%iles). S:SpEnD, D:Datahub

endpoints listed in Datahub and SpEnD separately, and the results are illustrated in Fig. 12. 373 endpoints from SpEnD and 193 endpoints from Datahub are tested (all returning `false`). Colored sections in each bar represent the relevant percentages of endpoints returning results within the given time period (sec). We can say that around 60% of endpoints return response under 1 second for almost all queries, and the maximum response time is 16 seconds. Similar to the `ASK` queries explained before, the join performances are tested by sending join queries for subject-subject, subject-object, and object-object joins as specified in [30]. The results are illustrated for Datahub and SpEnD in Fig. 13. 40% to 50% of endpoints return response under 1 second for join queries. There is no significant difference between datasets in SpEnD and Datahub. so- and ss- type of join queries run in shorter times than oo- queries. Apparently some datasets are not indexed for the object side of triples efficiently.

5.6 Evaluation of SPARQL Endpoints

In this section, we present two types experimental analysis to understand the content of the endpoints. First, a categorical analysis of the current endpoints is presented in Sect. 5.6.1. Then a domain specific vocabulary analysis is performed on the IoT use-case in Sect. 5.6.2.

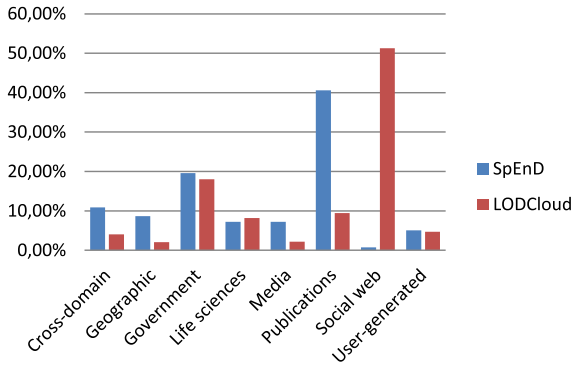

Fig. 14 Number of endpoints per category discovered by SpEnD

5.6.1 Content Evaluation

LOD Cloud Diagram[†] presents the current state of the Linked Open Data sources. In the diagram, all datasets are colored into 8 different categories, which are manually tagged by publishers in Datahub. We have compared the endpoints found in SpEnD against these tagged LOD datasets.

There are 196 datasets in LOD Cloud with a valid SPARQL endpoint, out of which 51 of them were offline when we checked. We have discovered all but only 5 of them in SpEnD. Figure 14 presents the distribution of endpoint categories for the endpoints found in SpEnD and LOD. The results illustrated in Fig. 14 shows that whereas the number of linked datasets categorized as Social Web is more than 50% for the LOD Cloud, only a few of them are served through SPARQL endpoints. On the contrary, the endpoints categorized as Publications are mostly served through SPARQL endpoints.

5.6.2 Vocabulary Evaluation

Semantic web vocabularies or ontologies are "used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms"[††] [31]. There are some commonly used and domain specific vocabularies in the linked data world such as Dublin Core, FOAF, SKOS, etc. Usage of these vocabularies in a dataset indicates there is data in those domains.

One of the emerging application domains where big data is created is Internet of Things (IoT) and there are already many commonly used vocabularies for the IoT domain. Some of these vocabularies are examined in [32]. We have extracted 37 IoT-related vocabularies from this work and checked their existence and therefore usage in the linked

[†]LOD Cloud, http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

[††]Vocabularies, http://www.w3.org/standards/semanticweb/ontology

Table 7 Number of ontologies identified in the IoT domain

Ontology	Union	SpEnD	Datahub
SSN	7	7	3
DUL	4	4	1
SmartBuilding	2	2	1
Km4City	2	2	1
DogOnt	2	2	1
OpenIoT	1	1	1
Fiemser	1	1	1
Fanfpai	1	1	1
Saref	1	1	0
Total	**21**	**21**	**10**

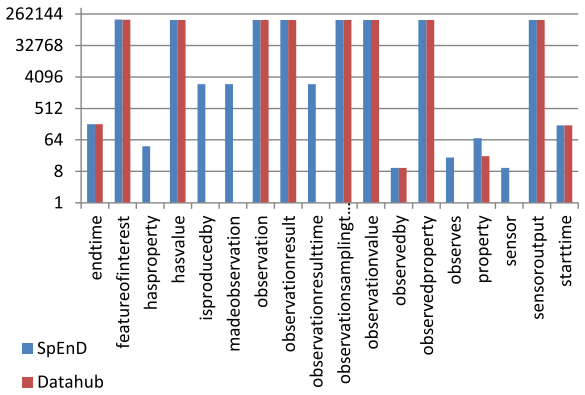

Fig. 15 Number of SSN properties available in SpEnD and Datahub

datasets found in SpEnD and Datahub. We have found that 9 of the vocabularies are used and the number of linked dataset using them are listed in Table 7. We have found that there are 11 more datasets than Datahub using these vocabularies. SSN (Semantic Sensor Networks) ontology or vocabulary is the top most used ontology in linked datasets. We have further checked the usage of SNN ontology in datasets by comparing the commonly used SSN properties (such as ssn:hasvalue, etc.).

