Semantic Interpretation of User Queries for Question Answering on Interlinked Data

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Saeedeh Shekarpour
aus
Fasa, Iran

Bonn 2014

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Sören Auer
2. Gutachterin:	Prof. Dr. Stefan Wrobel
Tag der Promotion:	7th January 2015
Erscheinungsjahr:	2015

Acknowledgements

I would like to express my gratitude to the many people who supported me through this thesis. First special thanks to my supervisor Prof. Dr. Sören Auer who always provided support throughout my research, I have been very lucky to have a supervisor who cared so much about my work, and who responded to my questions so quickly. His positive outlook and confidence in my research inspired me and gave me confidence.

I am grateful to the colleagues and the friends from AKSW research group especially those who made a direct contribution to the work like Prof. Dr. Sören Auer, Dr. Axel C. Ngonga Ngoma and Dr. Jens Lehman who discussed things over, read, wrote, offered comments, and assisted in the editing and proofreading.

My completion of this thesis could not have been accomplished without the support of my dear friends; special thanks to my first friend in Leipzig, Amrapali Zaveri for all her help, care and patience in getting things done. My Iranian friends who relieved my homesickness with their companionship, affection and encouragement.

Finally, I would like to thank my family, especially my mother, for their generous support, care, many sacrifices and love. Your encouragement when the times got rough are much appreciated and duly noted. This dissertation is dedicated to them.

Publications

This thesis is based on the following journal and conference publications, in which I have been an author or a contributor:

Journal Publications, peer-reviewed

- **In Press:** Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data", *Journal of Web Semantics Science, Services and Agents on the World Wide Web*, 2014.
- **Published:** Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler, "Generating SPARQL queries Using Templates", *Web Intelligence and Agent Systems Journal* 11.3 (2013) pp. 283–295.
- **Published:** Edgard Marx, Tommaso Soru, Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, and Karin Breitman, "Towards an Efficient RDF Dataset Slicing", *Int. J. Semantic Computing* 7.4 (2013) p. 455.
- **Published:** Sebastian Tramp, Philipp Frischmuth, Timofey Ermilov, Saeedeh Shekarpour, and Sören Auer, "An architecture of a distributed semantic social network", *Semantic Web* 5.1 (2014) pp. 77–95.

Conference Publications, peer-reviewed

- **Submitted:** Saeedeh Shekarpour and Sören Auer, "Query Reformulation on RDF Knowledge Bases using Hidden Markov Models", *Submitted to the Eighth International Conference on Web Search and Web Data Mining, WSDM 2015*, 2015.
- **Published:** Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "Question Answering on Interlinked Data", *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, 2013 pp. 1145–1156.
- **Published:** Saeedeh Shekarpour, Konrad Höffner, Jens Lehmann, and Sören Auer, "Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features", *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, 2013 pp. 191–197.
- **Published:** Edgard Marx, Saeedeh Shekarpour, Sören Auer, and Axel-Cyrille Ngonga Ngomo, "Large-scale RDF Dataset Slicing", *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, 2013.

- **Published:** Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, “Keyword-Driven Resource Disambiguation over RDF Knowledge Bases”, *Semantic Technology, Second Joint International Conference, JIST 2012, Nara, Japan, December 2-4, 2012. Proceedings*, Springer, 2012 pp. 159–174.
- **Published:** Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler, “Keyword-Driven SPARQL Query Generation Leveraging Background Knowledge”, *Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2011, Campus Scientifique de la Doua, Lyon, France, August 22-27, 2011*, 2011 pp. 203–210.

Workshop and Doctoral Consutiom Publications, peer-reviewed

- **Published:** Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, “Query Segmentation and Resource Disambiguation Leveraging Background Knowledge”, *Proceedings of WoLE Workshop*, 2012.
- **Published:** Saeedeh Shekarpour, “DC Proposal: Automatically Transforming Keyword Queries to SPARQL on Large-Scale Knowledge Bases”, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part II*, Springer, 2011 pp. 357–364.

Abstract

The Web of Data contains a wealth of knowledge belonging to a large number of domains. Retrieving data from such precious interlinked knowledge bases is an issue. By taking the structure of data into account, it is expected that upcoming generation of search engines is approaching to question answering systems, which directly answer user questions. But developing a question answering over these interlinked data sources is still challenging because of two inherent characteristics: First, different datasets employ heterogeneous schemas and each one may only contain a part of the answer for a certain question. Second, constructing a federated formal query across different datasets requires exploiting links between these datasets on both the schema and instance levels. In this respect, several challenges such as resource disambiguation, vocabulary mismatch, inference, link traversal are raised. In this dissertation, we address these challenges in order to build a question answering system for Linked Data. We present our question answering system Sina, which transforms user-supplied queries (i.e. either natural language queries or keyword queries) into conjunctive SPARQL queries over a set of interlinked data sources. The contributions of this work are as follows:

1. A novel approach for determining the most suitable resources for a user-supplied query from different datasets (disambiguation approach). We employed a Hidden Markov Model, whose parameters were bootstrapped with different distribution functions.
2. A novel method for constructing federated formal queries using the disambiguated resources and leveraging the linking structure of the underlying datasets. This approach essentially relies on a combination of domain and range inference as well as a link traversal method for constructing a connected graph, which ultimately renders a corresponding SPARQL query.
3. Regarding the problem of vocabulary mismatch, our contribution is divided into two parts, First, we introduce a number of new query expansion features based on semantic and linguistic inferencing over Linked Data. We evaluate the effectiveness of each feature individually as well as their combinations, employing Support Vector Machines and Decision Trees. Second, we propose a novel method for automatic query expansion, which employs a Hidden Markov Model to obtain the optimal tuples of derived words.
4. We provide two benchmarks for two different tasks to the community of question answering systems. The first one is used for the task of question answering on interlinked datasets (i.e. federated queries over Linked Data). The second one is used for the vocabulary mismatch task.

We evaluate the accuracy of our approach using measures like mean reciprocal rank, precision, recall, and F-measure on three interlinked life-science datasets as well as DBpedia. The results of our accuracy evaluation demonstrate the effectiveness of our approach. Moreover, we study the runtime of our approach in its sequential as well as parallel implementations and draw conclusions on the scalability of our approach on Linked Data.

Contents

CHAPTER 1

Introduction

Given the ever-increasing amount of information being published on the Web, organizing and integrating this information, data and knowledge is a major challenge. The Semantic Web initiative responds to this challenge by introducing standards such as the Resource Description Framework (RDF)[1], RDF-Schema[2] and OWL[3] for publishing information in machine-readable formats. The heart of the technologies behind Semantic Web is the RDF, which is a standard for describing Web resources. A resource can be any thing (either physical or conceptual). For example, a resource can be a Web site, a person, a device or anything else. The RDF data model expresses statements about Web resources in the form of subject-predicate-object (triple). The subject denotes a resource; the predicate expresses a property of subject or a relationship between the subject and the object; the object is either a resource or literal. For example, the statement "Jack knows Alice" in RDF denotes the relationship "knowing" between the two resources "Jack" and "Alice". For identifying resources, RDF uses Uniform Resource Identifiers (URIs)[4] and Internationalized Resource Identifier (IRIs)[5]. The rationale behind is that the names of resources must be universally unique.

RDF is used by Semantic Web tools and frameworks to publish structured data whose meaning is described in RDF Schema (RDFS) or the Web Ontology Language (OWL). In addition to the publishing of structured data, RDF allows the interlinking and merging of data across the Web [13, 14]. As a result of the Semantic Web vision and, more importantly publishing large amounts of structured data on the Web, the concept of the Web of Data emerged. The Web of Data refers to the set of knowledge bases published according to the Linked Data principles[6], i.e. a set of best practices for publishing and connecting structured data on the Web [15, 16].

Since its creation in 2007, the Linked Data Web has been growing at an astounding rate. Currently, LOD Cloud reports publishing more than 31 Billion triples[7]. The sheer amount of data contained therein poses an important challenge as to *how to query this amount of knowledge*. In other words, it is increasingly difficult for end users to find the information they are looking for on this enormous Web of Data.

[1] `http://www.w3.org/TR/rdf11-concepts/`
[2] `http://www.w3.org/TR/rdf-schema/`
[3] `http://www.w3.org/TR/owl-ref/`
[4] A URI is a string of characters used as unique identifier for a Web resource.
[5] A generalization of URIs enabling the use of international character sets.
[6] `http://www.w3.org/DesignIssues/LinkedData.html`
[7] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/` (Last update: September 19th, 2011)

Moreover, by taking into account the structure of data on the Web of Data, it is expected that search approaches directly provide answers to queries. Traditional information retrieval approaches are not applicable here because they cannot exploit the internal structure of data due to their use of bag-of-words semantics. Although traditional search services like *Google* have extended their functionality to provide direct answers to simple queries which match certain templates, e.g., *"Capital of Spain"*, yet, they lack the ability to answer complex queries or consolidate information from different resources (i.e. integration of, and reasoning on, data on the Web.). These limitations are due to the inherently unstructured nature of information in the Web of Documents. Thus, we need to investigate new search approaches relying on the structure of data for retrieving information. With this consideration, retrieval of information is more semantic-oriented leading to a more intelligent Web. We should note that a main obstacle to the realization of this approach lies in the sheer size of the Data Web, which requires efficient and scalable retrieval approaches.

1.1 Motivation

1.1.1 Objective of a Question Answering System on Linked Data

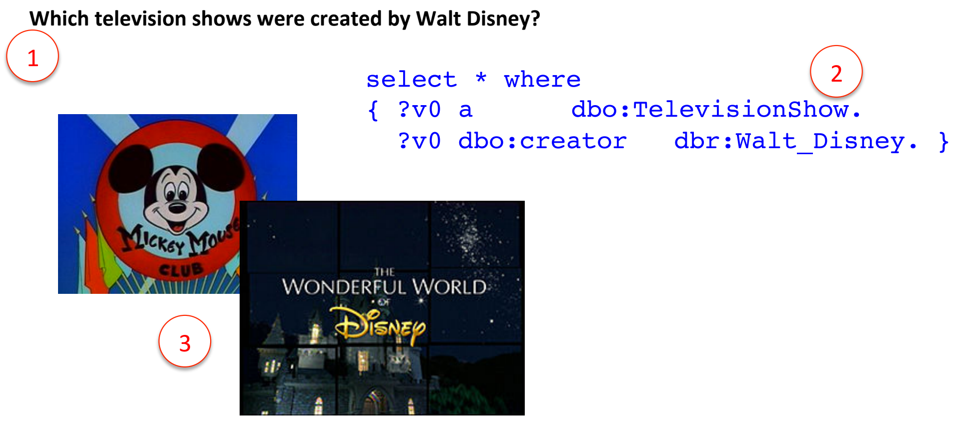
SPARQL[8] is an RDF query language, which enables the retrieval of data from RDF knowledge bases. SPARQL queries permit to express unambiguously which entities and relations are relevant to be retrieved. In order to express information needs in terms of SPARQL queries, non-expert users have to (a) understand the SPARQL concepts, (b) understand the SPARQL syntax (in absence of a visual query builder) and (c) know what information structures (i.e. schemas) are actually available. In other words, for a common user, textual query (either natural language or keyword-based) is a more convenient way of retrieving information.

To enable common users to access the Data Web, it becomes necessary to *simplify the access to the Web of Data* by providing search interfaces that resemble the search interfaces commonly used on the document-based Web. Figure 1.1 envisions three phases of such search engines on the Web of Data: (a) first, a user inserts a textual query, (b) then, it is transformed to a SPARQL query and (c) finally, the SPARQL query is run and desired entities are retrieved. These entities are the actual resources that the user desires.

1.1.2 Influence of a Question Answering System on Society

Advancing question answering can positively influence society in many ways, especially when, in addition to textual interfaces, voice interfaces are provided. Voice interfaces are more accessible for people with disabilities (e.g. vision impaired) or in situations in which typing is inconvenient (e.g. when driving a car). In the following, we discuss the influence of question answering systems on four important sectors of society: (i) science and technology, (ii) welfare, (iii) education, (iv) health and wellbeing (as shown in Figure 1.2. As the proof of applicability, we exemplify the influence of two recent technologies i.e. Google Glass and IBM Watson. Google Glass, developed by Google, is a optical head-mounted display with an integrated computer. It has a voice interface that allows users to interact with Internet services via natural language voice commands. Watson, developed by IBM, is an intelligent computer system which is able to answer natural language questions. Watson was build on the IBM's DeepQA software technology which generates hypotheses, gathers massive evidence, and analyzes data [17]. IBM intends to market the DeepQA software to large corporations for various applications.

[8] `http://www.w3.org/TR/sparql11-query/`



```
select * where
{ ?v0 a       dbo:TelevisionShow.
  ?v0 dbo:creator   dbr:Walt_Disney. }
```

Figure 1.1: A bird's-eye view of the envisioned semantic search engine on the Web of Data: (a) an input textual query (b) the equivalent SPARQL query and (c) the retrieved entities after running the SPARQL query.

Science and technology

The amount of structured as well as unstructured data is growing enormously. We need advanced technologies to help us make sense of this data and make better decisions. Question answering is the area which harnesses the right information to act upon and make us more aware of existing knowledge. A question answering system can be integrated in various applications and devices like web and mobile applications, GPS systems etc. Thus, completely new categories of science and ubiquitous technology become possible. "Cognitive computing systems" is one of the emerging fields which models human brain, reasons, senses[9]. IBM defines a cognitive computing system as "a system which learns and interacts naturally with people to extend what either humans or machine could do on their own. They help human experts make better decisions by penetrating the complexity of Big Data." Watson is a cognitive technology which processes information more like a human than a computer by understanding natural language, generating hypotheses based on evidence and learning as it goes.

IBM aims to market the Watson technology in the healthcare, finance, marketing and service domains. For instance, regarding finances, IBM states that: "the challenges the financial services industry faces are complex. On the one hand, regulatory measures as well as social and governmental pressure for financial institutions to be more inclusive, have increased; On the other hand, customers whom the industry serves, are more empowered, demanding and sophisticated than ever before." With Watson IBM, in particular, aims to support three areas of finance: (a) increasing customer satisfaction and attracting new capital using personalised advice as well as wealth advisor, (b) financial analysis and (c) risk management.

Welfare

It is necessary that information on the Web is accessible and usable to all users regardless of their capabilities. Question answering systems as an emerging assistive technology[10] provide simpler access to information especially when equipped with a voice interface. Thus, a question answering system will benefit people with different capabilities like children, elderly people, disabled or vision impaired people in particular. Since these groups of people are interested in minimum interactions with an application,

[9] `http://www.research.ibm.com/cognitive-computing`

[10] Basically, assistive technology is referring to a device, piece of equipment or product devised to increase functional capabilities of individuals. For example, a screen reader is a software application presenting what is displayed on the screen.

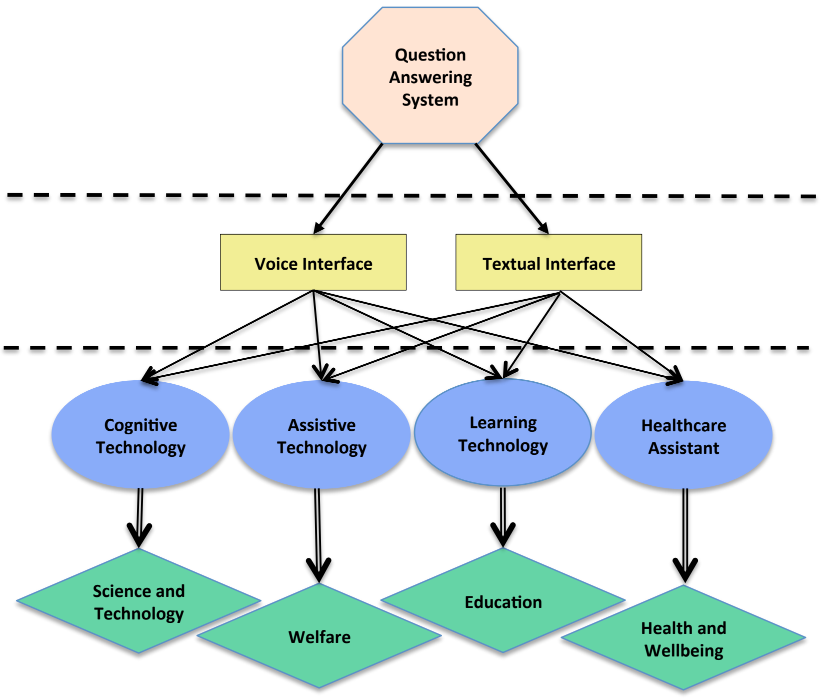


Figure 1.2: Influence of a question answering system on society.

question answering systems help them by reducing the complexity of interacting with an application via automatic access to information.

There is only little research on the accessibility of interfaces of web search engines. Oppenheim was one of the pioneers who challenged the accessibility of web search engine interfaces for vision impaired and blind users [18]. Yang proposed a specialized search engine for blind users [19]. This search engine is constructed with an accessible interface and some improved functions for blind users (i.e., searching assistance functions and specialized design). In September 2013, Google offered alternative access modes[11], such as the Chrome browser which supports assistive technologies including screen readers and magnifiers. This browser offers people with low vision a number of tools, including full-page zoom and high-contrast colour.

One of the recent technologies which is capable of helping blind and impaired vision people is Google Glass technology. The OpenGlass Project[12] is a project employing Google Glass technology to develop applications enabling blind and visually impaired users to identify objects and environments via established crowdsourcing technologies. The VisionAware[13] portal of the American Foundation for the Blind describes two OpenGlass applications which have been developed so far as follows:

1. The *Question-Answer application* enables blind and visually impaired users to use Google Glass to take a picture with a question attached, which is related to the context of the picture. Then this question is submitted to Twitter[14] or Amazon's Mechanical Turk platform[15] where sighted respondents answer. Finally, the answer is read aloud to the user through the speaker of the Google

[11] `http://www.google.com/accessibility/products/`
[12] `http://www.openshades.com/`
[13] `http://www.visionaware.org/`
[14] `https://twitter.com/`
[15] `https://www.mturk.com/mturk/welcome`

Glass headset.

2. The *Memento application* automatically recites notes when the blind or visually impaired user faces, or looks at, a recognizable scene. To use Memento, sighted users must first record descriptions or commentary about environmental features or a room setup. When a blind or visually impaired person using Google Glass approaches the same spot, Google Glass will recognize the feature or scene and read back the pre-recorded commentary.

Such examples show that assistive technology based on question answering enables completely new opportunities to compensate for disabilities or varying capabilities.

Education

Leveraging recent question answering and assistive technologies (i.e. also IBM Watson and Google Glass) opens new opportunities in education. For instance, these technologies assist learners as well as researchers with finding information faster; or they support integrating data from different data sources which will help teachers as well as researchers to make better decisions. In the following, we mention different possibilities of using these two recent technologies in education.

Watson helps researchers to find information faster. IBM says[16]: "The Economist estimates companies spent $603 billion on research and development in 2012. That is a lot of information to wade through. The IBM Watson Discovery Advisor is a research assistant that helps researchers collect information and synthesize insights to stay updated on recent findings and share information with colleagues. New York Genome Center[17] plans to use the IBM Watson cognitive computing system to analyze the genomic data of patients diagnosed with a highly aggressive and malignant brain cancer, and to more rapidly deliver personalized, life-saving treatment to patients of this disease."

