A Service-oriented Search Framework for Full Text, Geospatial and Semantic Search

Andreas Both
R&D, Unister GmbH
Leipzig (Germany)
andreas.both@unister.de

Axel-Cyrille Ngonga Ngomo
Universität Leipzig, IFI/AKSW
Leipzig (Germany)
ngonga@informatik.uni-leipzig.de

Ricardo Usbeck
Universität Leipzig, IFI/AKSW
Unister GmbH, Leipzig
usbeck@informatik.uni-leipzig.de

Denis Lukovnikov
Universität Leipzig, IFI/AKSW
Leipzig (Germany)
lukovnikov@informatik.uni-leipzig.de

Christiane Lemke
R&D, Unister GmbH
Leipzig (Germany)
christiane.lemke@unister.de

Maximilian Speicher
TU Chemnitz, VSR
Unister GmbH, Leipzig
speim@hrz.tu-chemnitz.de

ABSTRACT

Over the last decade, a growing importance of search engines could be observed. An increasing amount of knowledge is exposed and connected within the Linked Open Data Cloud, which raises users' expectations to be able to search for any information that is directly or indirectly contained. However, diverse data types require tailored search functionalities—such as semantic, geospatial and full text search.

Hence, using only one data management system will not provide the required functionality at the expected level. In this paper, we will describe search services that provide specific search functionality via a generalized interface inspired by RDF. In addition, we introduce an application layer on top of these services that enables to query them in a unified way. This allows for the implementation of a distributed search that leverages the identification of the optimal search service for each query and subquery. This is achieved by connecting powerful tools like *Openlink Virtuoso*, *ElasticSearch* and *PostGIS* within a single framework.

General Terms

Distributed Search, Semantic Web, Information Retrieval

1. INTRODUCTION

Over the last two decades, the content of the World Wide Web has grown to an enormous collection of webpages. In addition, users have shifted their preference from desktop to Web applications. This is primarily driven by growing technical capabilities of Web browsers as well as the demand for exploiting the knowledge available on the Web. Particularly, large Web application providers—including Google, Bing and Yahoo!—promise that any kind of information will be made available through simple search queries.

In recent years, a very large number of datasets have been published based on Semantic Web standards. More than 61 trillion triples[1] are available through the *Linked Open Data* (LOD) cloud by now. Instead of publishing text documents containing unstructured information, the new paradigms demand information which are structured by standards like the *Resource Description Framework* (RDF)[2]. RDF can be used for for publishing logical properties like `typeOf` relations (e.g., Germany is of the type `country`), standard type properties (e.g., the number of people living on a square kilometer within Germany: `229`) or complex data types such as geospatial coordinates (e.g., `POINT(13.3833, 52.5167)` in the case of Germany). Since this data is published in a standardized format (i.e., RDF), it can be read and processed by machines. This provides opportunities for creating novel industrial applications.

The trend just described leads to a situation in which a vast amount of unstructured information is available on the Web next to a large amount of structured data accessible for automatic processing by machines. While the latter data representation enables direct access of annotated knowledge and a derivation of insights, the knowledge of the textual representations (mostly HTML documents) is not directly accessible in an easy way. Thus, information retrieval methods are needed in a preprocessing step to compute semantic information which can be annotated within the (HTML) document.

In the course of the current developments, users will tend to reject search solutions based on the knowledge originating from one data representation only. On the one hand, Google's Web search[3] demonstrates the advantages of sear-

[1]`http://stats.lod2.eu/`, retrieved June 16, 2014.
[2]`http://www.w3.org/TR/rdf11-concepts/`
[3]`http://www.google.com`

ching huge collections of textual documents. On the other hand, applications like Wolfram Alpha[4] and Apple's Siri[5] present the advantages of knowledge-driven search functionalities.

Thus, a search infrastructure is required, which is capable of integrating the functionalities of textual search with search methods working on top of annotated semantics. User demands in the context of e-commerce are particularly driven by queries for products (i.e., the type of the subject of interest), such as "hotel", "flight" or "winter holiday". Additionally, properties can express the requirements posed on the subject of interest. Three different types have to be considered in this case:

- logical properties, e.g., "has wifi", "suitable for vegetarians";
- geospatial properties, e.g., "north of London", "close to a beach";
- properties driven by textual information, e.g., "pool for children"[6] or a part of the name of the searched entity (like "Kempinski").