Figure 15 shows the number of SSN properties used (number of triples) in 7 dataset endpoints. We found that datasets discovered by SpEnD only have more usage of the properties than Datahub. For example, important properties like ssn:sensor, ssn:observes, ssn:producedBy are not used in Datahub at all.

6. Conclusion and Future Work

Although linked data collections are mostly stored in centralized repositories, such as Datahub, it is clearly shown in this paper that this approach is not effective and dynamic enough for tracing and discovering new online SPARQL endpoints and identifying the existing ones going offline after sometime. Thus, in this study, we employed a metacrawling approach harnessing search engines, which are continuously queried and analyzed by our SpEC engine. Our results show that the majority of linked data SPARQL endpoints are available by Google search engine. Through this methodology, our SPARQL endpoint collection is growing incrementally and it is available to researchers for further analysis and research. At present, SPARQL endpoints collection through the SpEnD project has a better coverage of the Linked Open Data cloud both in terms of URLs and PLDs than any other project. The next steps in this research involve a semantic analysis [33] and a semantic ranking [34] of the collected SPARQL endpoints. This will help to have a better understanding about the content in the underlying linked data sources, and it will be possible to classify SPARQL endpoints according to their domain or context [35], [36].

References

[1] N. Shadbolt, T. Berners-Lee, and W. Hall, "The semantic web revisited," IEEE Intelligent Systems, vol.21, no.3, pp.96–101, 2006.
[2] I. Ermilov, J. Lehmann, M. Martin, and S. Auer, "LODStats: The Data Web Census Dataset," The Semantic Web – ISWC 2016, vol.9982, pp.38–46, Springer International Publishing, Cham, 2016.
[3] P. Vandenbussche, C. Aranda, A. Hogan, and J. Umbrich, "Monitoring the Status of SPARQL Endpoints," International Semantic Web Conference (Posters & Demos), pp.3–6, 2013.
[4] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, and S. Decker, "An empirical survey of Linked Data conformance," Web Semantics: Science, Services and Agents on the World Wide Web, vol.14, pp.14–44, July 2012.
[5] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far," International Journal on Semantic Web and Information Systems, vol.5, no.3, pp.1–22, 2009.
[6] 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," Proc. 3rd International Semantic Web User Interaction Workshop, 2006.
[7] K. Alexander and M. Hausenblas, "Describing linked datasets-on the design and usage of void, the'vocabulary of interlinked datasets," Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09), 2009.
[8] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American, vol.284, no.5, pp.34–43, May 2001.
[9] A. Batzios, C. Dimou, A.L. Symeonidis, and P.A. Mitkas, "BioCrawler: An intelligent crawler for the semantic web," Expert Systems with Applications, vol.35, no.1-2, pp.524–530, 2008.
[10] A.L. Symeonidis, E. Valtos, S. Seroglou, and P.A. Mitkas, "Biotope: An Integrated Framework for Simulating Distributed Multiagent Computational Systems," IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol.35, no.3, pp.420–432, 2005.
[11] A. Harth, A. Hogan, Y. Ding, F. Scharffe, and M. Hepp, "AuthorRank: Ranking Improvement for the Web," Int. Conf. Semantic Web and Web Services, pp.1–12, 2006.
[12] S.-Y. Yang, "OntoCrawler: A focused crawler with ontology-supported website models for information agents," Expert Systems with Applications, vol.37, no.7, pp.5381–5389, jul 2010.
[13] R. Isele, C. Bizer, and A. Harth, "LDSpider An open-source crawling framework for the Web of Linked Data," International Semantic Web Conference, pp.6–9, 2010.
[14] M. Kumar and R. Vig, "Learnable Focused Meta Crawling Through Web," Procedia Technology, vol.6, no.1994, pp.606–611, 2012.
[15] S. Lawrence and C. Giles, "United States Patent US6999959 B1-Meta search engine," 2006.
[16] A. Kenneth, J. Mcmahon, and C.A. Us, "United States Patent US7805432 B2 Meta Search Engine," 2012.
[17] D. Berton, B. Klock, E. Glover, and S. Kordik, "United States Patent US20040143644 A1-Meta-search engine architecture," 2004.
[18] A.E. Howe and D. Dreilinger, "SavvySearch: A Meta-Search Engine that Learns which Search Engines to Query," AI Magazine,