Google Glass, as a portable technology, has the potential to bring new possibilities to teachers and students. Foradian[18], a provider of enterprise software solutions for education institutions, reports the use cases of Google Glass as follows:

1. "Google Search is phenomenal, and the Glass will allow the student as well as teacher to stay connected to an interactive environment featuring online tools all the time. This could pave way to a leap into the future of educational system. Teachers as well as students can *refer to topics related to their studies* on the go. No fiddling through phones in the middle of the lecture; all you have to do is speak and voila, your search is done.

2. Teachers could use Google glass coupled with facial recognition to *take attendance and could be used to generate Student Information System.* Just by looking at the student you will get access to his/her student records with details of academic and non-academic performance, attendance etc. Creating students reports, schedules and class timings for students is only the tip of the iceberg."

Health and wellbeing

There is a vast amount of biomedical data (either structured or unstructured) which has been published[19] so far. A question answering system over such data can help researchers, doctors and ordinary people

[16] `http://www.ibm.com/smarterplanet/us/en/ibmwatson/work.html`
[17] `http://www.nygenome.org/`
[18] `http://foradian.com/`
[19] Regarding structured format, RDF biomedical datasets are available at: `http://download.bio2rdf.org/release/3/release.html`

gain recent biomedical knowledge; for example about drugs, diseases etc. Researchers can leverage a question answering system on biomedical data to update their knowledge about the recent findings in their own field or related fields and thus make interesting discoveries. A question answering system helps doctors in diagnosing and treatment phases [20, 21] like AskHermes[20] [22]. Previously, crowdsourcing was one of the first attempts to assist ordinary people in order to find answers for their questions related to health and wellbeing such as *CreateHealth.io*[21]. Nowadays, a question answering system empowers ordinary peoples to directly access information. In the following, we present the use cases of Watson as well as Google Glass in the field of healthcare.

Healthcare is the first real use case of Watson. IBM declares[22]: "In health care, Watson and Watson-like technologies are now assisting doctors at Memorial Sloan Kettering[23] in diagnosing patients by providing a variety of possible causes for a set of symptoms. Watson can help doctors narrow down the options and pick the best treatments for their patients. The doctor still does most of the thinking, but Watson is there to make sense of the data and help make the process faster and more accurate."

Several proofs of concept for Google Glass have been proposed in healthcare. In July 2013, Lucien Engelen initiated research on the usability of Google Glass in the health care field[24]. This research was carried out in operating rooms, ambulances, a trauma helicopter, general practice, and home care as well as the use in public transportation for visually or physically impaired[25]. Research contained making pictures, videos streaming to other locations dictating operative log, having students watch the procedures and tele-consultation through Hangout. On June 20, 2013, Rafael J. Grossmann, a Venezuelan surgeon practicing in the USA, was the first surgeon to ever demonstrate the use of Google Glass during a live surgical procedure[26].

In conclusion, question answering technology is increasingly penetrating various domains of society and technology. However, most systems currently available mainly use unstructured and semi-structured content or fixed, predefined structured data

1.1.3 Existing Question Answering Systems

In the following, we present a number of well-known question answering systems running on structured or unstructured data.

1. *IBM Watson*[27] is a question answering system developed by IBM. In 2011 it answered Jeopardy questions and outperformed former human winners. Watson uses a corpus containing more than 200 million pages of unstructured information as well as semi-structured data like Wikipedia and DBpedia. Watson applies advanced natural language processing, information retrieval, knowledge representation, automated reasoning, and machine learning technologies to analyze the natural language input query and find the answer in the corpus.

2. *START*[28] was initiated by the InfoLab Group at the MIT in 1993. Up to now, this system can answer millions of English questions about places, movies, people, dictionary definitions. START

[20] `http://www.askhermes.org/index2.html`
[21] `http://createhealth.io/`
[22] `http://www.ibm.com/smarterplanet/us/en/ibmwatson/work.html`
[23] Memorial Sloan Kettering Cancer Center is a cancer treatment and research institution founded in 1884 as the New York Cancer Hospital.
[24] `http://exponential.singularityu.org/medicine/`
[25] Recent findings at: `http://brengkenniscentrum.nl/blog/2013/12/een-nieuw-perspectief-google-glass/`
[26] `http://www.forbes.com/fdc/welcome_mjx.shtml`
[27] `http://www.ibm.com/smarterplanet/us/en/ibmwatson/`
[28] `http://start.csail.mit.edu/index.php`

receives natural language queries, then parses them in order to create parse trees, then it matches the created trees against its knowledge base and provides the appropriate information to the user.

3. *Ephyra*[29] is a modular framework for open domain question answering. Similar to other question answering systems, Ephyra retrieves answers for a given natural language question. It uses the whole of the Web as the underlying corpus.

4. *WolframAlpha*[30] is a question answering system which was developed by Wolfram research group and released in 2009. It is a computational knowledge engine or answer engine run on a curated structured datasets[31]. It answers factual queries (i.e. phrased natural-language fact-based questions) directly by computing the answer from different sources.

5. *PowerAqua*[32] is a multi-ontology-based question answering system running on Linked Data sources. It receives a natural language query as input and then returns answers drawn from the relevant distributed resources on the Semantic Web. PowerAqua is not restricted to a single ontology and therefore supports various domains.

6. *TBSL*[33] is a template-based question answering system on a single Linked Data knowledge base (i.e. DBpedia). It parses an input natural language question and then produces a SPARQL template reflecting the internal structure of the question.

1.1.4 Existing Search Engines on Linked Data

In the following, we present existing search services, which have been launched on the Linked Data so far.

1. *Sindice*[34] is a document-oriented index on Web of Data. Sindice collects RDF documents from the Semantic Web and indexes them on resource URIs, Inverse Functional Properties (IFPs) and keywords. This index allows applications to automatically locate documents containing semantic resources via an API. Furthermore, Sindice offers a user interface through which human users can retrieve these documents based on keywords, URIs, or IFPs.

2. *Sigma*[35] is established on top of Sindice which gives a very visual and interactive access to the *Web of Data* as a whole. Sigma is an entity-based retrieval service which employs large scale Semantic Web indexing, logic reasoning, data aggregation heuristics, ad-hoc ontology consolidation, external services and responsive user interaction to create entity descriptions. Sigma offers both a service and an end user application to access the Web of Data as an integrated information space.

3. *Watson*[36] is a search service indexing ontologies as well as semantic documents. Watson collects the available semantic content on the Web and then analyzes it to extract useful metadata and indexes. Watson enables user to query semantic data using either keyword interface or API.

[29] `http://www.ephyra.info/`
[30] `http://www.wolframalpha.com/`
[31] Curated datasets are checked for quality either by a scientist or an expert in an associated field
[32] `http://poweraqua.open.ac.uk:8080/poweraqualinked/jsp/index.jsp`
[33] `http://autosparql-tbsl.dl-learner.org/`
[34] `http://sindice.com/`
[35] `http://sig.ma/`
[36] `http://watson.kmi.open.ac.uk/WatsonWUI/`

1.2 Problem

While various search approaches differ in their details, they can all be positioned on the following spectrum: On one end of the spectrum are simple keyword search systems that rely on traditional information retrieval approaches to retrieve resources that bear a label similar to the "the user's input". We dub such approaches as *data-semantics-unaware keyword search* since they do not take the semantics explicated by the data into consideration. The main advantage of such approaches is that they scale well as they can make use of the results of decades of research carried out in the field of information retrieval. On the other end of the spectrum, we find *question answering systems*, which assume a natural-language query as input and convert this query into a full-fledged SPARQL query. These systems rely on natural-language processing tools such as Part of Speech (POS) tagging and dependency parsers to detect the relations between the elements of the query. The detected relations are then mapped to SPARQL constructs. The basic idea behind our work is to devise a *data-semantics-aware keyword search* approach, which stands in the middle of the spectrum as shown in Figure 1.3 Our approach, which is called **SINA**, aims to achieve maximal flexibility by being able to generate SPARQL queries from both natural-language queries and keyword queries. Several challenges need to be addressed to devise such an approach. In the following sections, we introduce these challenges.

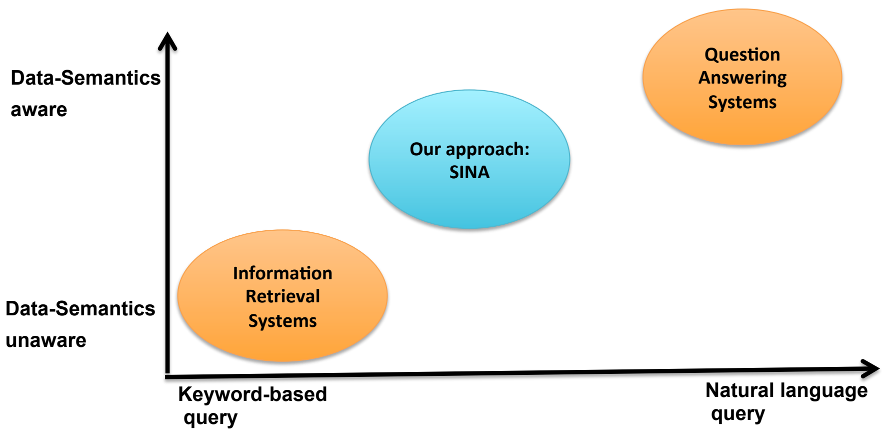


Figure 1.3: Comparison of search approaches.

1.3 Challenges

We aim to realize a search engine for the Data Web, which is as easy to use as search engines for the Document Web, but allows to create complex queries and returns comprehensive structured query results. In order to achieve this goal, a number of challenges are raised:

- Query Segmentation
- Resource Disambiguation
- Query Expansion

- Query Cleaning
- Formal Query Construction
- Data Fusion on Linked Data

In the rest of this section, we briefly describe each challenge. In order to obtain a better insight on these challenges, we use a few running examples throughout the discussion. Before presenting them, we introduce the datasets employed in this dissertation. We employed four different knowledge bases: first is *DBpedia* [23] as an individual knowledge base. DBpedia is a large knowledge base extracted from Wikipedia. Additionally, we used three interlinked knowledge bases i.e., *Drugbank*, *Sider* and *Diseasome* which have been published in RDF. Sider[37] contains information about drugs and their side effects. Diseasome [24][38] contains information about diseases and genes associated with these diseases. Drugbank [25][39] is a comprehensive knowledge base containing information about drugs, drug target (i.e. protein) information, interactions and enzymes. As can be seen in Figure 1.4 the classes representing drugs in Drugbank and Sider are linked using `owl:sameAs` and diseases from Diseasome are linked to drugs in Drugbank using `Diseasome:possible-Drug` and `Drugbank:possible-Disease-target` properties. Also, the diseases and side effects between Sider and Diseasome are linked using the `owl:sameAs` property.

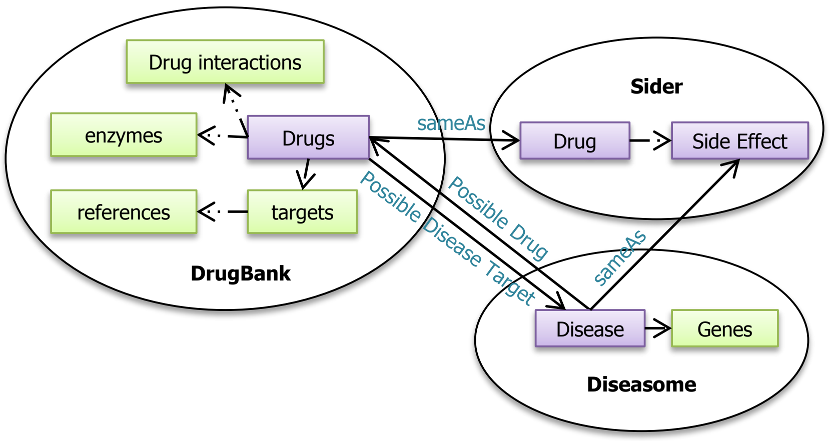


Figure 1.4: Schema interlinking between three datasets: DrugBank, Sider and Diseasome.

We use two query examples throughout our discussion in order to clarify better. Assume that the input queries are as follows:

Example 1.1 `What are the side effects of drugs used for Tuberculosis?`

Example 1.2 `Who produced films starring Natalie Portman?`

[37] http://sideeffects.embl.de/
[38] http://diseasome.kobic.re.kr/
[39] http://www.drugbank.ca/

1.3.1 Query Segmentation

Query segmentation is the process of identifying the *right segments of data items* that occur in the keyword queries. Regarding example 1, the input query *'What is the side effects of drugs used for Tuberculosis?'* is transformed to the 4-keyword tuple *(side, effect, drug, Tuberculosis)*. This tuple can be segmented into (*'side effect drug'*, *'Tuberculosis'*) or (*'side effect'*, *'drug'*, *'Tuberculosis'*). Similarly, the query of example 2 can be segmented to (*'produce'*, *'film star'*, *'Natalie'*, *'Portman'*) or (*'produce'*, *'film'*, *'star'*,*'Natalie Portman'*). Note that in both cases, the second segmentation is more likely to lead to a query that contains the results intended by the user.

1.3.2 Resource Disambiguation

In addition to detecting the right segments for a given input query, we also have to map each of these segments to a suitable resource in the underlying knowledge base. This step is dubbed *entity disambiguation* and is of increasing importance since the size of knowledge bases and the heterogeneity of schemas on the Linked Data Web grows steadily. With respect to example 1, the segment *'Tuberculosis'* is ambiguous when querying both Sider and Diseasome because it may refer to the resource `diseasome:Tuberculosis` describing the disease Tuberculosis or to the resource `sider:Tuberculosis` being the side effect caused by some drugs. Regarding the example 2, the segment *'film'* is ambiguous because it may refer to the class `dbo:Film` (the class of all movies in DBpedia) or to the properties `dbo:film` or `dbp:film` (which relates festivals and the films shown during these festivals). In fact in this step, we aim to map the input keywords to a suitable set of entity identifiers, i.e. resources $R = \{r_1, r_2, \ldots, r_m\}$. Note that several adjacent keywords can be mapped to a single resource, i.e. $m \leq n$. Thus, for each segment, a suitable resource has to be determined.

1.3.3 Query Expansion

Automatic query expansion is a long-standing research topic in information retrieval. It is a way of reformulating the input query in order to overcome the vocabulary mismatch problem. In case of a vocabulary mismatch, schema-aware search systems are unable to retrieve data. For instance, consider the input query *altitude of Everest*. The keyword *altitude*, should be matched to the keyword *elevation*, because the relevant property resource has the label *elevation* and not *altitude*. Therefore, query expansion can be a crucial step in question answering or keyword search pipeline. A naive way for automatic query expansion adds words derived from linguistic resources. In this regard, expansions are *synonyms*, *hyponyms* and *hypernyms* of the input keywords. In practice, this naive approach fails because of high retrieval cost and substantially decreasing precision. Regarding Linked Data, a research question arising here is whether interlinked data and vocabularies provide features, which can be taken into account for query expansion and how effective those new semantic features are in comparison to traditional linguistic ones.

1.3.4 Query Cleaning

The input query might contain some keywords, which semantically are either not related to the rest of the keywords or extra (i.e. does not have any matched resource in the corresponding SPARQL query). Since user usually is looking for information semantically closely related to each other, these unrelated keywords (i.e. noise) should be cleaned. An example is the question *"Through which countries does the Yenisei river flow?"* The keyword *flow* is not a stop word but does not have any matched resource in the corresponding SPARQL query of the benchmark (our benchmark contains a couple of natural language queries and their equivalent SPARQL query); and therefore should be ignored.

1.3.5 Formal Query Construction

Once the resources are detected, adequate formal queries (i.e. SPARQL queries) have to be generated. In order to generate a formal query (here: a conjunctive query), a connected subgraph $G' = (V', E')$ of the knowledge base graph $G = (V, E)$, called the **query graph**, has to be determined. The intuition behind constructing such a query graph is that it has to fully cover the set of mapped resources $R = \{r_1, \ldots, r_m\}$. In linked data, mapped resources r_i may belong to different graphs G_i. Thus, the query construction algorithm must be able to traverse the links between datasets at both schema and instance levels. With respect to the example 1, after applying disambiguation on the identified resources, we would obtain the following resources from different datasets:

`sider:sideEffect`, `diseasome:possibleDrug`, `diseasome:1154`. The appropriate conjunctive query contains the following triple patterns which the second triple pattern bridges between the Drugbank and Sider datasets.:

```
1. diseasome:1154  diseasome:possibleDrug      ?v1 .
2. ?v1             owl:sameAs                  ?v2 .
3. ?v2             sider:sideEffect            ?v3 .
```

1.3.6 Data Fusion on Linked Data

We aim at an approach for question answering over a set of interlinked data sources. In this respect, new challenges are raised that we have to deal with. A first challenge is that information for answering a certain question can be spread among different datasets employing heterogeneous schemas. This makes the mapping of the input keywords to data more challenging when compared to querying a single dataset. An example is the query: "side effect and enzymes of drugs used for ASTHMA". The answer to that query can only be obtained by joining data from Sider (side effects) and Drugbank (enzymes, drugs). The second challenge is constructing a formal query from the matched resources across different datasets which have to exploit links between the different datasets on the schema and instance levels. An example is the query: "side effects of drugs used for Tuberculosis". Tuberculosis is defined in Diseasome, drugs for curing Tuberculosis are described in Drugbank, while we find their side effects in Sider.

1.4 Approach and Contribution

To the best of our knowledge, our approach is the first approach for answering questions on interlinked datasets by constructing a federated SPARQL query. Our approach led to the implementation of the Sina search engine. Sina is a Java based web application, which is available online. We deployed two demo instances, one employing DBpedia as background knowledge[40] and the second one operating on three interlinked life-science datasets[41]. Sina has a simple interface similar to common search engines. All the steps are carried out automatically by Sina and the user does not have to interact with the system during the search process. Figure 1.5 illustrates the high-level architecture of Sina, which comprises of six main components. Each component consumes the output of the previous component.

The challenges, which our approach has addressed so far, are as follows:

[40] `http://sina.aksw.org/`
[41] `http://sina-linkeddata.aksw.org/`

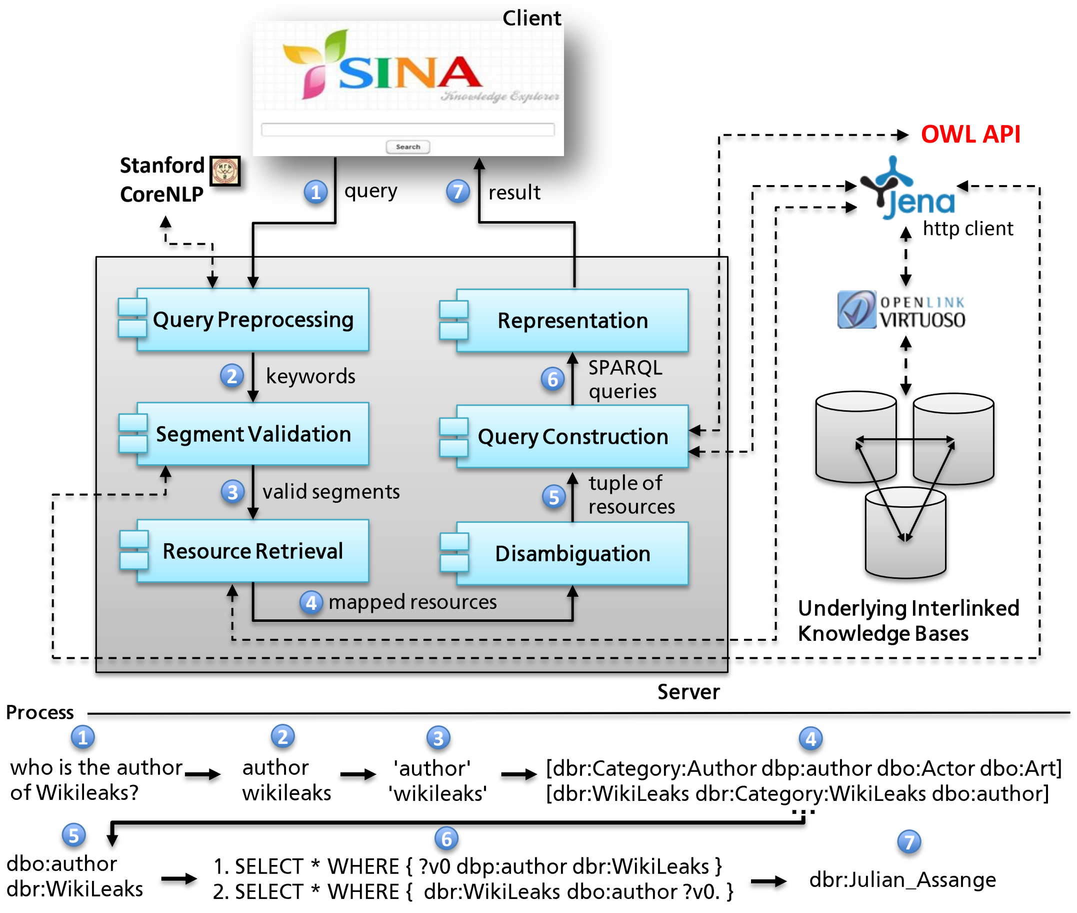


Figure 1.5: Architecture of the SINA search engine.