All three types of queries are well supported by specific instances of data stores:

- searches for logical properties are implemented by triple stores, e.g., the Openlink Virtuoso Server [6];
- searches for geospatial properties are supported by database management systems with extensions for geographic information system (GIS), e.g., PostGIS [23], an open source software program that adds support for geographic objects to the PostgreSQL object-relational database [20];
- searches within text documents are supported by index-based data stores, e.g., the scalable search solution ElasticSearch [16] based on Apache Lucene [11].

In this paper, we will present an architectural layer on top of these well-known search solutions. In particular, we will take care of preserving their scalability aspects, like the search on large sets of text documents. A data representation close to RDF will be used within our architecture. However, there is no need for using a particular data store for semantic data (i.e., mostly a triple store) since our approach is agnostic with respect to the backends used. Our main contribution is an engineering approach for creating a scalable search solution capable of conducting semantic search with geospatial aspects as well as information retrieval from text documents. The benefits of the different search systems are combined and integrated in order to better deliver on the expectations of the users.

[4]`http://www.wolframalpha.com/`

[5]`https://www.apple.com/de/ios/siri/`

[6]We assume that such properties are not directly modeled in the underlying structured knowledge base.

The rest of the paper is organized as follows. Section 2 addresses related work while Section 3 describes general requirements for our approach as well as its architecture. Subsequently, we present our approach to search query representation in Section 4 and introduce our approach to interpreting search queries in Section 5. Section 6 describes our federated search architecture, before giving concluding remarks in Section 7.

2. RELATED WORK

Our approach is related to the research area of search over Linked Data and thus to keyword search and question answering (QA) over Linked Data. Several systems have been developed for the latter task. One of the first systems was AquaLog [19], an ontology-driven QA system for the Semantic Web. Aqualog uses linguistic analysis to transform the input query to a set of query-triples. Then, these query triples are interpreted using lexical resources and the given ontology. The interpreted query-triples are sent to an inference engine to find the answer. One major drawback of AquaLog is that it is limited to one ontology at a time. To address this and other drawbacks of AquaLog, PowerAqua [18] was developed. PowerAqua follows an approach similar to that of AquaLog, reusing AquaLog's linguistic analysis. However, PowerAqua performs more advanced query-triple interpretation using different ontologies simultaneously.

Treo [7] is a method for querying Linked Data that relies on spreading activation. First, pivot entities in the query are identified. Then, from the dependency structure of the input sentence, Treo constructs a Partially Ordered Dependency Structure (PODS). The PODS is used to resolve the query in the spreading activation search step where semantic relatedness scores are used to rank candidates and subsequently spread activation. Pythia [29] is a more recent QA system. It uses lexica that define mappings between a syntactic and a semantic representation of an expression. With these lexica, Pythia also introduces a distinction between ontology-dependent and ontology-independent lexica. Whereas the former depends on the verbalizations of entities from some ontology, the latter describe ontology-independent expressions (determiners, question words,...). The sentence is parsed using such lexica and the resulting semantic representation is translated to a formal query. Building upon Pythia's dichotomy between ontology-dependent and -independent lexical entries, TBSL [28] presents a template-based QA approach. This approach consists of 3 steps. First, SPARQL query templates are generated by using domain-specific and generic language resources. The template slots are filled by using a combination of resource lookup and natural-language patterns extracted using the BOA framework [9]. The resulting SPARQL queries are finally scored and the best query (i.e., the highest ranking query that returns a non-empty result set) is selected and returned. More recently, several novel systems have participated in the QALD-3 challenge on QA over Linked Data [3]. CASIA [12] (the currently best-performing system on the QALD-3 benchmark dataset) relies on a three-step approach resembling AquaLog's architecture. During the first step, the question type is determined and text triples are constructed from the dependency parse tree of the question sentence. In the second step, RDF resources which match phrases from the text triples are detected. CASIA uses relation extraction patterns from PATTY [21] to map text fragments to properties and classes. In the final step, a SPARQL query is generated based on the question type and the RDF resources detected in the input question. CASIA achieves an F-score of 0.36 on the QALD-3 benchmark.