vol.18, no.2, pp.12–25, 1997.
[19] A. Gulli and A. Signorini, "Building an open source meta-search engine," Special interest tracks and posters of the 14th international conference on World Wide Web - WWW '05, pp.1004–1005, 2005.
[20] H. Wang, Q. Liu, T. Penin, L. Fu, L. Zhang, T. Tran, Y. Yu, and Y. Pan, "Semplore: A scalable IR approach to search the Web of Data," Web Semantics: Science, Services and Agents on the World Wide Web, vol.7, no.3, pp.177–188, Sept. 2009.
[21] Y. Lei, V. Uren, and E. Motta, "SemSearch: A Search Engine for the Semantic Web," Managing Knowledge in a World of Networks, vol.4248, pp.238–245, Springer Berlin Heidelberg, 2006.
[22] S. Campinas and D. Ceccarelli, "The Sindice-2011 dataset for entity-oriented search in the web of data," Proc. 1st International Workshop on Entity-Oriented Search (EOS), pp.26–32, 2011.
[23] T. Finin, J. Mayfield, A. Joshi, R.S. Cost, and C. Fink, "Information Retrieval and the Semantic Web," Proceedings of the 38th Annual Hawaii International Conference on System Sciences, pp.113a–113a, IEEE, 2005.
[24] A. Harth, A. Hogan, R. Delbru, S.O. Riain, and S. Decker, "SWSE: Answers Before Links !," Semantic Web Challenge, 2007.
[25] R.C. Miller and K. Bharat, "SPHINX: a framework for creating personal, site-specific Web crawlers," Computer Networks and ISDN Systems, vol.30, pp.119–130, 1998.
[26] E. Ferrara, P. De Meo, G. Fiumara, and R. Baumgartner, "Web data extraction, applications and techniques: A survey," Knowledge-Based Systems, vol.70, pp.301–323, 2014.
[27] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer, and J. Lehmann, "Crowdsourcing Linked Data Quality Assessment," International Semantic Web Conference, vol.8219, pp.260–276, 2013.
[28] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, and A. Zaveri, "Test-driven evaluation of linked data quality," Proceedings of t23rd international conference on World Wide Web, pp.747–758, 2014.
[29] P.N. Mendes, H. Mühleisen, and C. Bizer, "Sieve: Linked Data Quality Assessment and Fusion," Proceedings of the 2012 Joint EDBT/ICDT Workshops on - EDBT-ICDT '12, pp.116–123, 2012.
[30] C. Buil-Aranda, A. Hogan, J. Umbrich, and P.-Y. Vandenbussche, "SPARQL Web-Querying Infrastructure: Ready for Action?," The Semantic Web-ISWC 2013, vol.8219, pp.277–293, Springer Berlin Heidelberg, 2013.
[31] A. Gomez-Perez and O. Corcho, "Ontology languages for the semantic web," IEEE Intelligent Systems and Their Applications, vol.17, no.1, pp.54–60, 2002.
[32] A. Gyrard, G. Atemezing, C. Bonnet, K. Boudaoud, and M. Serrano, "Reusing and Unifying Background Knowledge for Internet of Things with LOV4IoT," 4th International Conference on Future Internet of Things and Cloud, pp.262–269, 2016.
[33] Z. Wang, J. Li, Y. Zhao, R. Setchi, and J. Tang, "A unified approach to matching semantic data on the web," Knowledge-Based Systems, vol.39, pp.173–184, 2013.
[34] J.M. García, M. Junghans, D. Ruiz, S. Agarwal, and A. Ruiz-Cortés, "Integrating semantic Web services ranking mechanisms using a common preference model," Knowledge-Based Systems, vol.49, pp.22–36, 2013.
[35] K. Oztoprak, "Subscriber Profiling for Connection Service Providers by Considering Individuals and Different Timeframes," IEICE Trans. Commun., vol.E99-B, no.6, pp.1353–1361, 2016.
[36] K. Oztoprak, "Profiling subscribers according to their internet usage characteristics and behaviors," Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015, pp.1492–1499, 2015.

Semih Yumusak received the B.S. degree in Computer Engineering from Koc University in 2005 and MBA degree from Istanbul Bilgi University in 2008. He is currently a PhD student in Computer Engineering at Selcuk University and a visiting researcher in The Insight Centre for Data Analytics, Galway, Ireland. His research interests include Semantic Web, Linked Data, Web Mining.

Erdogan Dogdu is a professor in the Computer Engineering Department at Cankaya University, Turkey. He received his BS degree in Computer Engineering and Science from Hacettepe University in 1987, MS and PhD degrees in Computer Science from Case Western Reserve University in 1992 and 1998 respectively. His recent research interests are in semantic web, web computing, big data analysis, and IoT.

Halife Kodaz graduated from Computer Engineering Department of Selcuk University with B.S. degree and M.S. degrees in 1999 and 2002, respectively. He received the Ph.D. degree in Electrical and Electronics Department from Selcuk University in 2008. He is an Associate Professor at the Computer Engineering Department at Selcuk University. His research interests are artificial intelligence and machine learning.

Andreas Kamilaris received the BSc degree in Computer Science from the University of Cyprus in 2007, the MSc degree in Distributed Systems from the ETH University of Zurich, Switzerland in 2009 and the PhD degree from University of Cyprus in 2013. His research interests include Internet/Web of Things, web computing, big data analysis and large-scale stream processing for smart cities and smart agriculture applications.

Pierre-Yves Vandenbussche received a BSc in Biology, a MEngg, a MSc in Computer science at Ecole Centrale Lille, a European MSc in Enterprise Interoperability and a PhD in 2011 in Information Technology from Paris VI University. Currently leading the Knowledge Engineering and Discovery research team in Fujitsu Ireland working with the Insight Data Center at Galway, his research interest concerns methods to improve data representation, knowledge systems management and knowledge graph mining.