- Query Segmentation (QS)
- Resource Disambiguation (RD)
- Query Expansion (QE)
- Formal Query Construction (FQC)
- Data Fusion on Linked Data (DF on LD)
- Benchmark Creation

In addition to the contribution in the above challenges, we took part in the RDF Data Slicing (RDF DS) project. Table 1.1 corresponds each challenge to the list of publications which address it. Furthermore, Table 1.2 presents an overview of the structure of this dissertation. Each chapter is dedicated to one of our publications, which addresses one or more challenges. Appendix A provides one of our papers [9] comprising a more experimental study on the approach employed for query segmentation and resource disambiguation. The article associated with our contribution in the *RDF Data Slicing* projects are included in Appendix B. In the rest of this chapter, we briefly point out the methodology employed for each challenge as well as the research questions used for the evaluation and the achievements. In order to gain a full insight, we refer you to the associated chapters and publications.

Challenges	Publications
Query Segmentation	[1, 6, 9, 11]
Resource Disambiguation	[1, 6, 9, 11]
Query Expansion	[5, 7]
Formal Query Construction	[1, 2, 6, 10]
Data Fusion on Linked Data	[1, 6]
RDF Data Slicing	[3, 8]

Table 1.1: List of the challenges and the publications addressing them.

1.4.1 Query Segmentation and Resource Disambiguation

As shown in Table 1.2, our contribution on this part can be found in chapter 3 or in our publications [1, 6, 9, 11]. We shortly describe the main attributes of the proposed model.

Methodology: To tackle this challenge, we propose an automatic query segmentation and resource disambiguation approach leveraging background knowledge. We employ a Hidden Markov Model (HMM) to obtain the optimal input query segmentation and disambiguation of possible matches in a single step. In fact, query segmentation and resource disambiguation are mutually tightly interwoven. Before running this model, we carry out two preprocessing functions as follows:

1. The segment validation which groups keywords to form segments. This function validates the grouped segments with respect to the available resources in the underlying knowledge base(s). Recognizing segments are based on a naive approach (we compare this approach with a gready fashion in [9, 11]).

2. The **resource retrieval** function obtains relevant resources from the underlying knowledge bases. The retrieval is based on the string matching between valid segments and the `rdfs:label` of resources. Furthermore, more resources are inferred from lightweight `owl:sameAs` reasoning.

In this model, the input n-tuple of keywords is considered as the sequence of observations. The state space is populated with states which a valid segment can be observed. Background knowledge (i.e. semantic relatedness of two resources) is used for parameter estimation. The semantic relatedness is defined in terms of two parameters: the distance between the two resources and the popularity of each

Chapters	QS & RD	FQC		QE		DF on LD	RDF DS
		Template	*Inference*	*Semantic Features*	*Query Reformulation*		
Chapter 2 [2]	-	✔	-	-	-	-	-
Chapter 3 [1]	✔	-	✔	-	-	✔	-
Chapter 4 [7]	-	-	-	✔	-	-	-
Chapter 5 [5]	-	-	-	-	✔	-	-
Appendix A [9]	✔	-	-	-	-	-	-
Appendix B [3]	-	-	-	-	-	-	✔

Table 1.2: Overview of the thesis structure: chapters along with their corresponding publication & the addressed challenges.

of the resources. The distance between two resources is the path length between those resources. The popularity of a resource is simply the connectivity degree (sum of both in and out degree) of the resource with other resources available in the state space. We use the HITS algorithm for transforming these two values to hub and authority values [1, 6].

Natural language processing (NLP) techniques are commonly used for text segmentation. We also developed a technique which is a combination of named entity and multi-word unit recognition services as well as POS-tagging for segmenting the input-query. Named entity, multi-word unit recognition and POS-tagging fail in the case of an incomplete sentences (e.g. for keyword-based queries), we show that our statistical approach is robust with respect to query expression variance (see Appendix A).

Evaluation: We rely on *bootstrapping*, a technique used to estimate an unknown probability distribution function. We test different bootstrapping methods for the HMM parameters using various distributions (Normal, Zipf, Uniform) as well as an algorithm based on Hyperlink-Induced Topic Search (HITS). The goal of ultimate evaluation was to determine effectiveness and efficiency the resource disambiguation. We measured the effectiveness of our resource disambiguation approach using the *Mean Reciprocal Rank* (MRR).

Results: Our proposed functions for HMM parameters produce the best results for both segmentation and disambiguation. As the output, this model can provide a ranked list of all paths generating the observation sequence with the corresponding probability. The following example shows a sample of the output.

Example 1.3 Let us consider the query: `What are the side effects of drugs used for Tuberculosis?`. The validated segments are: `'side effect'`, `'drug'` and `'Tuberculosis'`. After the retrieval and pruning process, the state space contains the resources listed in Table 1.3. By running the *Viterbi algorithm* with the associated probabilities, we obtain a ranked list of the chosen resources that the sequence *{side effect, drug, Tuberculosis}* is observable through them. In the following, we show the top-4 most likely paths along with their associated probability.

```
1. 0.0033:     Sider:sideEffect, Diseasome:possibleDrug, Diseasome:1154.
2. 0.0017:     Sider:sideEffect, Diseasome:possibleDrug, Sider:C0041296.
3. 6.0257E-4:  Sider:sideEffect, Sider:drugs,           Diseasome 1154.
4. 4.0805E-4:  Sider:sideEffect, Drugbank:Offer,        Diseasome:1154.
```

Segment	Resources	Label	Type
side effect	`1. Sider:sideEffect`	side effect	property
	`2. Sider:side_effects`	side effect	class
drug	`1. Drugbank:drugs`	drug	class
	`2. Drug bank:offer`	drug	class
	`3. Sider:drugs`	drug	class
	`4. Diseasome:possibleDrug`	possible drug	property
Tuberculosis	`1. Diseasome:1154`	tuberculosis	instance
	`2. Sider:C0041296`	tuberculosis	instance

Table 1.3: The resources contained in the state space for a given query.

1.4.2 Query Expansion

Our research and contribution in this field is divided in two parts. First, we introduce a number of new query expansion features based on semantic and linguistic inferencing over Linked Open Data. We evaluate the effectiveness of each feature individually as well as their combinations employing several machine learning approaches. Second, we propose a method for automatic query expansion. We employ a *Hidden Markov Model* (HMM) to obtain the optimal tuple of words. In the following, we shortly describe each part. The first part is presented in chapter 4 and the second part is presented in chapter 5.

Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features

The idea here is that automatic query expansion methods can use the graph structure of RDF and follow interlinks between datasets.

Methodology: We introduce several semantic features for query expansion. In addition to linguistic features (i.e. synonyms, hyponyms, hypernyms), the semantic features are defined as the following semantic relations:

- *sameAs:* deriving resources having the same identity as the input resource using *owl:sameAs*.
- *seeAlso:* deriving resources that provide more information about the input resource using *rdfs:seeAlso*.
- *class/property equivalence:* deriving classes or properties providing related descriptions for the input resource using *owl:equivalentClass* and *owl:equivalentProperty*.
- *superclass/-property:* deriving all super classes/properties of the input resource by following the *rdfs:subClassOf* or
  rdfs:subPropertyOf property paths originating from the input resource.
- *subclass/-property:* deriving all sub resources of the input resource r_i by following the *rdfs:subClassOf* or
  rdfs:subPropertyOf property paths ending with the input resource.
- *broader concepts:* deriving broader concepts related to the input resource r_i using the SKOS vocabulary properties *skos:broader* and *skos:broadMatch*.
- *narrower concepts:* deriving narrower concepts related to the input resource r_i using *skos:narrower* and *skos:narrowMatch*.
- *related concepts:* deriving related concepts to the input resource r_i using *skos:closeMatch*, *skos:mappingRelation* and *skos:exactMatch*.

Evaluation: When applying expansion methods, there is a risk of yielding a large set of irrelevant words, which can have a negative impact on further processing steps. For this reason, we measure the effectiveness of all linguistic as well as semantic features individually in order to decide which of those are effective in query expansion. In order to do that we employ two kinds of classifiers (i.e., *Support Vector Machine (SVM)* and *Decision Tree*). The second goal of our experimental study is how well a linear weighted combination of features can predict the relevant words? We do this by using machine learning methods to generate a linear combination of linguistic and semantic query expansion features with the aim of maximizing the F_1-score and efficiency on different benchmark datasets.

Results: Interestingly, the setting with only semantic features result in an accuracy at least as high as the setting with only linguistic features. This observation is an important finding that semantic features appear to be competitive with linguistic features. Our results allow developers of new search engines to integrate *automatic query expansion* with good results without spending much time on its design. This is important, since query expansion is usually not considered in isolation, but rather as one step in a pipeline for question answering or keyword search systems.

Query Reformulation on RDF Knowledge Bases using Hidden Markov Models

Methodology: In order to address the vocabulary mismatch problem, we introduce a novel method for automatic query expansion with respect to background knowledge. We define the concept of triple-based co-occurrence of words in RDF knowledge bases. Our method uses a Hidden Markov Model to determine the most suitable derived words from linguistic resources. We test different bootstrapping methods for the HMM parameters using various distributions as well as an algorithm based on *Hyperlink-Induced Topic Search* (HITS).

Evaluation: We analyze the effect of this approach for both queries requiring expansion and those which do not. The goal of our evaluation is to determine: (1) How effective is our method with regard to a correct reformulation of queries which have vocabulary mismatch problem? (2) How robust is the method for queries which do not have vocabulary mismatch problem?

Results: Our experimental study shows the feasibility and high accuracy of the method. This implementation can provide a ranked list of all paths generating the observation sequence with the corresponding probability.

Example 1.4 Let us consider the input query: `altitude of Everest`. The observation list is shown in the first column of Table 1.4. After constructing and pruning the state space, it contains the words listed in the second column of Table 1.4 (only a subset of state space is presented). By running the *Viterbi algorithm*, we have a ranked list of the chosen words that the sequence `altitude of Everest` is observable through. In the following, we show the top-4 most likely paths along with their associated probability.

```
0.02451:   altitude,    Everest.
0.01681:   elevation,   Everest.
0.01145:   length,      Everest.
0.01145:   height,      Everest.
```

Observation	States	Origin Type
Everest	`Everest`	original keyword
altitude	`altitude`	original keyword
	`elevation`	synonym
	`height`	synonym
	`ceiling`	hyponym
	`level`	hyponym
	`distance`	hypernym
altitude Everest	`altitude Everest`	original keyword

Table 1.4: A subset of the state space along with the origin type, list of all observations for the given query `altitude Everest`.

1.4.3 Formal Query Construction

We propose two methods for generating formal queries. In the first method presented in chapter 2, we use a set of predefined templates. This method is restricted to work only for a limited number of input keywords. In the second method presented in chapter 3, we do not have any predefined template and instead we dynamically generate templates using inference over the type of the identified resources. Thus, we do not have any limitation on the number of input keywords. In the following, we shortly present these two methods.

Generating SPARQL Queries using Templates

Figure 2.2 shows an overview of our approach. Our approach firstly retrieves relevant IRIs related to each user-supplied keyword from the underlying knowledge base and secondly injects them to a series of graph pattern templates for constructing formal queries. To find these relevant IRIs, two steps as (1) mapping keywords to IRIs and (2) ranking and selecting IRIs are carried out.

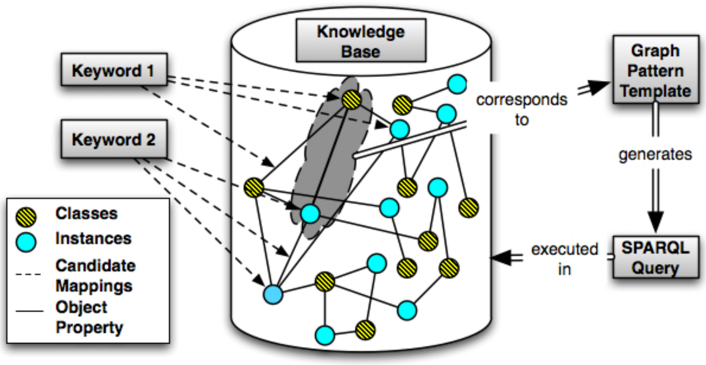


Figure 1.6: Overview of the proposed method.

Methodology: We propose a novel approach for generating SPARQL queries using *graph pattern templates*. We define the concept of *graph pattern template* as a set of triple patterns containing placeholders and variables. As an example, $(s_1, ?p_1, ?o_1)(?o_1, p_2, ?o_2)$ is a graph pattern template that contains two triple pattern templates. Symbols preceded by question marks denote variables while symbols without question marks are placeholders (e.g. s_1 and p_2 are placeholders). Placeholders are replaced by IRIs associated with the input keywords. A placeholder can stand either for a property (when occurring in the predicate position), an instance (when occurring at subject or object position) or a class (when occurring at the object position). The SPARQL queries generated with our approach are a restricted kind of SPARQL queries, since they use only *basic graph patterns*. We analysed 1,000 distinct queries from the query log of the public DBpedia endpoint[42] and learned that the number of IRIs is usually larger than the number of triple patterns occurring in the query. As a consequence of this finding we decided to assume graph patterns for generating SPARQL queries for two user-supplied keywords to consist of either one or two triple patterns. Therefore, this assumption leads to 17 possible graph pattern templates. We perform an accuracy study on all combinatorial possible graph pattern templates. This study showed that a filtered set of patterns limit the search space (thus leading to more efficiency) without reducing the accuracy of our approach significantly. Consequently, we only consider these patterns during the SPARQL-query generation process.

Example 1.5 After applying the mapping and ranking functions to the user query `island of germany`, we obtain two IRIs, i.e. `http://dbpedia.org/ontology/Island` with the

[42] The DBpedia SPARQL endpoint is available at: `http://dbpedia.org/sparql/` and the query log excerpt at: `ftp://download.openlinksw.com/support/dbpedia/`.

type *class* and `http://dbpedia.org/resource/Germany` with the type *instance*. The possible graph pattern templates for these two IRIs are:

1. (?island, a, dbo:Island), (?island, ?p, dbr:Germany)
2. (?island, a, dbo:Island), (dbr:Germany, ?p, ?island)

Our method would generate the following two queries:

1.
   ```
   SELECT * WHERE {
     ?island  a   dbo:Island  .
     ?island  ?p  dbr:Germany . }
   ```

2.
   ```
   SELECT * WHERE {
     ?island     a   dbo:Island .
     dbr:Germany ?p  ?island    . }
   ```

Evaluation: We introduce two new accuracy metrics used for evaluation. Since the user's intention in keyword-based search is ambiguous, judging the correctness of the retrieved answers is a challenging task. Therefore, we introduce the *Correctness Rate (CR)* as a measure for the preference of certain RDF terms. This metric allows a user to rate the correctness of each individual answer based on its own perception. Furthermore, we introduce *fuzzy precision* metric (FP), which measures the overall correctness of a template's corresponding answers with respect to a set of keyword queries Q.

This experiment was done in the following three levels:

1. Accuracy evaluation of possible graph pattern templates: In this study we aim at selecting those templates that lead to a high accuracy
2. Application evaluation: we evaluated the approach based on the three metrics, i.e. fuzzy precision, recall and f-score.
3. Comparative study based on relevance feedback: We compared our approach with a traditional Web search engine (Google) and three Semantic Web search engines (Sindice, Sig.ma and Falcon).

Results: Since the approach is based on simple operations, it can generate and execute SPARQL queries very efficiently. Another advantage of this approach is that it is completely agnostic of the underlying knowledge base as well as its ontology schema. We implemented it as a Java Web application which is publicly available at: `http://lod-query.aksw.org`. The whole query interpretation and processing is performed typically on average in 15 seconds (while first results are already obtained after one second) when using DBpedia as knowledge base. We currently use DBpedia as background knowledge, but the approach is easily transferable to the whole Web of Data.

Generation of SPARQL Queries using Inference

Here, the goal is to construct a conjunctive queries (i.e. SPARQL queries) for a given set of resource identifiers. We address automatic construction of conjunctive queries without using any predefined templates.

Methodology: The core of SPARQL queries is *basic graph patterns*, which can be viewed as a query graph QG. We define relevance probability for triple patterns. We aim at constructing QG with the highest relevance probability. As a result of these considerations, we devise an algorithm that minimizes the number of both free variables and triple patterns in a query graph.

Forward Chaining: One of the prerequisites of our approach is the inference of implicit knowledge on the types of resources as well as domain and range information of the properties. To construct possible query graphs, we generate in a first step an *incomplete query graph* IQG such that the vertices are either equal or subset of the vertices (resp. edges) of the final query graph. In fact, an incomplete query graph (IQG) contains a set of disjoint sub-graphs, i.e. there is no vertex or edge in common between the sub-graphs. For the second step, we use an extension of the minimum spanning tree method that takes subgraphs (and not sets of nodes) as input and generates a minimal spanning graph as output. Since in the second step, the minimum spanning tree does not add any extra intermediate node (except nodes connected by `owl:sameAs` links), it eliminates both the need of keeping an index over the neighborhood of nodes, and of using exploration for finding paths between nodes.

Example 1.6 We look at the query: `What is the side effects of drugs used for Tuberculosis?`. Assume the resource disambiguation process has identified the following resources:

```
1. diseasome:possibleDrug          (type property)
   Domain={diseasome:disease},     Range={drugbank:drugs}
2. diseasome:1154                  (type instance)
   Type={diseasome:disease}
3. sider:sideEffect                (type property)
   Domain={sider:drug},            Range={sider:sideeffect}
```

After running the IQGs generation, we will have two disjoint graphs shown in Figure 1.7. Then we connect these two disjoint graphs depicted in Figure 1.8 and thier corresponding SPARQL queries are as follows:

The corresponding SPARQL queries are as follows:

```
1. SELECT * WHERE {
     diseasome:1154    diseasome:possibleDrug    ?v0 .
     ?v1               sider:sideEffect          ?v2 .
     diseasome:1154    owl:SameAs                ?v2 .  }

2. SELECT * WHERE {
     diseasome:1154    diseasome:possibleDrug    ?v0 .
     ?v1               sider:sideEffect          ?v2 .
     ?v0               owl:SameAs                ?v1 . }
```

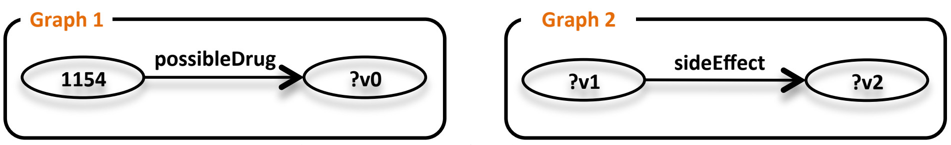


Figure 1.7: *IQG* for the Example 14.