Approaches on keyword-based querying of the Web of Data include SINA [26] and the work of Tran et al. [27]. SINA [26] aims at answering a keyword question using diverse datasets. First, simultaneous disambiguation and segmentation is performed using Hidden Markov Models (HMM) and the Hyperlink-Induced Topic Search (HITS) algorithm. The found resources are used to construct an Incomplete Query Graph (IQG) consisting of disjoint sub-graphs. To build the federated SPARQL query that retrieves the results, the IQG's are connected using a Minimum Spanning Tree approach inspired by Prim's algorithm. The work of Tran et al. [27] tackles the problem of keyword search over RDF data. More specifically, the work of Tran et al. is concerned with mapping keywords to a list of ranked conjunctive queries, with a special focus on efficient inference of implied connections. To accomplish this, a top-k algorithm is proposed that computes the best query interpretations of the keyword query using bidirectional graph exploration. The interpretations are then scored and mapped to conjunctive queries. The performance of the proposed top-k approach is evaluated on DBLP.

Generic frameworks for searching over RDF data have been suggested in the past. For example, the OKBQA framework[7] presents a modular architecture for search over structured and unstructured sources. This architecture is yet not fully instantiated and it is thus difficult to compare with our approach.

Usbeck [30] presents a generic architecture for hybrid search using a holistic framework comprising information extraction methods for unstructured [31] and semi-structured data sources.Afterwards, the framework combines the underlying heterogeneous data stores to answer keyword and natural language queries via transforming each query into generic SPARQL queries returning only the highest ranked results.

In contrast to the state of the art, we propose a generic framework which integrates state of the art approaches for search over both Linked Data, unstructured data and arbitrary APIs.

In [22] a federated SPARQL search engine–FedSearch– is presented, which presents a hybrid combination of SPARQL and full-text search tackling data heterogeneity and lacking statistical data. Since SPARQL lacks full-text search support the authors propose a triple-store-independent way of querying different RDF stores such as OWLIM, Virtuoso and LuceneSail. Their vendor independent approach of keyword query search pattern is evaluated next to several optimizations against two benchmarks showing superior performs against other state-of-the-art systems.

Meta-search engines (e.g., [17, 10]) are a different approach for taking advantage of the power of different search services. However, meta-search engines do not re-use intermediate results to refine parts of the search query, at the most they improve the representation by re-ranking or clustering the summarized search results (e.g., [2]).

[7]`http://www.okbqa.org`

Service-oriented architecture (SOA) [8, 24] can be seen as an architectural pattern providing the needed tools for organizing the communication between distributed (software) components [1]. The SOA manifesto[8] prioritizes intrinsic interoperability over custom integration, shared services over specific-purpose implementations and flexibility over optimization. Its main purpose is providing an infrastructure for connecting loosely coupled components by defining processes using dynamic component discovery and registration [5].

Thus, the goals of SOAs are close to the ones of Linked Data. In contrast, SOA is interface-driven not data-driven like Linked Data. In particular, it is not connected to SPARQL or RDF. However, the concepts can be combined aiming for the best of both worlds, e.g., as [34] has shown for the e-learning domain.

3. HIGH-LEVEL ARCHITECTURE AND GENERAL REQUIREMENTS

The concept of a service-oriented search framework poses several specific requirements that have to be met to realize a corresponding architecture. They will be described in the following. First of all, the framework has to ensure the integrability of the various intended search functionalities, which is derived from the properties of (extensible) service-oriented architectures:

Requirement 1. *An interface for search services has to be defined. The definition has to comprise a summarization of all needed attributes for controlling the execution of (sub)queries.*

We focus on a generalized approach, especially w.r.t. the communication across the diverse search services. Thus, we demand:

Requirement 2. *The communication with the search services has to be stateless and transparent.*

Derived from the required stateless and transparent communication, each search service has to encode all information about itself via its interface ,i.e., in the message during the communication[9]. Hence, the information about each search service has be retrieved by querying the implemented interface methods. However, to provide a structured search query, an analysis of the textual user input is needed, which we consider to be beyond the scope of this paper. We will assume the search query is already available in a structured representation (e.g., by having used approaches like [4, 19, 28]).