Evaluation: The goal of our evaluation was to determine effectiveness and efficiency of the query construction with respect to accuracy and runtime. we measured the accuracy of the query construction in terms of precision and recall. The results of the experimental study show high accuracy on our benchmarks. In addition, in other to speedup runtime, we implement parallelization.

Results: The output of this algorithm is a set of graph templates. Each graph template represents a comprehensive set of query graphs, which are isomorphic regarding edges.

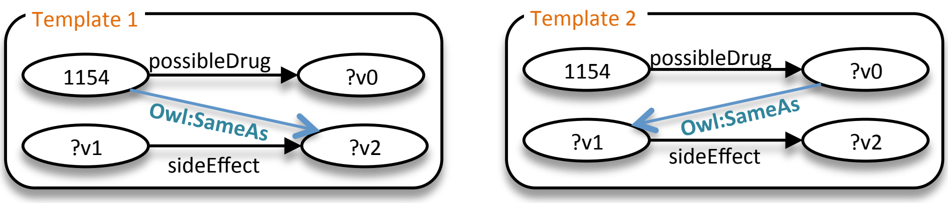


Figure 1.8: Generated query graph templates.

1.4.4 Data Fusion on Linked Data

The main challenges associated with data fusion on Linked Data are 1. How to disambiguate resources retrieved from different datasets, and 2. How to generate a federated formal query using resources from different datasets. This part of work has been published in [1, 6].

Methodology: Regarding the first challenge, our approach resembles a horizontal search, where query segments derived from an input query are matched against all available datasets. Then, we extend the Hidden Markov Model approach for disambiguating resources from different datasets. As an example of this extension, we mention the extension of the state space with reasoning. It includes resources inferred from lightweight `owl:sameAs` reasoning. Consequently, for extending the state space, for each state representing a resource `x` we just include states for all resources `y`, which are in an `owl:sameAs` relation with `x`. With respect to the second challenge, we construct a formal query (expressed in SPARQL) using the disambiguated matches by traversing links in the underlying datasets. By taking links between the matched resources (including `owl:sameAs` links) into account we obtain the *minimum spanning graph* covering all matches in the different datasets.

Example 1.7 Let us consider the input query: `Which are the drugs whose side effects are associated with the gene TRPM6?`. According to our approach, the following federated SPARQL query (on two datasets, i.e. Diseasome and sider) is generated.

```
SELECT ?v2 WHERE {
  ?v0  diseasome:associatedGene  diseasome:TRPM6 .
  ?v0  owl:sameAs                ?v1             .
  ?v2  sider:sideEffect          ?v1             . }
```

Evaluation: The goal of our evaluation was to determine effectiveness and efficiency of (1) the resource disambiguation and (2) the query construction on interlinked datasets with respect to accuracy and runtime. We measure the effectiveness of our resource disambiguation approach using the *Mean Reciprocal Rank* (MRR) and the accuracy of the query construction in terms of precision and recall. Our underlying knowledge base is life-science benchmark. We study the runtime in its mono-core and parallel implementations and draw preliminary conclusions on the scalability of keyword search on Linked Data.

Results: The result of evaluation shows the effectiveness as well as scalability of this approach. We are able to answer queries on distributed sources. The output is a federated query connecting resources from different datasets.

1.4.5 Benchmark Creation

A significant contribution of this thesis is providing two benchmarks to the community of question answering systems. We developed two benchmarks for two different tasks. First one is used for federated queries over Linked Data task. The second one is used for query expansion task.

Since there was no benchmark for federated queries over Linked Data, we created a benchmark consisting of 25 queries (natural language queries and their equivalent SPARQL queries) on the three interlinked datasets Drugbank, Sider and Diseasome[43] [1, 6]. Currently, our created benchmark as an standard benchmark is contributed in QALD-4[44] campaign. QALD-4 is the fourth in a series of evaluation campaigns on multilingual question answering over linked data.

Moreover, since there is no benchmark for query expansion tasks over Linked Data we created one benchmark dataset [7]. This benchmark contains 37 keyword queries obtained from the *QALD-1* and *QALD-2* benchmarks[45]. The *QALD-1* and *QALD-2* benchmarks are essentially tailored towards comparing question answering systems based on natural language queries. We extracted all those keywords contained in the natural language queries requiring expansion for matching to the target knowledge base resource. An example is the keywords `wife` and `husband` which should be matched to `dbpedia-owl:spouse`.

1.4.6 RDF Data Slicing

Motivation: An increasing amount of structured data is being published on the Web as Linked Open Data (LOD). Yet, consuming and using Linked Open Data within an organization is still a substantial challenge because many of the LOD datasets are quite large. In this work, we focus on the selection and extraction processes. Selection comprises the definition and specification of a relevant fragment of a dataset, which is envisioned to be used internally by a consuming organization. Extraction processes the dataset dump and extracts the relevant fragment.

Methodology: We devise a fragment of SPARQL dubbed SliceSPARQL, which enables the selection of well-defined slices of datasets fulfilling typical information needs. SliceSPARQL supports graph patterns for which each connected subgraph pattern involves a maximum of one variable or IRI in its join conditions. This restriction guarantees the efficient processing of the query against a sequential dataset dump stream [3, 8].

Evaluation: The goal of our evaluation was to determine: 1. How efficient is the slicing approach for various queries? 2. How does the slicing scale for datasets of different size? 3. How does our approach compare to the traditional approach (i.e. loading complete dumps into the triple store and extracting by querying).

Results: As a result, our evaluation shows that dataset slices can be generated an order of magnitude faster than by using the conventional approach of loading the whole dataset into a triple store and retrieving the slice by executing the query against the SPARQL endpoint of triple stores.

1.5 Conclusion and Future Work

Massive amount of structured data, which are distributed and interlinked has been published on Linked Data. Yet, retrieving data from such precious interlinked knowledge bases is a challenge. Thus, a new generation of search approaches is demanding to enable common users to easily but more semantically retrieve knowledge. The aim of this thesis was to research and address challenges in the way of a semantic search system on Linked Data. Such a system lies on a simple interface (i.e. textual query) similar to traditional search engines, but it takes the structure of data into account, therefore, retrieval of data is more semantic-oriented. During our research, we tackled a number of existing challenges (i.e. query

[43] The benchmark queries are available at `http://wiki.aksw.org/Projects/lodquery`
[44] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`
[45] `http://www.sc.cit-ec.uni-bielefeld.de/qald-`*n* for $n = 1, 2$.

segmentation and disambiguation, query expansion, formal query construction and data fusion on Linked Data). We summarize our contributions as follows:

1. A novel approach for determining the most suitable resources for a user-supplied query from different datasets (disambiguation approach). We employed a Hidden Markov Model, whose parameters were bootstrapped with different distribution functions.

2. A novel method for constructing federated formal queries using the disambiguated resources and leveraging the linking structure of the underlying datasets. This approach essentially relies on a combination of domain and range inference as well as a link traversal method for constructing a connected graph, which ultimately renders a corresponding SPARQL query.

3. Regarding the problem of vocabulary mismatch, our contribution is divided into two parts: First, we introduce a number of new query expansion features based on semantic and linguistic inferencing over Linked Data. We evaluate the effectiveness of each feature individually as well as their combinations, employing Support Vector Machines and Decision Trees. Second, we propose a novel method for automatic query expansion, which employs a Hidden Markov Model to obtain the optimal tuples of derived words.

4. We provide two benchmarks for two different tasks to the community of question answering systems. The first one is used for the task of question answering on interlinked datasets (i.e. federated queries over Linked Data). The second one is used for the vocabulary mismatch task.

In each part, we evaluate the accuracy of our approach using measures like mean reciprocal rank, precision, recall, and F-measure. The results of our accuracy evaluation demonstrate the effectiveness of our approach. Moreover, we study the runtime of our approach in its sequential as well as parallel implementations and draw conclusions on the scalability of our approach on Linked Data.

Our proposed method for query segmentation and resource disambiguation as well as query expansion achieved significant accuracy. Nevertheless, there is still room for further improvements. Furthermore, there are challenges which have not been addressed yet. Below, we briefly mention research areas in which we can extend the work.

1. Our proposed methods for query segmentation and resource disambiguation as well as query expansion achieved significant accuracy. But its parameters were estimated using bootstrapping. We ponder that the accuracy can be enhanced further using a learning approach. To employ a learning approach, we need large enough benchmarks as training and test datasets.

2. Our method for query construction is limited to generating only conjunctive queries, we may extend this method to be able to generate quantifiers, comparative as well as superlative clauses.

3. With respect to scalability, we must apply our approach on a bigger number of interlinked datasets in order to figure out upcoming challenges.

4. Query cleaning is a challenge that was not addressed in this thesis; thus it is one of our main focuses in future.

Apart from the above plans, we can extend our system to interact with traditional search engines. This is because traditional search engines retrieve data from rich textual content on the Web. Researchers separately have investigated information retrieval from structured and unstructured data. The envisioned plan for future can be defined as a hybrid search which takes both the unstructured as well as structured data into account. Therefore, we can take advantage of the high amount of data, which is unstructured. In other words, we aim at promoting search by a mutual benefit from both types of data (i.e. structured and unstructured). With this respect, the following challenges are raised:

1. How can we retrieve data using an approach that combines schema-unaware techniques (i.e. information retrieval techniques) and schema-aware techniques (NLP as well as inference techniques)?

2. How can structured data (resp. unstructured data) support data retrieval on unstructured data (resp. structured)?

CHAPTER 2

Generating SPARQL Queries Using Templates[1]

Abstract: The search for information on the Web of Data is becoming increasingly difficult due to its considerable growth. Especially novice users need to acquire both knowledge about the underlying ontology structure and proficiency in formulating formal queries (e. g. SPARQL queries) to retrieve information from Linked Data sources. So as to simplify and automate the querying and retrieval of information from such sources, this paper presents an approach for constructing SPARQL queries based on user-supplied keywords. Our approach utilizes a set of predefined basic graph pattern templates for generating adequate interpretations of user queries. This is achieved by obtaining ranked lists of candidate resource identifiers for the supplied keywords and then injecting these identifiers into suitable positions in the graph pattern templates. The main advantages of our approach are that it is completely agnostic of the underlying knowledge base and ontology schema, that it scales to large knowledge bases and is simple to use. We evaluate all 17 possible valid graph pattern templates by measuring their precision and recall on 53 queries against DBpedia. Our results show that 8 of these basic graph pattern templates return results with a precision above 70%. Our approach is implemented as a Web search interface and performs sufficiently fast to provide answers within an acceptable time frame even when used on large knowledge bases.

2.1 Introduction

Google[2] is the most widely used search engine in retrieving information from documents on the Web. Recently, *Google* has extended its functionality so as to provide direct answers to queries which match certain templates, e.g., *"Capital of Spain"*. However, web search engines still lack the ability to answer complex queries or consolidate information from different resources (i.e. integration of, and reasoning on, data on the Web.). These limitations are due to inherent unstructured nature of information in Web of Documents.

The Semantic Web introduces technologies (RDF, OWL, SKOS, SPARQL, etc.) for publishing machine-readable formats of information[3]. The heart of the technologies behind the Semantic Web is

[1] Corresponding publication is: Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler, "Generating SPARQL queries Using Templates", *Web Intelligence and Agent Systems Journal* 11.3 (2013) pp. 283–295

[2] `http://www.google.com`

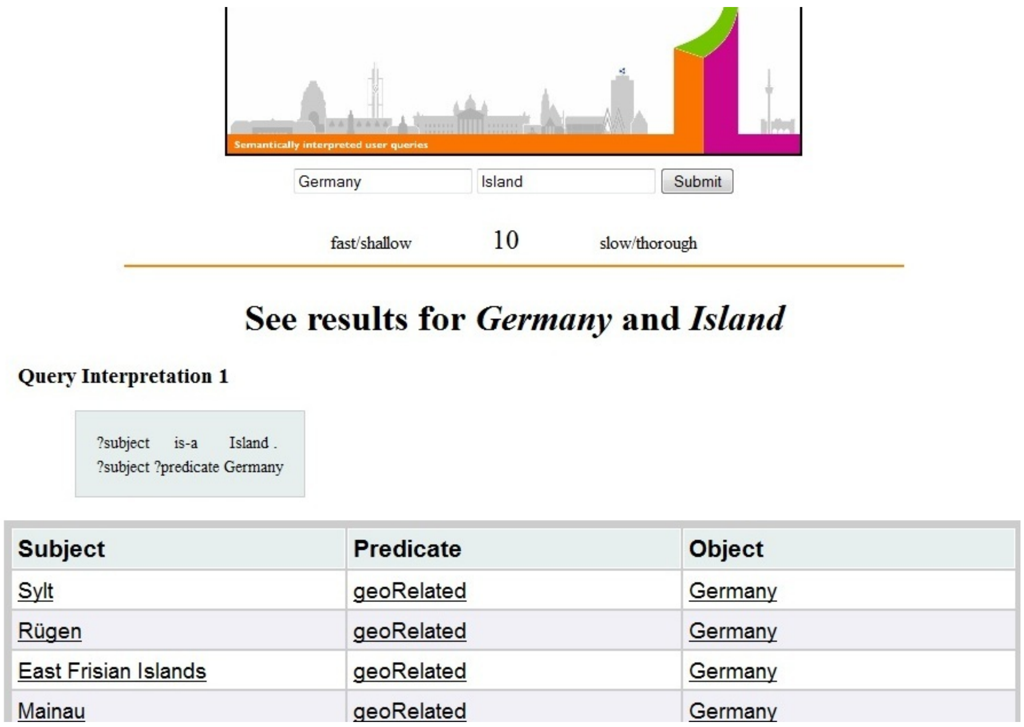
[3] `http://www.w3.org/TR/rdf11-concepts/`

the Resource Description Framework (RDF). RDF uses URIs[4] and IRIs[5] to refer to entities (an entity can be every thing) and is used by Semantic Web tools and frameworks to publish structured data whose meaning is defined in ontologies described in RDF Schema(RDFS) or the Web Ontology Language (OWL). In addition to the publishing of structured data, RDF allows the interlinking and merging of data across the Web [13, 14]. As a result of the Semantic Web idea and more importantly the existence of large amounts of structured data distributed across the Web, the idea of the Web of Data emerged. The Web of Data refers to the set of knowledge bases published according to the Linked Data principles, i.e., a set of best practices for publishing and connecting structured data on the Web [16].

Since its creation in 2007, the Linked Data Web has been growing at an astounding rate. Currently, it amounts to more than 31 Billion triples[6]. The mere amount of data contained therein poses an important challenge as to how to query this amount of knowledge. In fact, it is increasingly difficult for end users to find the information they are looking for. Services such as *Sindice* [26], *Sig.ma* [27], *Swoogle* [28] or *Watson* [29] offer simple search services[7], but are either restricted to the retrieval of single RDF documents or in the case of Sig.ma to the retrieval of information about a single entity from different sources. Some services[8] on the other hand load the complete the Data Web into a large triple store cluster and enable issuing SPARQL queries on top of it. However, in order to express their information needs in terms of SPARQL queries, users have to (a) understand the SPARQL concepts, (b) understand the SPARQL syntax (in absence of a visual query builder) and (c) know what information structures are actually available in order to formulate queries that also return results. To enable lay users to access the Data Web, it becomes necessary to *simplify the access to the Data Web* by providing search interfaces that resemble the search interfaces commonly used on the document-oriented Web. However, because queries based on natural language (NL) are inherently ambiguous, their precise interpretation is extremely challenging. While SPARQL queries permit to express unambiguously which entities and relations are relevant for the query, keyword-based search as implemented in current web search engines does not permit the explicit expression of relations. Services such as *PowerAqua*[9][30] demand from the user to enter a question in natural language, but this is often inconvenient because most users prefer to obtain information by the lowest number of keywords[10]. Another obstacle to the realization of this approach lies in the sheer size of the Data Web, which requires very efficient and scalable query processing algorithms.

In this paper, we propose a novel approach for generating SPARQL queries based on user-supplied keywords. Our approach presupposes the availability of background knowledge in the form of a set of Linked Data sources upon which the user wants her search to be carried out. Based on a set of user-supplied keywords, we first compute a list of candidate IRIs for each of the keywords issued by the user. In a second step, we restrict the set of valid IRIs to those which are related to each other via a link in the background knowledge. Finally, we use the filtered set of IRIs to generate SPARQL queries that aim to encompass the semantics of the query supplied by the user. We currently use DBpedia as background knowledge, but the approach is easily transferable to the whole Data Web. The basis for this claim is simple. Knowledge bases differ in the schemas they use and not in their format, which is always RDF. As our approach is clearly schema-agnostic, it can be deployed on any Linked Data source on the Web of data. Since the approach is based on simple operations, it can generate and execute SPARQL queries

[4] Uniform resource identifier, a string of characters used as unique identified for a resource.
[5] Internationalized Resource Identifier, a generalization of URIs.
[6] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/` (Februiary 25th, 2013)
[7] These systems are available at: `http://sindice.com`, `http://sig.ma`, `http://swoogle.umbc.edu`, `http://kmi-web05.open.ac.uk/WatsonWUI`
[8] For example `http://lod.openlinksw.com`
[9] `http://poweraqua.open.ac.uk:8080/poweraqua2`
[10] `http://www.keyworddiscovery.com/keyword-stats.html`



See results for *Germany* and *Island*

Query Interpretation 1

```
?subject    is-a      Island .
?subject ?predicate Germany
```

Subject	Predicate	Object
Sylt	geoRelated	Germany
Rügen	geoRelated	Germany
East Frisian Islands	geoRelated	Germany
Mainau	geoRelated	Germany

Figure 2.1: Example query in GUI available at `lod-query.aksw.org` for the search keywords *germany* and *island*.

very efficiently. Another advantage of this approach is that it is completely agnostic of the underlying knowledge base as well as its ontology schema.

This paper is organized as follows: In section 2.2 we present the background definitions and an overview of the approach along with our method for choosing candidate IRIs. The subsequent section introduces all possible graph pattern templates for pairs of IRIs. In section 2.4 we describe our approach to the construction of SPARQL queries based on graph pattern templates. We elaborate on our experimental setup and the selection of graph patterns, as well as analyze our results in section 3.6. The related work is reviewed in section 2.6. We close with concluding remarks and an outlook on future work in the last section.

2.2 Approach

2.2.1 Preliminaries

The basic assumption underlying our approach is that natural language queries cannot always be converted into formal queries automatically. This is due to the meaning of some of the query elements being either unknown, ambiguous, or implicit. For example, in the query 'Which are the islands in Germany?', the relation between *Germany* and *islands* is indicated by *are*, but the precise relationship is *are located in*. The problem of mapping a user query to a formal query gets even more complex when the user uses keywords instead complete natural language queries, because even more information is omitted. In addition, experience with classical search engines shows that users prefer to enter the lowest possible number of keywords in order to retrieve information related to their query. For example, the query mentioned above would be naturally expressed with the keywords *Germany* and *islands*.

In the context of the Semantic Web, the expected answer to a query is usually a set of RDF resources linked by certain relations (representing a connected graph). Consequently, the second assumption underlying our approach is that user-supplied keywords must play the role of anchor points (i.e., matched nodes or edges of the RDF graph) that are to be used to retrieve knowledge from the background knowledge via some form of bootstrapping. We illustrate the difficulties of the bootstrapping processing with the following examples:

Example 2.8 Consider two keywords *"Germany"* and *"capital"* which a user uses to search for an RDF graph containing *Berlin* as the answer. The corresponding SPARQL query is[11]:

```
SELECT * WHERE {
  dbr:Germany  dbp:capital  ?var .
}
```

and the desired answer is shown as an RDF triple as follows:

```
dbr:Germany dbp:capital dbr:Berlin .
```

Example 2.9 Consider two keywords *"Germany"* and *"island"* used with the intention to search for the list of *Germany's islands*. The suitable SPARQL query is:

```
SELECT * WHERE {
  ?island       a      dbo:Island  .
  ?island       ?p     dbp:Germany .
}
```

Some desired answers to be retrieved are:

```
1:db:Sylt      a            dbo:Island  .
  db:Sylt      dbp:country  dbr:Germany .
2:db:Vilm      a            dbo:Island  .
  db:Vilm      dbp:country  dbr:Germany .
3:db:Mainau    a            dbo:Island  .
  db:Mainau    dbp:country  dbr:Germany .
```

Here, we encounter two issues. First, we need to find a set of IRIs (anchor points) corresponding to each keyword. Second, we have to construct suitable triple patterns based on the anchor points extracted previously so as to retrieve appropriate data. These goals are achieved by the approach presented in the following.

2.2.2 Terminology and Definitions

We call an IRI matching to a keyword an *anchor point*. The process of finding a sub-graph covering all anchor points is called *induction*. Note, that most semantic search approaches (e.g., [30–34]) perform induction first on the ontology level to extract appropriate graph pattern templates, and then apply those templates to the instance level. We, however, do not separate *induction* in the ontology level from the

[11] for the prefixes see http://prefix.cc/[dbpedia | dbp | dbo | dbr]

instance level since ontology statements are usually available either in the knowledge base or via Linked Data de-referencing as RDF triples. Consequently, instances and ontology statements are connected based on `rdf:type` properties, allowing our *induction* not to have to separate between ontology and instance knowledge. Formally, we base our approach on the following definitions:

Definition 1 (Keyword set). *We define the set of user-supplied keywords as* $K = \{k_1, k_2, ..., k_n\}$.

Definition 2 (Knowledge base signature). *The knowledge base signature* KBS *is represented by* $KBS = (C, I, P)$ *, where* C *denotes the set of classes,* I *denotes the instances of these classes and* P *denotes the set of properties used in the relationships between classes or instances (also including datatype properties).*

Definition 3 (Connected query result). *A single connected query result denoted* $R = \{(s, p, o) | (s, p, o)$ *a triple*$\}$, *consists of a set of triples which are connected through common subjects or objects, i.e.:*

$$(|R| \leq 1) \vee (\forall (s_1, p_1, o_1) \in R : \exists (s_2, p_2, o_2) \in R|$$

$$(s_2 = s_1 \vee s_2 = o_1 \vee o_2 = o_1 \vee o_2 = s_1))$$

These sets of triples express sentences which represent a sort of integrated information around the user keywords.

2.2.3 Overview

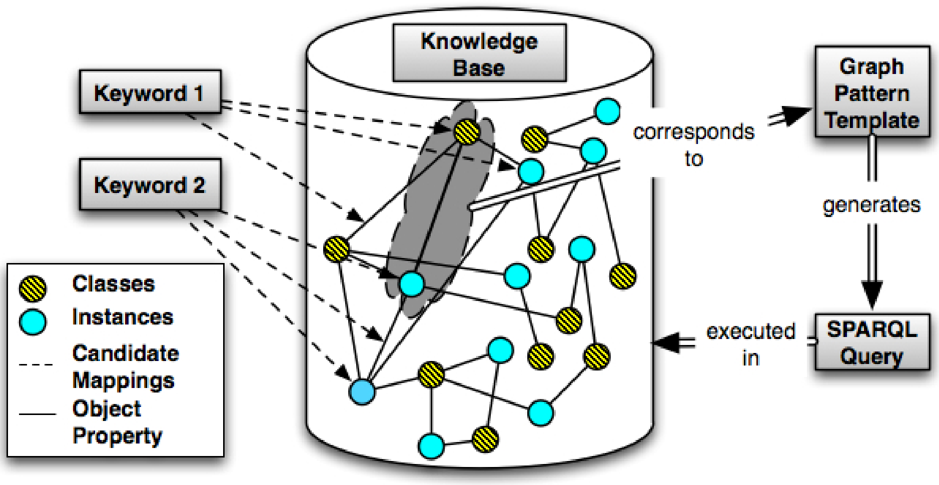


Figure 2.2: Overview of the proposed method.

Figure 2.2 shows an overview of our approach. Our approach firstly retrieves relevant IRIs related to each user-supplied keyword from the underlying knowledge base and secondly injects them to a series of graph pattern templates for constructing formal queries. So as to find these relevant IRIs, the following two steps are carried out:

2.2.4 Mapping Keywords to IRIs

The goal of this function is the retrieval of entities that match with the user-supplied keywords. Matching entities and keywords is carried out by applying a string similarity function on the keywords

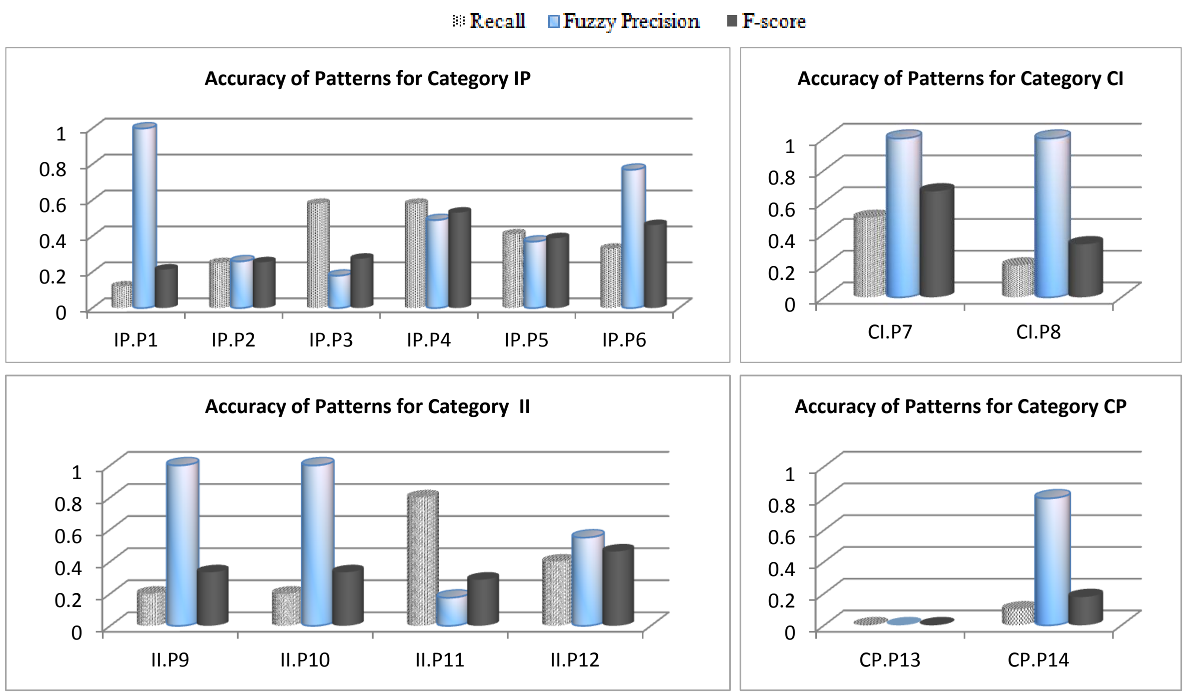


Figure 2.3: Accuracy of each categorized graph pattern.

and the label properties of all entities in the knowledge base. This similarity evaluation is carried out on all types of entities (i.e., classes, properties and instances). As a result, for each keyword, we retrieve a list of IRI candidates, i.e. *anchor points*.

Definition 4 (Mapping function)**.** *Let K be the set of user-supplied keywords. The mapping function $M : K \to 2^{C \cup I \cup P}$ applies the sub-string similarity measure on each $k_i \in K$ and on the* `rdfs:label` *of all IRIs in our underlying knowledge base and returns the set $AP_{k_i} \subseteq C \cup I \cup P$ (where C, I and P are the sets of classes, instances and properties contained in the knowledge base respectively), whose labels contain k_i as a sub-string or are equivalent to k_i.*

Ranking and Selecting Anchor Points

This step aims at excluding anchor points which are probably unrelated to any interpretation of the user keyword; thereby reducing the potentially high number of anchor points to a minimum. This reduction is carried out by applying a ranking method over the string similarity score and the connectivity degree of the previously detected IRIs in each AP_{k_i}. For each $u \in AP_{k_i}$ a *specificity score*, denoted by S, is defined based on two parameters, i. e. a *string similarity score* and a *connectivity degree*.

The string similarity score σ calculates the similarity of the `rdfs:label` of $u \in AP_{k_i}$ and of the keyword k_i by measuring the normalized edit distance between $u_{rdfs:label}$ and k_i. As the query we use for retrieving IRIs guarantees that k_i is a substring of u_{label}, computing the edit distance between these two strings is equivalent to computing the difference in their length. Note that edit distance is the most common string similarity metric for typos. We normalize the *string similarity score* of each label by using the *max-min normalization method* to compute similarity values between 0 and 1. Consequently,

$\sigma(u_{label}, k_i) = 1$ means that the two strings are equal. Formally,

$$\sigma(u_{label}, k_i) = 1 - \frac{|u_{label}| - \min\limits_{v \in AP(k_i)} |v_{label}|}{\max\limits_{v \in AP(k_i)} |v_{label}| - |k_i|} \tag{2.1}$$

We also compute a simplified approximation of the *connectivity degree* $CD(u)$ for each $u \in AP_{k_i}$ by counting how often u occurs in the triples of the knowledge base. It is important to note that IRIs with type `class` and `property` have higher CD values. In DBpedia, for example, classes have an average connectivity degree of 14,022, while properties have in average 1,243 and instances 37. Since connectivity degree values are exponential, we use the **logarithm** of these values to compute $S(u)$. The intuition is that both *string similarity score* and *connectivity degree* has a direct effect on the specificity S of each u. Therefore, the specifity for each $u \in AP(k_i)$ is finally calculated as the multiplication of both of these parameters as follows:

$$S(u) = \sigma(u_{label}, k_i) \times \log(CD(u)) \tag{2.2}$$

Definition 5 (Ranking and selection function)**.** *The ranking and selection function RS maps AP_{k_i} to the set U_{k_i} as top-10 of the IRIs contained in AP_{k_i} sorted in descending order based on $S(u)$ where $u \in AP_{k_i}$.*

2.3 Graph Pattern Templates

Throughout the paper, we use the standard notions of the RDF[12] and SPARQL[13] specifications, such as *graph pattern*, *triple pattern* and *RDF graph*. The SPARQL queries generated with our approach are a restricted kind of SPARQL queries, since they use only *basic graph patterns* without blank nodes. We analysed 1,000 distinct queries from the query log of the public DBpedia endpoint[14] and learned that the number of IRIs is usually larger than the number of triple patterns occurring in the query. As a consequence of this finding we decided to assume graph patterns for generating SPARQL queries for two user-supplied keywords to consist of either one or two triple patterns.

Definition 6 (Graph pattern template)**.** *Let H be a set of placeholders and V be a set of variable identifiers being disjoint from each other and from $C \cup I \cup P$. A graph pattern template is defined as $GPT = \{(s, p, o) | (s \in V \cup H) \wedge (p \in V \cup H) \wedge (o \in V \cup H)\}$ that contains exactly two placeholders. Two triple patterns being part of the same graph pattern template have to share a common subject or object. In our triple pattern templates, a placeholder can stand either for a property (when occurring in the predicate position), an instance (when occurring at subject or object position) or a class (when occurring at the object position) depending on its position. After replacing the placeholders in a graph pattern template with the detected IRIs, a graph pattern with triple patterns of the form $(V \cup I) \times (V \cup P) \times (V \cup C \cup I)$ is obtained.*

Note that our notion of graph pattern templates is a slight restriction of the SPARQL basic graph patterns in the general case, since our definition does not consider blank nodes and restricts the set of possible IRIs at a certain position in the triple pattern. Definition 6 leads to the 17 possible graph pattern templates shown in Table 2.1. In this table, we subdivided the patterns in different categories, depending

[12] http://www.w3.org/TR/rdf-schema/
[13] http://www.w3.org/TR/rdf-sparql-query/
[14] The DBpedia SPARQL endpoint is available at: `http://dbpedia.org/sparql/` and the query log excerpt at: `ftp://download.openlinksw.com/support/dbpedia/`.

Category	Patterns	Pattern Schema
Instance-Property (IP)	**IP.P1**	$(s, p, ?o)$
	IP.P2	$(?s, p, o)$
	IP.P3	$(?s_1, ?p_1, o_1)(?s_1, p_2, ?o_2)$
	IP.P4	$(?s_1, ?p_1, o_1)(?o_2, p_2, ?s_1)$
	IP.P5	$(s_1, ?p_1, ?o_1)(?s_2, p_2, ?o_1)$
	IP.P6	$(s_1, ?p_1, ?o_1)(?o_1, p_2, ?o_2)$
Class-Instance (CI)	**CI.P7**	$(?s_1, a, c)(?s_1, ?p_1, o_1)$
	CI.P8	$(?s_1, a, c)(s_2, ?p_1, ?s_1)$
Instance-Instance (II)	**II.P9**	$(s, ?p, o)$
	II.P10	$(s, ?p_1, ?x)(?x, ?p_2, o)$
	II.P11	$(s_1, ?p_1, ?x)(s_2, ?p_2, ?x)$
	II.P12	$(?s, ?p_1, o_1)(?s, ?p_2, o_2)$
Class-Property (CP)	CP.P13	$(?s, a, c)(?s, p, ?o)$
	CP.P14	$(?s, a, c)(?x, p, ?s)$
Property-Property (PP)	PP.P15	$(?s, p_1, ?x)(?x, p_2, ?o)$
	PP.P16	$(?s_1, p_1, ?o)(?s_2, p_2, ?o)$
	PP.P17	$(?s, p_1, ?o_1)(?s, p_2, ?o_2)$

Table 2.1: Categorization of all possible graph pattern templates for each typed pair of placeholders.

Category	Patterns	Pattern Schema
Instance-Property(IP)	IP.P1	$(s, p, ?o)$
	IP.P4	$(?s_1, ?p_1, o_1)(?o_2, p_2, ?s_1)$
	IP.P6	$(s_1, ?p_1, ?o_1)(?o_1, p_2, ?o_2)$
Class-Instance(CI)	CI.P7	$(?s_1, a, c)(?s_1, ?p_1, o_1)$
	CI.P8	$(?s_1, a, c)(s_2, ?p_1, ?s_1)$
Instance-Instance(II)	II.P9	$(s, ?p, o)$
	II.P10	$(s, ?p_1, ?x)(?x, ?p_2, o)$
Class-Property(CP)	CP.P14	$(?s, a, c)(?x, p, ?s)$
Property-Property(PP)	-	-

Table 2.2: Appropriate identified graph pattern templates.

on whether they map instances to instances, classes to instances etc. Symbols preceded by question marks denote variables while symbols without question marks are placeholders which will be replaced by IRIs referring to the identified anchor points.

Example 2.10 After applying the mapping and ranking functions to the user keywords (from Example 9), we obtain two IRIs, i.e. `http://dbpedia.org/ontology/Island` with the type *class* and `http://dbpedia.org/resource/Germany` with the type *instance*. The possible graph pattern templates for these two IRIs are:

1. (?island, a, dbo:Island), (?island, ?p, dbr:Germany)
2. (?island, a, dbo:Island), (dbr:Germany, ?p, ?island)

As detailed in section 3.6, we performed an accuracy study on all combinatorial possible graph pattern templates. This study showed that the patterns contained in Table 2.2 limit the search space (thus leading to more efficiency) without reducing the accuracy of our approach significantly. Consequently, we only considered these patterns during the SPARQL-query generation process described below.

2.4 SPARQL Query Generation

Algorithm 17 outlines the procedure for generating SPARQL queries based on the graph pattern templates shown in Table 2.2. After selecting the top ranked IRIs based on Definition 5, according to the type of each pair of IRIs issued from the cross-product of U_{k_i}, a set of suitable graph pattern templates is selected from Table 2.2 for generating SPARQL queries.

> **Example 2.11** For the pair of IRIs `http://dbpedia.org/resource/Germany` and `http://dbpedia.org/ontology/Island`, our algorithm would generate the following two queries:
>
> 1.
> ```
> SELECT * WHERE {
>   ?island  a   dbo:Island .
>   ?island  ?p  dbr:Germany . }
> ```
> 2.
> ```
> SELECT * WHERE {
>   ?island     a   dbo:Island .
>   dbr:Germany ?p  ?island . }
> ```

The results of this algorithm are the output of our approach. To validate the approach, we implemented as a Java Web application which is publicly available at: `http://lod-query.aksw.org`. A screenshot of the search results is shown in Figure 2.1. The whole query interpretation and processing is performed typically on average in 15 seconds (while first results are already obtained after one second) when using DBpedia as knowledge base.

```
Data : $K$ Keyword Set, knowledge base $KB$
Result : A set of connected query results
foreach keyword $k_i$ do
    retrieve $AP_{k_i}$;
    sort $AP_{K_i}$;
    $RS(AP_{K_i})$ = top-10 ranked IRIs from $AP_{K_i}$;
end
foreach $u \in RS(AP_{K_i})$ & $u' \in RS(AP_{K_j})$ do
    switch Category of $u, u'$ do
    endsw
    case Class-Instance
        query(CI.P7,$u, u'$);
        query(CI.P8,$u, u'$);
    case Class-Property
        query(CP.P14,$u, u'$);
    case Instance-Instance
        query(II.P9,$u, u'$);
        query(II.P10,$u, u'$);
end
```

Algorithmus 1 : Query generation algorithm. The function `query` constructs the query based on the query pattern given as first argument and the entity identifier to placeholder mapping supplied as 2nd and 3rd argument.

2.5 Evaluation

This section is divided to four parts. First, we introduce the accuracy metrics used for evaluation. Second, we outline the results of an accuracy study on all valid graph pattern templates introduced in

Table 2.1 with the aim of selecting those templates that lead to a high accuracy. Third, we evaluate our whole application by using the metrics presented in the following subsection. In the last section, a comparison study based on relevance feedback is presented.