[8]http://soa-manifesto.org/

[9]This requirement is comparable to REStful communication [25] via the stateless HTTP protocol `http://tools.ietf.org/html/rfc7231`.

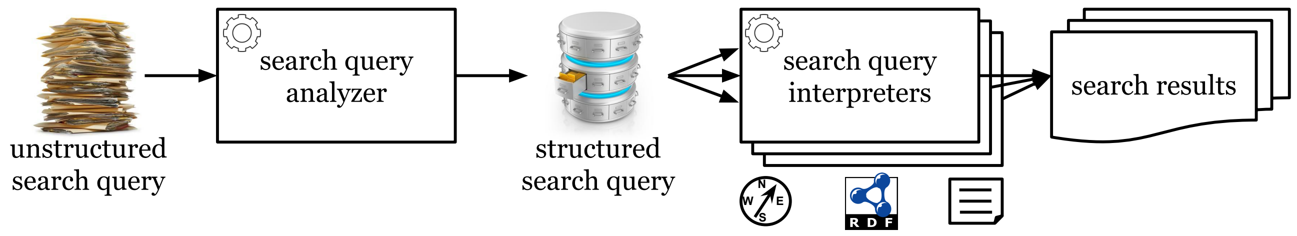

Figure 1: High-level architecture of our distributed search framework.

Requirement 3. *The search query requires a structured representation which encapsulates information about all relevant parts of the search query.*

Hence, the high-level infrastructure of a federated search requires a two-step process:

1. search query analyzer
   - input: unstructured search query
   - output: structured search query
2. search query interpreter
   - input: structured search query
   - output: ranked search results

A graphical overview of the workflow is shown in Figure 1. To achieve the intended service-oriented architecture, it is crucial to hide the implementation details of the contained search services [5]. As sketched in Figure 1, the structured search query is the main carrier of information. Hence, no knowledge about the behavior of a particular search service may be contained in a structured search query, except in the case that it was computed by a search service itself. This is because the structured search query has to be independent from the particular implementation of any search service to ensure their exchangeability within the service-oriented architecture. The following requirement is derived:

Requirement 4. *The structured search query representation has to be independent from the implementation of any particular search service.*

In the following, we will describe the data structure required for the representation of an analyzed search query.

4. SEARCH QUERY REPRESENTATION

Understanding what the user is looking for is the starting point of every information retrieval process. Yet, in the following section, the focus is not on the analysis of unstructured search queries. Rather, search query analysis is considered a black box, i.e., we focus on the representation of the structured search query after analysis.

For our purpose, we build on a search query representation that is close to RDF:

Definition 1 (RDF triples). *Assume there are pairwise disjoint infinite sets I, B, and L representing IRIs*[10]*, blank nodes, and RDF literals, respectively.*

A triple $(v_1, v_2, v_3) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF triple. We call v_1 the subject, v_2 the predicate and v_3 the object. We denote the union $I \cup B \cup L$ by T called RDF terms.

However, we need a definition of the target resource. Therefore, we allow a specific IRI to indicate the target elements, i.e., our system performs a pure resource search. For the sake of simplicity, we denote it as follows:

Definition 2 (Searched resource). *The place holder for the searched resource is denoted as `urn:placeholder`.*

Obviously, this mechanism can be used for referencing different variables by adding any ID (e.g., `urn:placeholder2`).

For example, it is possible to express an actual search query, e.g., `family-friendly hotel in Leipzig` in the following way: Potential target hotels (`urn:id:hotel`) are restricted to those being located in the city center of Leipzig (expressed using the relation `urn:rel:cityCenter`). Furthermore, a target hotel has to be marked (`urn:rel:hasFeature`) as family-friendly (`urn:id:familyFriendly`) and the description of the hotel has to contain the word family-friendly. The corresponding search query representation is shown in Figure 2.