2.5.1 Accuracy Metrics

Since the user's intention in keyword-based search is ambiguous, judging the correctness of the retrieved answers is a challenging task. Let us consider the following example:

Example 2.12 Given the keywords `France` and `President` the following RDF graphs (i.e. *answers*) are presented to the user:

```
1. Nicolas_Sarkozy  nationality  France    .
   Nicolas_Sarkozy  a         President    .

2. Felix_Faure      birthplace   France    .
   Felix_Faure      a         President    .

3. Yasser_Arafat    deathplace   France    .
   Yasser_Arafat    a         President    .
...
```

The input of the user can be interpreted in at least two ways:

1. Who is the current president of France?
2. Who are the people that have ever been presidents of France?

Depending on the meaning intended by the users, these patterns can be considered as being accurate or not. If the second interpretation is correct, then *Felix Faure*, who was the president of France from 1895 to 1899, is a correct answer, else it is not. We only consider those answers correct that meet our original intention whereas all other ones are considered incorrect. According to this the correct answers are (1) and (2). However, among the correct answers note the difference in the involved predicates, namely *birthplace* and *nationality*. An observation is that we can draw a distinction between whether an answer contains statements relevant to our search intention and whether these statements are the preferred ones. We will measure the preference of an answer based on the occurring RDF terms. RDF terms (short *terms*) comprise each individual subject, object or predicate in the triples of the answer. In our example, we prefer *nationality* over *birthplace* because if a person is born in one country may be president in a different one, it is very unlikely that a president has a different nationality than the country he is president in.

Therefore, besides distinguishing between answers related to different interpretations, we also differentiate between pure answers (just containing preferred terms) and those which contain some impurity. In fact, the correctness of an answer is not a bivalent value but based on the user's perception. Rather, it may vary between completely irrelevant and exactly correct. In essence, within this evaluation, we address two main questions: 1) For how many of the keyword queries do the templates yield answers at all with respect to the original intention? 2) If answers are returned, how correct are they?

Therefore, we introduce the *Correctness Rate (CR)* as a measure for the preference of certain RDF terms. This metric allows a user to rate the correctness of each individual answer based on its own perception.

Definition 7 (Correctness rate)**.** *For an individual answer a for a query q, we define $CR_q(a)$ as the fraction of correct (preferred) RDF terms occurring in it.*

$$CR_q(a) = \frac{|correct\ terms|}{|total\ terms|}$$

Based on the CR for individual answers, we can derive the average CR (ACR) for a set of answers:

Definition 8 (Average CR)**.** *For a given set of answers A of a query q, we define $ACR_q(A)$ as the arithmetic mean of the CRs of its individual answers.*

$$ACR_q(A) = \frac{1}{|A|} * \sum_{a \in A} CR_q(a)$$

The ACR is the basis for the *fuzzy precision* metric (FP), which measures the overall correctness of a template's corresponding answers A_q with respect to a set of keyword queries Q.

$$FP = \frac{\sum_{q \in Q} ACR_q(A_q)}{|queries\ with\ answers|}$$

By using the fuzzy precision, we can now measure the quality of the results returned by each individual graph pattern template. The rationale between our measurements is that a template is not required to contribute answers to the set of all answers (as other templates of the same corresponding category may compensate for that. However, if answers are provided, they are subject to the correctness evaluation.

We also measured the recall as the fraction of keyword queries for which answers were found:

$$Recall = \frac{|queries\ with\ answers|}{|total\ queries|}$$

Finally, we use the following definition of the F-Score [35]:

$$F = 2 * \frac{FP * R}{FP + R}$$

2.5.2 Accuracy Evaluation of Possible Graph Pattern Templates

Since we are interested in using those graph pattern templates which typically result in precise answers with respect to the user intention of keywords, we evaluated the accuracy of each graph pattern template by running a SPARQL query containing each individual graph pattern template introduced in Table 2.1 by injecting a series of IRI pairs. We selected 40 natural language queries[15] of the *TREC-9 - question answering track* from which we extracted the two main keywords conveying the general meaning. The selection was performed based on balancing between different query types (i.e. associations, similar instances and characteristics) and expressibility by two keywords. For example, the query *'How many people live in Chile?'* can be expressed by the keywords *Chile* and *population*. Thereafter, the mapping function was applied to these keywords and from the retrieved IRIs, the most suitable ones were *manually* selected and assigned to the related dataset with regard to their type. We used DBpedia 3.5.1 [23] as the underlying knowledge base. After preparing the datasets, we performed a series of SPARQL queries for each single graph pattern template over the corresponding dataset. The results of the SPARQL queries along with the keywords were shown to two evaluators to score the *CR* metric for each individual answer.

[15] Queries are available online at:`http://aksw.org/Projects/lodquery`.

After rating *CR* for all retrieved answers related to a graph pattern template, *fuzzy precision*, *recall* and *F − score* were computed. Figure 2.3 shows the accuracy of each graph pattern template based on these three metrics. In the category Property-Property, the number of retrieved answers for all graph pattern templates was zero.

Our results show that some pattern templates such as P1 in the Instance-Property category as well as P7 and P8 in the Instance-Class category have a high fuzzy precision while their recall is low. In the case of P11 from the Instance-Instance category we have a high recall while the fuzzy precision is low. Hence, this graph pattern template generates a large number of irrelevant answers.

We discarded all templates with a fuzzy precision of less than 0.5, resulting in an increase of the overall precision and only a small loss in recall. We monitored the *ACR* for a set of queries before and after the reduction of graph pattern templates in the category IP and II, because most reductions occurred there. In the category IP, all queries with *ACR* higher than 0.4 and in the category II with *ACR* higher than 0.6 were properly answered with the same accuracy. So, this reduction maintained precise results (i.e. high *ACR* value).

As an interpretation of graph pattern templates, we present different scenarios in which a user is interested in retrieving different kinds of information. This categorization is based on the matter of information which is retrieved from the knowledge base.

Finding special characteristics of an instance Data type properties which emanate from instances/classes to literals or simple types and also some kinds of object properties state characteristics of an entity and information around them. So, in the simplest case of a query, a user intends to retrieve specific information of an entity such as *"Population of Canada"* or *"Language of Malaysia"*. Since this information is explicit, the simple graph patterns IP.P1, IP.P4 and IP.P6 can be used for retrieving this kind of information.

Finding similar instances In this case, the user asks for a list of instances which have a specific characteristic in common. Examples for these type of queries are: *"Germany Island"* or *"Countries with English as official language"*. A possible graph structure capturing potential answers for this query type is depicted in Figure 2.4. It shows a set of instances from the same class which have a certain property in common. Graph pattern templates CI.P7, CI.P8, and CP.P14 retrieve this kind of information.

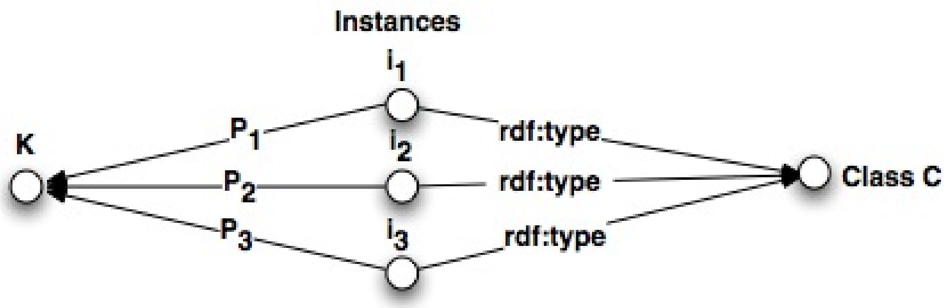


Figure 2.4: Similar instances with an instance in common.

Finding associations between instances Associations between instances in knowledge bases are defined as a sequence of properties and instances connecting two given instances (cf. Figure 2.5). Therefore, each association contains a set of instances and object properties connecting them which is the

Category	**Recall**	**Fuzzy precision**	**F-score**
General accuracy	0.625	0.724	0.670
Similar instances	0.700	0.735	0.717
Characteristics of an instance	0.625	0.700	0.660
Associations between instances	0.500	0.710	0.580

Table 2.3: Accuracy results.

purpose of the user query. As an example, the query *Volkswagen Porsche* can be used to find associations between the two car makers. The graph pattern templates II.P9 and II.P10 extract these associations.

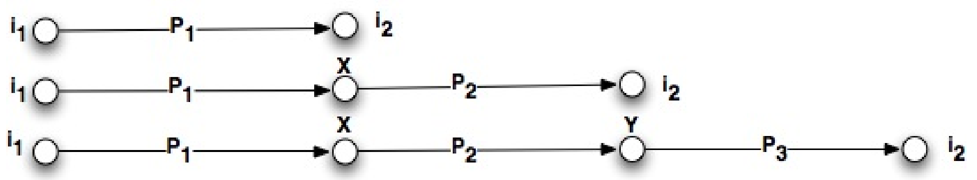


Figure 2.5: Associations between two instances.

2.5.3 Application Evaluation

In this step, we evaluated the approach based on the three previously defined metrics, i.e. fuzzy precision, recall and f-score. The experimental setup consisted of giving a novice user 40 queries from TREC 9, and asking him to run each of the queries against DBpedia using our prototype implementation. Then, for each single answer of a query, he assigned *CR* according to his own intention. Subsequently, fuzzy precision and recall were computed based on the user's ratings. Note that since hyperlinks among pages are inserted as `wikilink` in DBpedia and they do not convey special meaning between resources, we removed all triples containing the IRIs `http://dbpedia.org/property/wikilink`. Table 2.3 shows the evaluation results after running 40 queries against DBpedia. The overall precision of our system is 0.72.

Essentially, the accuracy of this method, specifically *recall*, does not depend on using suitable graph pattern templates, because on the one hand, the mapping approach for choosing relevant IRIs significantly influences the results, and on the other hand the quality of the data in DBpedia severely affects the accuracy. For example, the query *"Greece population"* returns the correct answer while the similar query *"Canada population"* led to no results.

In addition to the overall evaluation, in order to make a comparison between functionality of the approach for different types of queries (i.e. finding special characteristics of an instance, finding similar instances and finding associations between instances) the employed queries were categorized based on their type and a separate evaluation was computed for each type. Our evaluation in Table 2.3 shows the precision does not differ significantly for different types of queries, while the recall is type dependent. For instance, in the category *"similar instances"* the recall is significantly higher rather than in the category *"association between instances"*.

2.5.4 Comparative Study based on Relevance Feedback

In the relevance feedback evaluation, four users were asked to assess the quality of the results returned by several engines by assigning a relevance score to these results. We employed an explicit relevance feedback approach, in which the users could grade the output of the systems with values between 0 (not

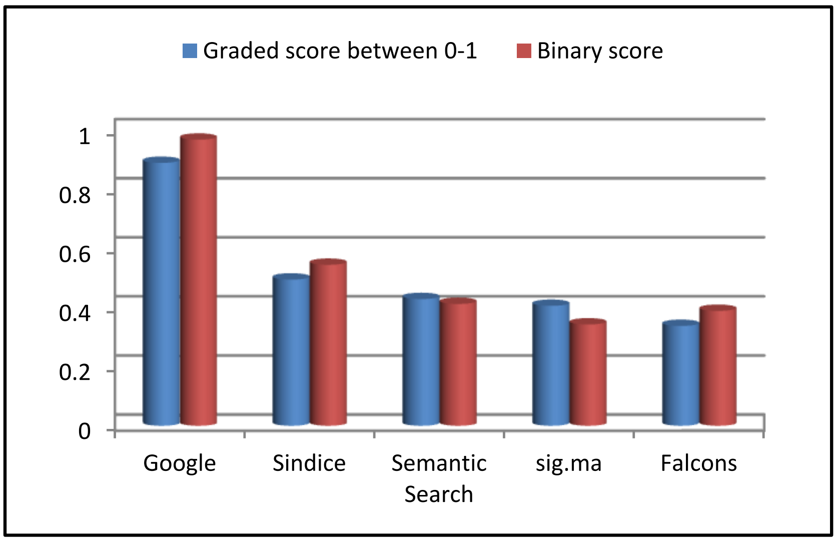


Figure 2.6: Relevance feedback comparison.

relevant) and 1 (very relevant) as well as with a binary score. The binary relevance feedback was meant to indicate whether or not the presented results were relevant for a given query, while the graded relevance feedback scored the relevance on the predefined scale. We compared our approach with a traditional Web search engine (Google) and three Semantic Web search engines (Sindice, Sig.ma and Falcon) on 40 queries that were selected from *Trec-9*[16]. In addition, the users were asked to evaluate the engines based on the top-10 results exclusively. Our search service currently is running on DBpedia data set, whereas other search services are running on the whole of Web (Web of Data) which probably comparison of the results is unfair. Therefore, to alleviate this problem, in the case of Google (resp. Sindice), user was requested to limit the search domain to Wikipedia (DBpedia). Figure 2.6 shows the average of relevance scores for both graded and binary scales. Alongside, Table 2.4 indicates the agreement rates among assessors for each search service. Google scores best and consistently returns as rated highly relevant results. Similarly, Sindice achieves a higher recall due to its larger index and is thus considered to be more relevant by our assessors. On average, our search service (Semantic search) is at the third position.

2.6 Related Work

Several information retrieval and question answering approaches for the Semantic Web have been developed over the past years. While several of these approaches are adaptation of document retrieval approaches for the Semantic Web, some approaches have been devised especially with RDF data as focus. In the following, we give an overview of these approaches.

[16] `http://aksw.org/Projects/lodquery/files?get=queries.xlsx`

Category	**Google**	**Sindice**	**Sem. S.**	**Sig.ma**	**Falcons**
4 agreements	0.8	0.325	0.475	0.275	0.425
3 agreements	0.2	0.475	0.3	0.475	0.525
2 agreements	-	0.2	0.225	0.25	0.05

Table 2.4: Percentage of agreement rates in relevance feedback study.

Ontology-based information retrieval: Approaches falling into this category annotate and index documents using a background ontology. The retrieval process is subsequently carried out by mapping user query terms onto these semantic document annotations. The approaches described in [36–38] are examples of this paradigm. All these approaches use background knowledge to enhance the retrieval accuracy. However, they do not utilize the background knowledge for semantically answering user queries.

Swoogle[17] [28] for example is a document-based search engine over ontologies. It enables users to query ontologies using a keyword-based paradigm. As the result, it represents documents which query-keywords occur somewhere in them. Swoogle crawls and indexes semantic web documents(SWD), i.e. documents annotated by RDF or OWL. The metadata around SWDs such as basic information about the SWDs as well as relations between them are extracted and indexed. Then, a ranking mechanism different from IR ranking methods is applied to assign a weight to each SWD. A weak point of this search engine is that relations between documents are limited to relations which explicitly has been stated. Similarly, *Watson*[18] [29] gives access to semantic web documents. In addition, it allows retrieving entities in documents. Moreover, Watson also provides APIs to external services. *Sindice*[19] [26] on the other hand sticks with the document-centric paradigm but integrates data from different sources and locates relevant sources for the data being queried. It benefits from an indexing approach developed especially for linked data and integrates reasoning, SPARQL endpoint indexing and the ability to index large repositories through the Sitemap extension.

In addition to document-centric approaches, entity-centric approaches have emerged over the past years. *Sig.Ma*[20] [27] for example uses Sindice in the background to present users with aggregated information about entities. It combines information about a single entity from different sources. Although it does not provide a high-quality disambiguation engine, its interface facilitates the manual disambiguation of entities to some extent by allowing users to easily reject irrelevant sources.

A lightweight data consolidation and reasoning approach is implemented by *SWSE*[21] [39], which also integrates entities from different sources. The data consolidation is based on two types of properties i.e., *owl:sameaAs* relating two equivalent entities and *owl:FunctionalProperty* defining a class of properties whose value uniquely identifies an entity. SWSE enables users to filter the resulting objects by specifying class expressions. In a similar fashion, *Falcons*[22] [40] implements an entity-centric approach to Semantic Web search. In addition to retrieving entities, it can find relations between entities as well as characteristics of entities. However, the basis for all these services are keyword indexing and retrieval relying on the matching user-issued keywords and indexed terms.

Ontology-based question answering: Approaches falling into this category take a natural language question or a keyword-based query and return matching knowledge fragments drawn from the knowledge base as the answer. Two main categories of approaches achieve this goal. The first category relies on using linguistic approaches for extracting complete triple-based patterns (including relations) from the user query and matching these triples to the underlying ontology. Examples of such approaches are those implemented by *AquaLog* [30] and *OntoNL* [31]. AquaLog is a domain-specific question answering system and it is the predecessor of PowerAqua which is a domain-agnostic system. *PowerAqua* [41]

[17] http://swoogle.umbc.edu
[18] http://kmi-web05.open.ac.uk/WatsonWUI/
[19] http://sindice.com/
[20] http://sig.ma/
[21] http://swse.deri.org
[22] http://ws.nju.edu.cn/falcons/objectsearch/

can automatically combine information from multiple ontologies at runtime. The input of this system is a natural language query and the output is a list of ontology entities that are relevant. PowerAqua lacks a deep linguistic analysis and can not handle complex queries. *Pythia* [42] is another question answering system that employs a deep linguistic analysis. It can handle linguistically complex questions, but it is highly dependent on a manually created lexicon. Therefore it fails to scale to datasets for which the lexicon was not designed. Pythia was recently used as kernel for *TBSL* [43], a more flexible question-answering system that combines Pythia's linguistic analysis and the *BOA framework* [44, 45] for detecting properties to natural language patterns. The second category of approaches aims at detecting entities in the user query and discovering relations between these entities by analyzing the knowledge base. In this second category of approaches, RDF data is regarded as a directed graph and relations among entities are found through sequences of links (e.g., using graph traversal). The graph traversal approaches mainly visits nodes using different methods of exploration such as breadth first search [33], bidirectional search [46], back tracking [47] and top-k exploration [48]. All these approaches start the exploration from a set of anchor points (nodes bound to the input IRIs) and search iteratively through the graph until they find a connecting node, i.e., a node that links the paths traversed from all of the anchor points. System examples for this second group are *KIM* [32], *OntoLook* [49] and [33, 34, 50, 51].

Sheth [52] introduced the term *semantic association* for describing meaningful and complex relations between entities. Our work differs from these approaches in that it is completely independent of the underlying schema. Furthermore, schema information is in our approach just *implicitly* taken into account, so a complex induction procedure is not required.

Keyword search on relational and XML data: Currently, keyword-based search is the most popular and convenient way for finding information on the Web [53]. Although the field of IR and database approximately initiated at the same time, keyword-based search was studied mainly in the area of IR; because users of databases are mainly experts who have knowledge about a specific language to interact with a database system. From the last decade, research on keyword search on relational and XML data attracted research interest. Recently, by emerging the tendency towards annotating web documents and publishing a vast amount of structured data, this model of search has obtained more focus. Essentially, current exposure of most Web users to keyword search, the large amount of research on the successful application of keyword-based search in document retrieval and the acknowledged usability of this paradigm are convincing reasons for employing the keyword search paradigm on the Semantic Web. Meanwhile there exist many approaches such as [54–57] for the relational domain and [58–60] for the XML domain. Especially the relational domain is relevant to our work due to the similarities to the RDF data model. All these approaches are based on *schema graphs* (i.e. a graph where tables and their primary-foreign key relations are represented as nodes and edges, respectively).

Our work lies in an entity-centric model which not only entities but also characteristics and association of entities simply can be retrieved. Furthermore, it is independent of any linguistic analysis. The main advantage is that we do not rely on an explicitly given schema, which is often missing for datasets on the Web of Data. However, achieving sufficient performance for instant query answering is more an issue in the RDF case, which is why our approach is currently limited to two keywords.