4.1 Search Query Tagging

Given the intention of using different search queries within a federated search architecture connecting diverse search functionalities, there is a need for separating a given search query into smaller parts. These parts should be solvable by one or more search services providing the best match to the required search functionality.

In accordance with Requirement 4, it is not allowed to encode the control of the execution directly within the representation of a (sub-)query. Instead, the triples (i.e., subqueries) contained in the structured search query are assigned to *every* search service for annotation. That is, a subquery is annotated with (a) whether the given service is eligible for interpreting the sub-query, (b) the estimated costs

[10] *Internationalized Resource Identifier*, i.e., a URI that may contain any Unicode character (cf. `http://tools.ietf.org/html/rfc3987`).

t_1	`urn:placeholder`	`owl:typeof`	`urn:id:hotel`
t_2	`urn:placeholder`	`urn:rel:citycenter`	`urn:id:Leipzig`
t_3	`urn:placeholder`	`urn:rel:description`	`urn:placeholder2`
t_4	`urn:placeholder2`	`urn:rel:sublabel`	"family-friendly"

Figure 2: Example of a structured search query `family-friendly hotel in Leipzig` according to our RDF-like representation.

t_1	`urn:placeholder`	`owl:typeof`	`urn:id:hotel`	(`urn:service:triplestore`,250,500)
t_2	`urn:placeholder`	`urn:rel:citycenter`	`urn:id:Leipzig`	(`urn:service:gis`,100,400)
t_3	`urn:placeholder`	`urn:rel:description`	`urn:placeholder2`	(`urn:service:triplestore`,4000,1000)
t_3	`urn:placeholder`	`urn:rel:description`	`urn:placeholder2`	(`urn:service:fulltext`,300,200)
t_4	`urn:placeholder2`	`urn:rel:sublabel`	"family-friendly"	(`urn:service:fulltext`,300,200)

Figure 3: Example of an annotated search query `family-friendly hotel in Leipzig`.

for processing the sub-query and (c) the estimated number of results. In particular, it is possible that a triple is tagged by several search services. The annotation processes for a given sub-query are started for each search service concurrently.

After this step, the triples contained in a structured search query are annotated. Thus, the structured search query is now defined as follows:

Definition 3. *A set A of pairs (t, a) where t is a triple of a search query Q and a is the annotation of a search service, is called structured search query with annotations.*

Note: It is possible that not all triples of Q are annotated within A. However, without loss of generality, it is assumed that all triples are tagged at least once.

To ensure performant computation of a query plan, an annotation is defined as follows:

Definition 4. *A query annotation a is a triple (i, r, e), where i is the IRI of the service, r is the estimated number of results and e is the estimated execution time (in milliseconds).*

This definition satisfies Requirement 3. Imagine a user wants to pose the following query to our framework:

Example 1. *family-friendly hotel in Leipzig*

The annotated example query is shown in Figure 3. In this example, `urn:service:fulltext` is an IRI of a service providing a full text search, `urn:service:triplestore` references a triple store and `urn:service:gis` points to a search service encapsulating a GIS server.

5. SEARCH QUERY INTERPRETATION

Given a structured and annotated search query, a reasonable query plan for the eligible search services has to be computed. This can be done by scheduling algorithms with different optimization strategies. In this paper, we will focus on the general requirements only.

Requirement 5. *All triples of a structured and annotated search query have to be interpreted by at least one search service.*

Note: As mentioned before, it is explicitly possible to have a triple interpreted by more than one search service.

Hence, a query plan can be computed by creating a topological order [14] of the graph defined by the triples (i.e., the triple S P O is interpreted as $S \xrightarrow{P} O$) of the structured search query where the result node is the root node of the graph. A straight-forward implementation is shown in Figure 4. There is numerous existing work dedicated to scheduler implementations [13, 15, 32]. A schedule graph is shown in Figure 5 with respect to our running example.

6. FEDERATED SEARCH ARCHITECTURE

In this section, we present our federated search infrastructure. Particularly, it contains a service layer to which the diverse search services can be connected.