2.7 Conclusion and Future Work

We regard this work as a first step towards the user-friendly querying of the Data Web using rich semantic structures. By tightly intertwining the keyword interpretation and query generation with the available background knowledge we are able to obtain results of relatively good quality. We applied a number of techniques such as a thorough analysis of potential graph patterns so as to limit the search space and enable instant question answering. A problem, however, beyond our control is the data quality and coverage. Currently our evaluation is still limited to 150M facts comprised in DBpedia, yet due to the generic nature and efficiency of the approach we will be able extend it quickly to the whole Data Web. For doing so, we aim to apply some optimizations original to our approach, since we currently use just a plain SPARQL interface. These optimizations will, for example, comprise the pre-computation of views and statistical summaries for each of our graph pattern templates. A current limitation is the restriction to two keywords. The rational behind this was to restrict the search space of possible query interpretations, in order to return the most meaningful results to the user quickly in a compact form. There are a number of possible advancements: (a) to allow a larger number of keywords or (b) to enable users to refine obtained queries and to add additional constraints and (c) to make more extensive use of linguistic features and techniques.

Acknowledgments

We would like to thank our colleagues from AKSW research group for their helpful comments and inspiring discussions during the development of this approach. This work was supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943) and by the Eurostars Project SCMS (E!4604).

CHAPTER 3

SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data[1]

Abstract: The architectural choices underlying Linked Data have led to a compendium of data sources which contain both duplicated and fragmented information on a large number of domains. One way to enable non-experts users to access this data compendium is to provide keyword search frameworks that can capitalize on the inherent characteristics of Linked Data. Developing such systems is challenging for three main reasons. First, resources across different datasets or even within the same dataset can be homonyms. Second, different datasets employ heterogeneous schemas and each one may only contain a part of the answer for a certain user query. Finally, constructing a federated formal query from keywords across different datasets requires exploiting links between the different datasets on both the schema and instance levels. We present Sina, a scalable keyword search system that can answer user queries by transforming user-supplied keywords or natural-languages queries into conjunctive SPARQL queries over a set of interlinked data sources. Sina uses a hidden Markov model to determine the most suitable resources for a user-supplied query from different datasets. Moreover, our framework is able to construct federated queries by using the disambiguated resources and leveraging the link structure underlying the datasets to query. We evaluate Sina over three different datasets. We can answer 25 queries from the QALD-1 correctly. Moreover, we perform as well as the best question answering system from the QALD-3 competition by answering 32 questions correctly while also being able to answer queries on distributed sources. We study the runtime of SINA in its mono-core and parallel implementations and draw preliminary conclusions on the scalability of keyword search on Linked Data.

3.1 Introduction

The principles underlying Linked Data have been applied worldwide to engender the Linked Open Data Cloud, a compendium of more than 300 datasets and more than 31 billions triples.[2] Within this compendium, millions of resources are described, partly over several datasets [61]. The current standard for accessing this wealth of data is the SPARQL query language. Yet, SPARQL is too complex to be used by non-expert users. Consequently, several search approaches have been developed over the last years

[1] Corresponding publication is: Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer, “SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data”, *Journal of Web Semantics Science, Services and Agents on the World Wide Web*, 2014

[2] See `http://lod-cloud.net/state/` for more details.

to enable non-experts to access these datasets (e.g., [26, 28, 29, 31, 41, 62]). While these approaches differ in their details (see section 5.5), they can all be positioned on the following spectrum: On one end of the spectrum are simple keyword search systems that rely on traditional information retrieval approaches to retrieve resources that bear a label similar to the input of the user. We dub such approaches *data-semantics-unaware keyword search* as they do not take the semantics explicated by the data into consideration. The main advantage of such approaches is that they scale well as they can make use of the results of decades of research carried out in the field of information retrieval. On the other end of the spectrum, we find *question answering systems*, which assume a natural-language query as input and convert this query into a full-fledged SPARQL query. These systems rely on natural-language processing tools such as POS tagging and dependency parsers to detect the relations between the elements of the query. The detected relations are then mapped to SPARQL constructs.

The basic idea behind this work is to devise a *data-semantics-aware keyword search* approach, which stands in the middle of the spectrum. Our approach aims to achieve maximal flexibility by being able to generate SPARQL queries from both natural-language queries and keyword queries. This goal is achieved by limiting the type of formal queries (i.e. SPARQL queries) that our approach can generate to conjunctive SPARQL queries. Several challenges need to be addressed to devise such an approach: First, a query segmentation and disambiguation approach for mapping input query to resources has to be devised. To do so, statistical information of the retrieved resources has to be retrieved. Then, a method for generating conjunctive federated SPARQL queries, which can be sent to SPARQL endpoint to retrieve relevant data must be developed.

In this paper, we show how our framework, Sina, implements these different steps. In contrast to previous approaches, Sina can make use of the topology of Linked Data by exploiting links between resources to devise federated SPARQL queries. Thus, it can deal with both retrieving data from either a single dataset or several interlinked datasets. Consequently, it can be used over the whole of the Linked Open Data Cloud. We present a thorough evaluation of our approach on three different datasets: We use the QALD-1 to detect optimal parameters for Sina and present a first evaluation of the approach on this benchmark dataset. We then run Sina on the QALD-3 benchmark and show that we can generate the correct SPARQL queries for 32 of these queries, thus achieving the same results as the best system tested in the benchmark. Finally, we evaluate Sina in a federated scenario against the queries from the life science domain used in [61]. In an effort to make Sina easily portable, we refrained from using dataset-specific indexes and rely fully on the SPARQL endpoint when constructing SPARQL queries. To ensure that our approach still achieves acceptable runtimes, we implemented both a parallel and a sequential version of Sina. In the second part of the evaluation, we thus present a study of Sina's runtime on the QALD-3 benchmark.

This paper is organized as follows: In the subsequent section, we introduce the architecture of the proposed search engine. In Section 3.3, we present the problem at hand in more detail and some of the notations and concepts used in this work. Section 5.3 presents the proposed disambiguation method in detail along with the evaluation of the bootstrapping. In Section 3.5, we then present the key steps of our algorithm for constructing a conjunctive query. Our evaluation results are presented in Section 3.6 while related work is reviewed in Section 5.5. We close with a discussion and future work.

3.2 Overview

In this section, we describe the high-level architecture and implementation of the Sina search engine. Figure 3.1 illustrates the architecture of Sina, which comprises six main components. Each component consumes the output of the previous component:

1. The **query preprocessing** component receives the textual input query and applies three functions: 1) Tokenization: extraction of individual keywords, removing punctuation and capitalization. 2) Stop word removal: removal of common words such as articles and prepositions. Since in this version, we do not recognize type of answers, thus wh-questions are removed as stop words. 3) Word lemmatization: determining the lemma of the remaining keywords.

2. The **segment validation** component groups keywords to form segments. This component validates the grouped segments with respect to the available resources in the underlying knowledge base(s).

3. The **resource retrieval** component obtains relevant resources from the underlying knowledge bases. The retrieval is based on the string matching between valid segments and the `rdfs:label` of resources. Furthermore, more resources are inferred from lightweight `owl:sameAs` reasoning.

4. The **disambiguation** component determines the best subset of resources for the given input query.

5. The **query construction** component results in formal queries (i.e. SPARQL query) using the graph-structure of data.

6. The **representation** component shows the retrieved results after evaluating the generated SPARQL queries.

We implemented SINA as a Java web application which is available online. We deployed two demo instances, one employing DBpedia as background knowledge[3] and a second one operating on several interlinked life-science datasets[4]. SINA has a simple interface similar to common search engines. All steps of SINA are carried out automatically and the user does not have to interact with the system during the search process. Moreover, SINA is accessible via api.

3.3 Problem and Preliminaries

In this section, we introduce some crucial notions employed throughout the paper and describe the main challenges that arise when transforming user queries to formal, conjunctive queries on linked data.

An RDF knowledge base can be viewed as a directed, labeled graph $G_i = (V_i, E_i)$ where V_i is a set of nodes comprising all entities and literal property values, and E_i is a set of directed edges, i.e. the set of all properties. We define linked data in the context of this paper as a graph $G = (V = \bigcup V_i, E = \bigcup E_i)$ containing a set of RDF knowledge bases, which are linked to each other in the sense, that their sets of nodes overlap, i.e. that $V_i \cap V_j \neq \emptyset$.

In this work we focus on user-supplied queries in natural language, which we transform into an ordered set of keywords by tokenizing, stop-word removal and lemmatization. Our input query thus is an n-tuple of keywords, i.e. $Q = (k_1, k_2, ..., k_n)$.

Challenge 1: Resource Disambiguation. In the first step, we aim to map the input keywords to a suitable set of entity identifiers, i.e. resources $R = \{r_1, r_2...r_m\}$. Note that several adjacent keywords can be mapped to a single resource, i.e. $m \leq n$. In order to accomplish this task, the input keywords have to be grouped together into segments. For each segment, a suitable resource is then to be determined. The challenge here is to determine the right segment granularity, so that the most suitable mapping to identifiers in the underlying knowledge base can be retrieved for constructing a conjunctive query answering the input query.

[3] `http://sina.aksw.org/`
[4] `http://sina-linkeddata.aksw.org/`

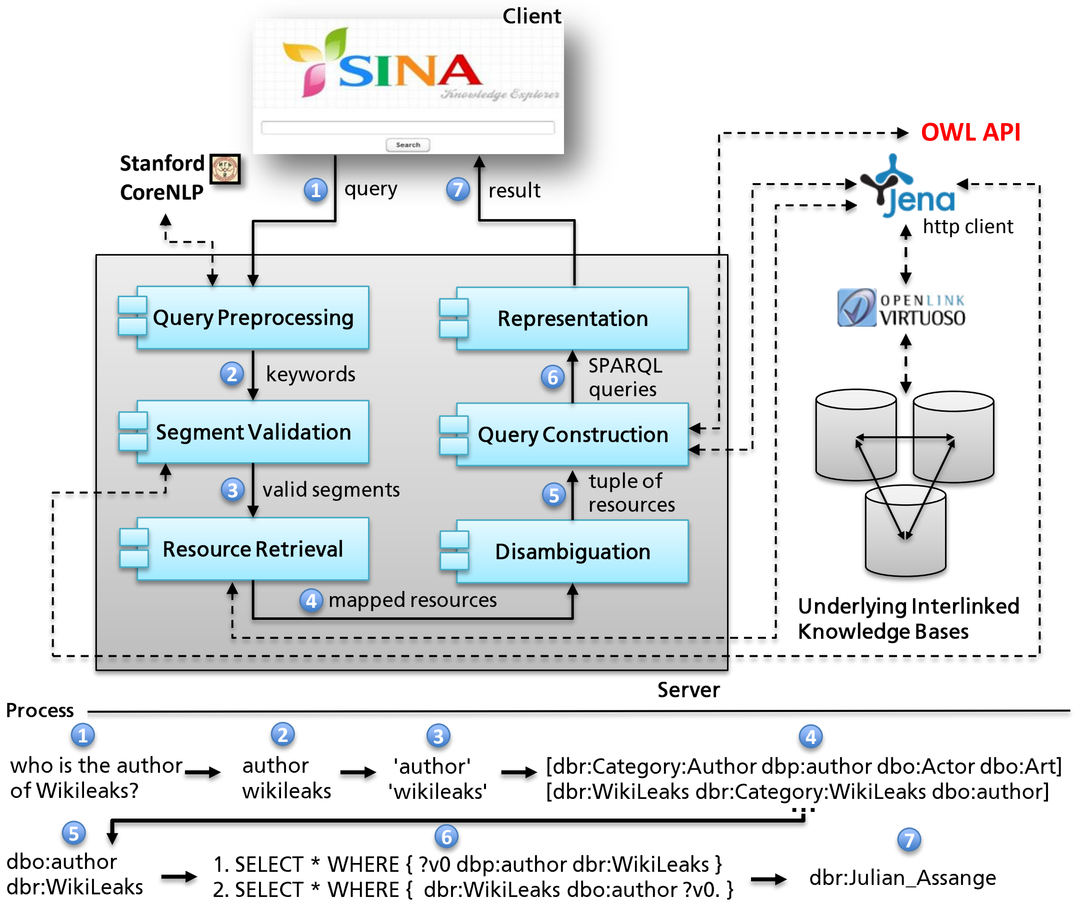


Figure 3.1: Architecture of SINA search engine.

For example, the question *'What are the side effects of drugs used for Tuberculosis?'* is transformed into the 4-keyword tuple *(side, effect, drug, Tuberculosis)*. This tuple can be segmented into (*'side effect drug'*, *'Tuberculosis'*) or (*'side effect'*, *'drug'*, *'Tuberculosis'*). Note that the second segmentation is more likely to lead to a query that contains the results intended by the user. In addition to detecting the right segments for a given input query, we also have to map each of these segments to a suitable resource in the underlying knowledge base. This step is dubbed *entity disambiguation* and is of increasing importance since the size of knowledge bases and schemes heterogeneity on the Linked Data Web grows steadily. In this example, the segment *'Tuberculosis'* is ambiguous when querying both Sider and Diseasome because it may refer to the resource `diseasome:Tuberculosis` describing the disease Tuberculosis or to the resource `sider:Tuberculosis` being the side effect caused by some drugs.

Challenge 2: Query Construction. Once the segmentation and disambiguation have been completed, adequate SPARQL queries have to be generated based on the detected resources. In order to generate a conjunctive query, a connected subgraph $G' = (V', E')$ of G called the **query graph** has to be determined. The intuition behind constructing such a query graph is that it has to fully cover the set of mapped resources $R = \{r_1, ..., r_m\}$ while comprising a minimal number of vertices and edges $(|V'| + |E'|)$. In linked data, mapped resources r_i may belong to different graphs G_i. Thus, the query construction algorithm must be able to traverse the links between datasets at both schema and instance levels. With respect to the previous example, after applying disambiguation on the identified resources, we would obtain the following resources from different datasets: `sider:sideEffect`, `diseasome:possibleDrug`, `diseasome:1154`. The appropriate conjunctive query contains the following triple patterns:

```
1. diseasome:1154  diseasome:possibleDrug     ?v1 .
2. ?v1             owl:sameAs                 ?v2 .
3. ?v2             sider:sideEffect           ?v3 .
```

The second triple pattern bridges between the datasets Drugbank and Sider.

3.3.1 Resource Disambiguation

In this section, we present the formal notations for addressing the resource disambiguation challenge, aiming at mapping the n-tuple of keywords $Q = (k_1, k_2, ..., k_n)$ to the m-tuple of resources $R = (r_1, ..., r_m)$.

Definition 9 (Segment and Segmentation)**.** *For a given query $Q = (k_1, k_2, ..., k_n)$, the segment $S_{(i,j)}$ is the sequence of keywords from start position i to end position j, i.e., $S_{(i,j)} = (k_i, k_{i+1}, ..., k_j)$. A query segmentation is an m-tuple of segments $SG(Q) = (S_{(0,i)}, S_{(i+1,j)}, ..., S_{(l,n)})$ with non-overlapping segments arranged in a continuous order, i.e. for two continuous segments $S_x, S_{x+1} : Start(S_{x+1}) = End(S_x) + 1$. The concatenation of segments belonging to a segmentation forms the corresponding input query Q.*

Definition 10 (Resource Disambiguation)**.** *Let the segmentation $SG' = (S^1_{(0,i)}, S^2_{(i+1,j)}, ..., S^m_{(l,n)})$ be the suitable segmentation for the given query Q. Each segment S^i of SG' is first mapped to a set of candidate resources $R_i = \{r_1, r_2...r_h\}$ from the underlying knowledge base. The aim of the disambiguation step is to detect an m-tuple of resources $(r_1, r_2, ..., r_m) \in R_1 \times R_2 \times \ldots \times R_m$ from the Cartesian product of the sets of candidate resources for which each r_i has two important properties: First, it is among the highest ranked candidates for the corresponding segment with respect to the similarity as well as popularity and second it shares a semantic relationship with other resources in the m-tuple. Semantic relationship refers to the existence of a path between resources.*

The disambiguated m-tuple is appropriate if a query graph [capable of answering the input query] can be constructed using all resources contained in that m-tuple. The order in which keywords appear in the original query is partially significant for mapping. However, once a mapping from keywords to resources is established the order of the resources does not affect the SPARQL query construction anymore. This is a fact that users will write strongly related keywords together, while the order of only loosely related keywords or keyword segments may vary. When considering the order of keywords, the number of segmentations for a query Q consisting of n keywords is $2^{(n-1)}$. However, not all these segmentations contain valid segments. A *valid segment* is a segment for which at least one matching resource can be found in the underlying knowledge base. Thus, the number of segmentations is reduced by excluding those containing invalid segments.

Algorithm 2 shows a naive approach for finding all valid segments when considering the order of keywords. It starts with the first keyword in the given query as first segment, then adds the next keyword to the current segment and checks whether this addition would render the new segment invalid. This process is repeated until we reach the end of the query. The input query is usually short. The number of keywords is mainly less than 6[5]; therefore, this algorithm is not expensive. Table 3.1 shows the set of valid segments along with some samples of the candidate resources computed for the previous example using the naive algorithm. Note that 'side effect drug', 'side', 'effect' are not valid segments.

[5] http://www.keyworddiscovery.com/keyword-stats.html?date=2012-08-01

Valid Segments	Samples of Candidate Resources
side effect	1. sider:sideEffect 2. sider:side_effects
drug	1. drugbank:drugs 2. class:Offer 3. sider:drugs 4. diseasome:possibledrug
tuberculosis	1. diseases:1154 2. side_effects:C0041296

Table 3.1: Generated segments and samples of candidate resources for a given query.

```
Data : q: n-tuple of keywords, knowledge base
Result : SegmentSet: Set of segments
SegmentSet=new list of segments;
start=1;
while start <= n do
    i = start;
    while $S_{(start,i)}$ is valid do
        $SegmentSet.add(S_{(start,i)})$;
        i++;
    end
    start++;
end
```
Algorithmus 2 : Naive algorithm for determining all valid segments taking the order of keywords into account.

3.3.2 Construction of Conjunctive Queries

The second challenge addressed by this paper tackles the problem of generating a federated conjunctive query leveraging the disambiguated resources i.e. $R = (r_1, ..., r_m)$. Herein, we consider conjunctive queries being conjunctions of SPARQL algebra triple patterns[6]. We leverage the disambiguated resources and implicit knowledge about them (i.e. types of resources, interlinked instances and schema as well as domain and range of resources with the type property) to form the triple patterns.

For instance, for the running query which asks for a list of resources (i.e. side effects) which have a specific characteristic in common (i.e. 'caused by drugs used for Tuberculosis'). Suppose the resources identified during the disambiguation process are: `sider:sideEffect`, `Diseasome:possibleDrug` as well as `Diseasome:1154`. Suitable triple patterns which are formed using the implicit knowledge are:

```
1. Diseasome:1154  Diseasome:possibleDrug      ?v1 .
2. ?v1             owl:sameAs                  ?v2 .
3. ?v2             sider:sideEffect            ?v3 .
```

The second triple pattern is formed based on interlinked data information. This triple connects the resources with the type `drug` in the dataset Drugbank to their equivalent resources with the type `drug` in the Sider dataset using `owl:sameAs` link. These triple patterns satisfy the information need expressed in the input query. Since most of common queries commonly lack of a quantifier, thus conjunctive queries to a large extend capture the user information need. A conjunctive query is called query graph and formally defined as follows.