6.1 Search Service Interface

For integration into the overall system, all search services have to implement a specific interface that facilitates exchangeability. As a consequence, by implementing this given interface, services can be loosely coupled to the service layer. The search service interface is defined as follows:

```
interface SearchService {
    /* annotation
     * in: search query Q
     * out: annotated search query A
     */
    A annotateQuery(Q);

    /* query execution
     * in: annotated search query A
     * out: search results R
     */
    R executeQuery(A);
}
```

```
/* compute topological order of the triples of the structured search query
 * S P O is interpreted as directed edge S -> O with label P
 * S,O are interpreted as nodes */
A scheduleSubQueries(A){
  List<node> resultList  // list will contain the ordered results
  Set<node> startSet= findNodesWithoutIncEdges(A) // set with nodes without incoming edges
  int orderTag = 0;
  while(!startSet.isEmpty()){
      n = startSet.pop() // removes a node
      n.orderTag = orderTag++; // assign schedule order
      L.queue(n) // adds the node to the end of the list L
      for(node m where (n,m) in A){
          remove (n,m) from A
          if( no p exists with (p, m)){ // no other incoming edges for m
              startSet.push(m)
          }
      }
  }
  if(hasEdges(A))
      return error // graph has at least one cycle
  else
      return L // a topologically sorted order
}
```

Figure 4: Exemplary scheduler implementation

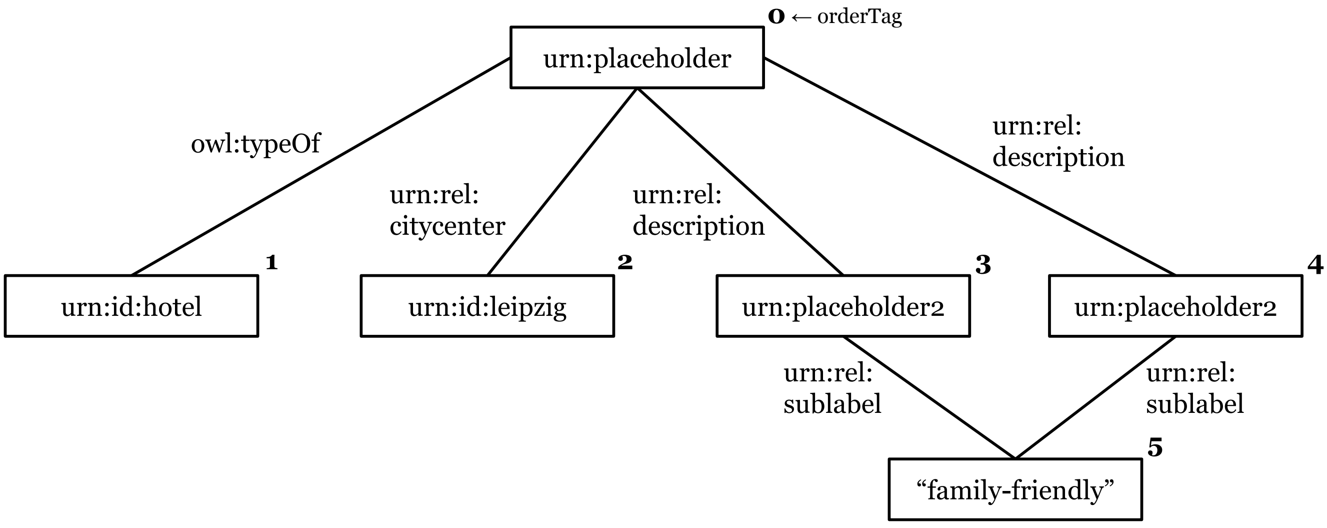

Figure 5: Scheduled query plan after topological sorting.

It is common to call `annotateQuery` with a complete structured query `Q` to get a complete annotated query `A` of all triples contained in `Q`. Yet, the `executeQuery` method of an implementing search service is usually called passing only the sub-query the service is eligible for (e.g., passing only the property `in Leipzig` to the geospatial search service). Eligibility of a service is decided by the scheduler.

6.2 Search Provider

The search provider is an interface for integrated search services and defined as follows:

```
interface SearchProvider {
    /* query execution
     * in: search query Q
     * out: search results R
     */
    R executeQuery(Q);

    /* search service registration
     * in: IRI of the search service
     */
    void register(IRI);

    /* deregister a search service
     * in: IRI of the search service
     */
    void deregister(IRI);
}
```

Hence, the search provider provides a method (`executeQuery`) for been accessed from a client while it also works as directory server (similar to a UDDI server [33]). The directory functionality is achieved by the `register` and `deregister` methods allowing search services to connect and disconnect from the search provider.

7. CONCLUSIONS

We presented an architecture for a distributed search following the principles of service-oriented architectures. For this, we introduced a descriptive data representation, i.e., the data representation does not contain implementation details of the underlying search services. Our implementation of such a structured search query is based on RDF. After the tagging process, the annotated structured search query can be expressed using our RDF-like representation only.

Our main contribution is the proposed architecture's capability of integrating and disintegrating (RDF unaware) search services dynamically. This property is achieved by the descriptive message format as well as the generalized service interface, which allows for a stateless and transparent communication with the integrated search services. Hence, completely novel use cases are possible, in which search service integration happens on-the-fly to facilitate more dynamic environments. Since the architecture is agnostic to the functionality of the search services, it is possible to integrate search services having not only (RDF-driven) semantic search functionalities but also other specialized search capabilities. For example, search services could comprise any specialized search functionality like ElasticSearch (providing high-performance federated large-scale full text search), or Web APIs encapsulating search functions with special semantics like SoundCloud[11] or Github[12]. Our approach is superior as it does not burden these search services with the interpretation of the complete RDF (or SPARQL) standard. Moreover, the architecture is scalable by design in the sense of the capability of integrating scalable and distributable search services. Finally, our `SearchProvider` provides a generalized interface being capable of receiving search queries via SPARQL (or any other representation, such as SQL or SeRQL). Therefore, a consolidated view on the dynamically summarized search services can be provided. This allows for a transparent integration as a SPARQL endpoint, or using any other data access standard.

In the future, we will evaluate the capabilities of our architecture of providing rapid access to distributed data resources by comparing an implementation of the architecture to monolithic approaches. As possible data resources we consider datasets of the Linked Open Data Cloud as well as unstructured data provided by the Document Web in general or private data sets with restricted accessibility (by account or on the network level). Hence, providing personalized search functionalities seems to be achievable.

Finally, the implementations are yet to be evaluated regarding performance and (re-)ranking issues. In the context of providing a distributed search, these are the main challenges considering the user's demands w.r.t. usability.

Acknowledgments We thank Didier Cherix and Lars Wesemann for their valuable support. This work has been partly supported by the ESF and the Free State of Saxony and by grants from the European Union's 7th Framework Programme provided for the project GeoKnow (GA no. 318159).

8. REFERENCES

[1] M. Bell. *SOA modeling patterns for service oriented discovery and analysis*. John Wiley & Sons, 2009.

[2] C. Carpineto and G. Romano. Optimal meta search results clustering. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 170–177, New York, NY, USA, 2010. ACM.

[3] P. Cimiano, V. Lopez, C. Unger, E. Cabrio, A.-C. N. Ngomo, and S. Walter. Multilingual Question Answering over Linked Data (QALD-3): Lab Overview. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 321–332. Springer, 2013.

[4] R. De Virgilio, A. Maccioni, and R. Torlone. A similarity measure for approximate querying over RDF data. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, pages 205–213, New York, NY, USA, 2013. ACM.

[5] T. Erl. *SOA: principles of service design*, volume 1. Prentice Hall Upper Saddle River, 2008.

[11]https://developers.soundcloud.com/

[12]https://developer.github.com/

[6] O. Erling and I. Mikhailov. RDF support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media*, pages 7–24. Springer, 2009.

[7] A. Freitas, J. G. Oliveira, S. O'Riain, E. Curry, and J. C. P. Da Silva. Querying linked data using semantic relatedness: a vocabulary independent approach. In *Natural Language Processing and Information Systems*, pages 40–51. Springer, 2011.

[8] Gartner Inc. Research note SPA-401-068, 12 April 1996, "'service oriented' architectures, part 1" und SSA research note SPA-401-069, "'service oriented' architectures, part 2", 1996.

[9] D. Gerber and A.-C. N. Ngomo. Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction ISWC*, 2011.

[10] A. Gulli and A. Signorini. Building an open source meta-search engine. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1004–1005. ACM, 2005.

[11] E. Hatcher, O. Gospodnetic, and M. McCandless. Lucene in action, 2004.

[12] S. He, S. Liu, Y. Chen, G. Zhou, K. Liu, and J. Zhao. CASIA@ QALD-3: A question answering system over linked data.

[13] L. Ismail and L. Khan. Implementation and performance evaluation of a scheduling algorithm for divisible load parallel applications in a cloud computing environment. *Software: Practice and Experience*, 2014.

[14] D. E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition.* Addison-Wesley, 1981.

[15] S. Kuankid, A. Aurasopon, and W. Sa-Ngiamvibool. Effective scheduling algorithm and scheduler implementation for use with time-triggered co-operative architecture. *Elektronika ir Elektrotechnika*, 20(6):122–127, 2014.

[16] R. Kuc and M. Rogozinski. *Elasticsearch Server.* Packt Publishing Ltd, 2013.

[17] U. Liebel, B. Kindler, and R. Pepperkok. âĂŸharvesterâĂŹ: a fast meta search engine of human protein resources. *Bioinformatics*, 20(12):1962–1963, 2004.

[18] V. Lopez, E. Motta, and V. Uren. Poweraqua: Fishing the semantic web. In *The Semantic Web: research and applications*, pages 393–410. Springer, 2006.

[19] V. Lopez, V. Uren, E. Motta, and M. Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Web Semant.*, 5(2):72–105, June 2007.

[20] B. Momjian. *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley New York, 2001.

[21] N. Nakashole, G. Weikum, and F. Suchanek. PATTY: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics, 2012.

[22] A. Nikolov, A. Schwarte, and C. Hütter. FedSearch: Efficiently combining structured queries and full-text search in a SPARQL federation. In *The Semantic Web–ISWC 2013*, pages 427–443. Springer, 2013.

[23] R. Obe and L. Hsu. *PostGIS in action.* Manning Publications Co., 2011.

[24] Organization for the Advancement of Structured Information Standards. Reference model for service oriented architecture 1.0,committee specification 1, 2006.

[25] L. Richardson and S. Ruby. *RESTful web services.* " O'Reilly Media, Inc.", 2008.

[26] S. Shekarpour, A.-C. Ngonga Ngomo, and S. Auer. Question answering on interlinked data. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1145–1156. International World Wide Web Conferences Steering Committee, 2013.

[27] T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 405–416. IEEE, 2009.

[28] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 639–648, New York, NY, USA, 2012. ACM.

[29] C. Unger and P. Cimiano. Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In *Natural Language Processing and Information Systems*, pages 153–160. Springer, 2011.

[30] R. Usbeck. Combining linked data and statistical information retrieval - next generation information systems. In V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, and A. Tordai, editors, *ESWC*, volume 8465 of *Lecture Notes in Computer Science*, pages 845–854. Springer, 2014.

[31] R. Usbeck, A.-C. Ngonga Ngomo, S. Auer, D. Gerber, and A. Both. AGDISTIS - Agnostic Disambiguation of Named Entities Using Linked Open Data. In *International Semantic Web Conference.* 2014.

[32] J. Vilaplana, F. Solsona, J. Mateo, and I. Teixido. SLA-aware load balancing in a web-based cloud system over openstack. In *Service-Oriented Computing–ICSOC 2013 Workshops*, pages 281–293. Springer, 2014.

[33] A. E. Walsh, editor. *Uddi, Soap, and Wsdl: The Web Services Specification Reference Book.* Prentice Hall Professional Technical Reference, 2002.

[34] H. Q. Yu, S. Dietze, N. Li, C. Pedrinaci, D. Taibi, N. Dovrolls, T. Stefanut, E. Kaldoudi, and J. Domingue. A linked data-driven & service-oriented architecture for sharing educational resources. 2011.