Definition 11 (Query Graph)**.** *Let a set $R = \{r_1, ..., r_n\}$ of resources (from potentially different knowledge bases) be given. A query graph $QG_R = (V', E')$ is a directed, connected multi-graph such that $R \subseteq E' \cup V'$. Each edge $e \in E'$ is a resource that represents a property from the underlying knowledge bases. Two nodes n and $n' \in V'$ can be connected by e if n (resp. n') satisfies the domain (resp. range) restrictions of e. Each query graph built by these means corresponds to a set of triple patterns, i.e. $QG \equiv \{(n, e, n') | (n, n') \in V^2 \wedge e \in E\}$.*

[6] Throughout the paper, we use the standard notions of the RDF and SPARQL specifications, such as graph pattern, triple pattern and RDF graph.

3.4 Resource Disambiguation using Hidden Markov Models

In this section, we describe how we use a HMM for the concurrent segmentation of queries and disambiguation of resources. First, we introduce the notation of HMM parameters and then we detail how we bootstrap the parameters of our HMM for solving the query segmentation and entity disambiguation problems.

Hidden Markov Models: Formally, a hidden Markov model (HMM) is a quintuple $\lambda = (X, Y, A, B, \pi)$ where:

- X is a finite set of states. In our case, X is a subset of the resources contained in the underlying graphs.
- Y denotes the set of observations. Herein, Y equals to the valid segments derived from the input n-tuple of keywords.
- $A : X \times X \rightarrow [0, 1]$ is the transition matrix of which each entry a_{ij} is the transition probability $Pr(S_j|S_i)$ from state i to state j;
- $B : X \times Y \rightarrow [0, 1]$ represents the emission matrix. Each entry $b_{ih} = Pr(h|S_i)$ is the probability of emitting the symbol h from state i;
- $\pi : X \rightarrow [0, 1]$ denotes the initial probability of states.

Commonly, estimating the hidden Markov model parameters is carried out by employing supervised learning. We rely on *bootstrapping*, a technique used to estimate an unknown probability distribution function. Specifically, we bootstrap[7] the parameters of our HMM by using string similarity metrics (i.e., *Levenshtein* and *Jaccard*) for the emission probability distribution and more importantly the topology of the graph for the transition probability. The results of the evaluation show that by using these bootstrapped parameters, we achieve a high mean reciprocal rank (MRR) above 84% (discussed in Section 5.3.1).

Constructing the State Space: A-priori, the state space should be populated with as many states as the total number of entities in the knowledge base. The number of states in X is thus potentially large given that X will contain all RDF resources contained in the graph G on which the search is to be carried out, i.e. $X = V \cup E$. For DBpedia, for example, X would contain more than 3 million states. To reduce the number of states, we exclude irrelevant states based on the following observations: (1) A relevant state is a state for which a valid segment can be observed (we described the recognition of valid segments in subsection 3.3.1). (2) A valid segment is observed in a state if the probability of emitting that segment is higher than a given threshold θ. The probability of emitting a segment from a state is computed based on the similarity score which we describe in subsection 5.3.1. Thus, we can prune the state space such that it contains solely the subset of the resources from the knowledge bases for which the emission probability is higher than θ. In addition to these states, we add an **unknown entity state** (UE) which represents all entities that were pruned. Based on this construction of state space, we are now able to detect likely segmentations and disambiguation of resources, the segmentation being the labels emitted by the elements of the most likely sequence of states. The disambiguated resources are the states determined as the most likely sequence of states.

Extension of State Space with reasoning: A further extension of the state space can be carried out by including resources inferred from lightweight `owl:sameAs` reasoning. We precomputed and added the triples inferred from the symmetry and transitivity property of the `owl:sameAs` relation. Consequently, for extending the state space, for each state representing a resource `x` we just include states for all resources `y`, which are in an `owl:sameAs` relation with `x`.

[7] For the bootstrapping test, we used 11 sample queries from the QALD benchmark 2012 training dataset.

3.4.1 Bootstrapping the Model Parameters

Our bootstrapping approach for the model parameters A and π is based on the HITS algorithm and semantic relations between resources in the knowledge base. The rationale is that the semantic relatedness of two resources can be defined in terms of two parameters: the distance between the two resources and the popularity of each of the resources. The distance between two resources is the path length between those resources. The popularity of a resource is simply the connectivity degree of the resource with other resources available in the state space. We use the HITS algorithm for transforming these two values to hub and authority values (as detailed below). An analysis of the bootstrapping shows significant improvement of accuracy due to this transformation. In the following, we first introduce the *HITS* algorithm, since it is employed within the functions for computing the two HMM parameters A and π. Then, we discuss the distribution functions proposed for each parameter. Finally, we compare our bootstrapping method with other well-known distribution functions.

Hub and Authority of States. *Hyperlink-Induced Topic Search* (HITS) is a link analysis algorithm that was developed originally for ranking Web pages [63]. It assigns a hub and an authority value to each Web page. The *hub value* estimates the value of links to other pages and the *authority value* estimates the value of the content on a page. Hub and authority values are mutually interdependent and computed in a series of iterations. In each iteration the authority value is updated to the sum of the hub scores of each referring page; and the hub value is updated to the sum of the authority scores of each referring page. After each iteration, hub and authority values are normalized. This normalization process causes these values to converge eventually.

Since RDF data forms a graph of linked entities, we employ a weighted version of the HITS algorithm in order to assign different popularity values to the states based on the distance between states. We compute the distance between states employing weighted edges. For each two states S_i and S_j in the state space, we add an edge if there is a path of maximum length k between the two corresponding resources. Note that we also take `property` resources into account when computing the path length.The weight of the edge between the states S_i and S_j is set to $w_{i,j} = k - pathLength(i, j)$, where $pathLength(i, j)$ is the length of the path between the corresponding resources. The authority of a state can now be computed by: $auth(S_j) = \sum_{S_i} w_{i,j} \times hub(S_i)$. The hub value of a state is given by $hub(S_j) = \sum_{S_i} w_{i,j} \times auth(S_i)$. These definitions of hub and authority for states are the foundation for computing the transition and initial probabilities in the HMM.

Transition Probability. To compute the transition probability between two states, we take both, the connectivity of the whole space state as well as the weight of the edge between the two states, into account. The transition probability value decreases while increasing distance between states. For example, transitions between entities in the same triple have a higher probability than transitions between entities in triples connected through auxiliary intermediate entities. In addition to edges representing the shortest path between entities, there is an edge between each state and the *unknown entity (UE)* state. The transition probability of state S_j following state S_i is denoted as $a_{ij} = Pr(S_j|S_i)$. Note that the condition $\sum_{S_i} Pr(S_j|S_i) = 1$ holds.The transition probability from the state S_i to UE is defined as:

$$a_{iUE} = Pr(UE|S_i) = 1 - hub(S_i)$$

Consequently, a good hub has a smaller probability of transition to *UE*. The transition probability from the state S_i to the state S_j is computed by:

$$a_{ij} = Pr(S_j|S_i) = \frac{auth(S_j)}{\sum\limits_{\forall a_{ik}>0} auth(S_k)} \times hub(S_i)$$

Here, the probabilities from state S_i to the neighbouring states are uniformly distributed based on the authority values. Consequently, states with higher authority values are more probable to be met.

Initial Probability. The initial probability $\pi(S_i)$ is the probability that the model assigns to the initial state S_i at the beginning. The initial probabilities fulfill the condition $\sum\limits_{\forall S_i} \pi(S_i) = 1$. We denote states for which the first keyword is observable by $InitialStates$. The initial states are defined as follows:

$$\pi(S_i) = \frac{auth(S_i) + hub(S_i)}{\sum\limits_{\forall S_j \in InitialStates} (auth(S_j) + hub(S_j))}$$

In fact, $\pi(S_i)$ of an initial state is uniformly distributed on both hub and authority values.

Emission Probability. Both the labels of states and the segments contain sets of words. For computing the emission probability of the state S_i and the emitted segment h, we compare the similarity of the label of state S_i with the segment h in two levels, namely string-similarity and set-similarity level:

- The *string-similarity level* measures the string similarity of each word in the segment with the most similar word in the label using the *Levenshtein distance*.
- The *set-similarity level* measures the difference between the label and the segment in terms of the number of words using the *Jaccard similarity*.

Our similarity score is a combination of these two metrics. Consider the segment $h = (k_i, k_{i+1}, ..., k_j)$ and the words from the label l divided into a set of keywords M and stopwords N, i.e. $l = M \cup N$. The total similarity score between keywords of a segment and a label is then computed as follows:

$$b_{ih} = Pr(h|S_i) = \frac{\sum\limits_{t=i}^{j} \text{argmax}_{m_i \in M}(\sigma(m_i, k_t))}{|M \cup h| + 0.1 \times |N|}$$

This formula is essentially an extension of the *Jaccard similarity coefficient*. The difference is that we use the sum of the string-similarity score of the intersections in the numerator instead of the cardinality of intersections. As in the Jaccard similarity, the denominator comprises the cardinality of the union of two sets (keywords and stopwords). The difference is that the number of stopwords is down-weighted by the factor 0.1 to reduce their influence since they do not convey much supplementary semantics.

Viterbi Algorithm for the K-best Set of Hidden States. The optimal path through the HMM for a given sequence (i.e. input query keywords) generates disambiguated resources which form a correct segmentation. The *Viterbi algorithm* or *Viterbi path* [64] is a dynamic programming approach for finding the optimal path through a HMM for a given input sequence. It discovers the most likely sequence of underlying hidden states that might have generated a given sequence of observations. This discovered path has the maximum joint emission and transition probability of the involved states. The sub-paths of this most likely path also have the maximum probability for the respective sub sequence of observations. The naive version of this algorithm just keeps track of the most likely path. We extended this algorithm using a tree data structure to store all possible paths generating the observed query keywords. Thus, our implementation can provide a ranked list of all paths generating the observation sequence with the corresponding probability.

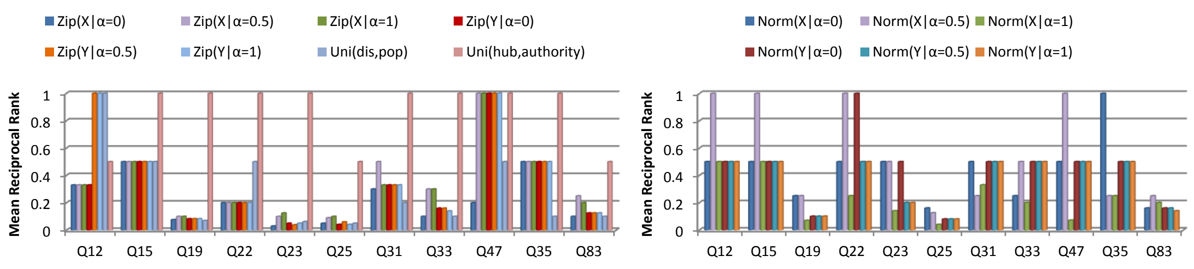


Figure 3.2: MRR of different distributions per query for bootstrapping the transition probability.

Example 3.13 Let us consider the query: `What are the side effects of drugs used for Tuberculosis?`. The validated segments are: `‘side effect’, ‘drug’ and ‘Tuberculosis’`. After the retrieval and pruning process, the state space contains the resources listed in Table 5.1:
By running the *Viterbi algorithm* with the associated probabilities, we have a ranked list of the chosen resources that the sequence *{side effect, drug, Tuberculosis}* is observable through them. In the following, we show the top-4 most likely paths along with their associated probability.

1. `0.0033:  Sider:sideEffect, Diseasome:possibleDrug, Diseasome:1154.`
2. `0.0017:  Sider:sideEffect, Diseasome:possibleDrug, Sider:C0041296.`
3. `6.0257E-4:  Sider:sideEffect, Sider:drugs,    Diseasome 1154.`
4. `4.0805E-4:  Sider:sideEffect, Drugbank:Offer,    Diseasome:1154.`

Segment	Resources	Label	Type
side effect	1. `Sider:sideEffect`	side effect	property
	2. `Sider:side_effects`	side effect	class
drug	1. `Drugbank:drugs`	drug	class
	2. `Drug bank:offer`	drug	class
	3. `Sider:drugs`	drug	class
	4. `Diseasome:possibleDrug`	possible drug	property
Tuberculosis	1. `Diseasome:1154`	tuberculosis	instance
	2. `Sider:C0041296`	tuberculosis	instance

Table 3.2: The resources contained in the state space for a given query.

3.4.2 Evaluation of Bootstrapping

We evaluated the accuracy of our approximation of the transition probability A (which is basically a kind of uniform distribution) in comparison with two other distribution functions, i.e., *Normal* and *Zipfian* distributions. Moreover, to measure the effectiveness of the *hub* and *authority* values, we ran the distribution functions with two different inputs, i.e. *distance* and *connectivity degree* values as well as *hub* and *authority* values. Note that for a given edge the source state is the one from which the edge originates and the sink state is the one where the edge ends. We ran the distribution functions separately with X being defined as the weighted sum of the normalized distance between two states and normalized connectivity degree of the sink state:

$$X_{ij} = \alpha \times distance_{(S_i - S_j)} + (1 - \alpha) \times (1 - connectivityDegree_{S_j})$$

Similarly, Y was defined as the weighted sum of the hub of the source state and the authority of the sink state: $Y = \alpha \times hub(S_i) + (1 - \alpha) \times (1 - authorithy_{s_j})$. In addition to measuring the effectiveness of *hub* and *authority*, we also measured a similar uniform function with the input parameters *distance* and *connectivity degree* defined as:

$$a_{ij} = \frac{distance(S_i - S_j)}{\sum\limits_{\forall S_k > 0} distance(S_i - S_k)} \times connectivitydegree(S_i)$$

Given that the model at hand generates and scores a ranked list of possible tuples of resources, we compared the results obtained with the different distributions by looking at the *mean reciprocal rank* (MRR) [65] they achieve. For each query $q_i \in Q$ in the benchmark, we compare the rank r_i assigned by different algorithms with the correct tuple of resources and set $MRR(\mathcal{A}) = \frac{1}{|Q|} \sum\limits_{q_i} \frac{1}{r_i}$. Note that if the correct tuple of resources was not found, the reciprocal rank was assigned the value 0. We used 11 queries from QALD2-Benchmark 2012 training dataset for bootstrapping[8]. The criterion of choosing the bootstrapped queries was the number of the keywords as well as the associated resources. Figure 3.2 shows the *MRR* achieved by bootstrapping the transition probability of this model with 3 different distribution functions per query in 14 different settings. Figure 3.3 compares the average *MRR* for different functions employed for bootstrapping the transition probability per setting. Our results show clearly that the proposed function is superior to all other settings and achieves an MRR of approximately 81%. A comparison of the MRR achieved when using *hub* and *authority* with that obtained when using *distance* and *connectivity degree* reveals that using *hub* and *authority* leads to an 8% improvement on average. This difference is trivial in Zipfian and Normal settings, but very significant in the case of a uniform distribution. Essentially, *HITS* fairly assigns qualification values for the states based on the topology of the graph.

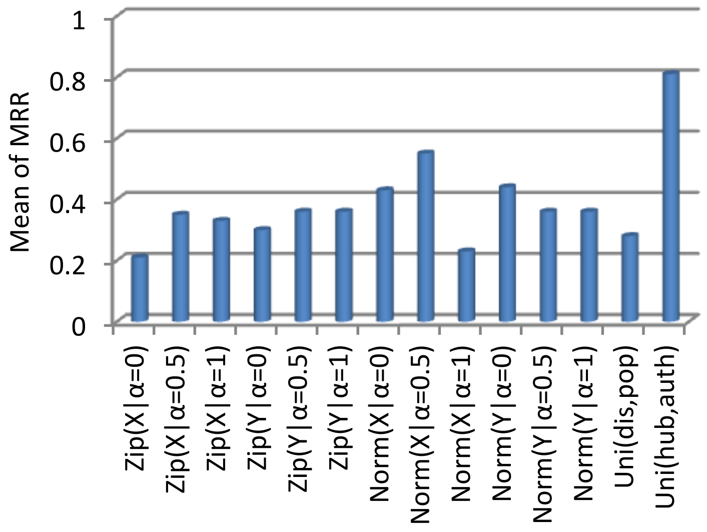


Figure 3.3: Comparison of different functions and settings for bootstrapping the transition probability. Uni stands for the uniform distribution, while Zip stands for the Zipfian and Norm for the normal distribution.

We bootstrapped the emission probability B with two distribution functions based on (1) Levenshtein similarity metric, (2) the proposed similarity metric as a combination of the Jaccard and Levenshtein

[8] http://www.sc.cit-ec.uni-bielefeld.de/qald-2

measures. We observed the *MRR* achieved by bootstrapping the emission probability of this model employing those two similarity metrics per query in two settings (i.e. natural and reverse order of query keywords). The results show no difference in *MRR* between these two metrics in the natural order. However, in the reverse order the Levenshtein metric failed for 81% of the queries, while no failure was observed with the combination of Jaccard and Levenshtein. Hence, our combination is robust with regard to change of input keyword order. For bootstrapping the initial probability π, we compared the uniform distribution on both – hub and authority – values with a uniform distribution on the number of states for which the first keyword is observable. The result of this comparison shows a 5% improvement for the proposed function. Figure 3.4 shows the mean of *MRR* for different values of the threshold θ employed for pruning the state space. A high value of θ prevents inclusion of some relevant resources and a low value adds irrelevant resources. It can be observed that the optimal value of θ is in the range $[0.6, 0.7]$. Thus, we set θ to 0.6 in the rest of our experiments.

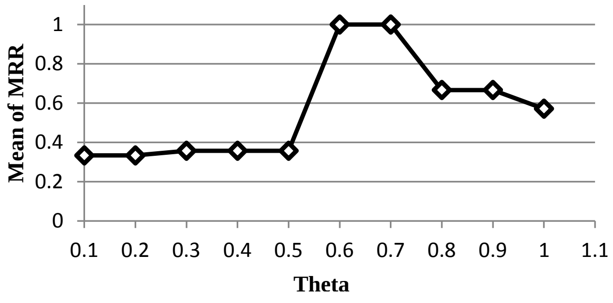


Figure 3.4: Mean MRR for different values of θ.

3.5 Query Graph Construction

The goal of query graph construction is generating a conjunctive query (i.e. SPARQL query) from a given set of resource identifiers i.e., $R = \{r_1, r_2, ...r_m\}$. The core of SPARQL queries are *basic graph patterns*, which can be viewed as a query graph *QG*. In this section, we first discuss the formal considerations underlying our query graph generation strategy and then describe our algorithm for generating the query graph. The output of this algorithm is a set of graph templates. Each graph template represents a comprehensive set of query graphs, which are isomorphic regarding edges. A query graph *A* is isomorphic regarding its edges to a query graph *B*, if *A* can be derived from *B* by changing the labels of edges.

3.5.1 Formal Considerations

A query graph *QG* consists of a conjunction of triple patterns denoted by (s_i, p_i, o_i). When the set of resource identifiers *R* is given, we aim to generate a query graph *QG* satisfying the *completeness* restriction, i.e., each r_i in *R* maps to at least one resource in a triple pattern contained in *QG*. For a given set of resources *R*, the probability of a generated query graph $\Pr(QG|R)$ being relevant for answering the information need depends on the probability of all corresponding triple patterns to be relevant. We assume that triple patterns are independent with regard to the relevance probability. Thus, we define the relevance probability for a *QG* as the product of the relevance probabilities of the *n* containing triple patterns. We denote the triple patterns with $(s_i, p_i, o_i)_{i=1...n}$ and their relevance probability with $\Pr(s_i, p_i, o_i)$, thus rendering $\Pr(QG|R) = \prod_{i=1}^{n} \Pr(s_i, p_i, o_i)$. We aim at constructing *QG* with the highest relevance probability, i.e.

$\arg\max \Pr(QG|R)$. There are two parameters that influence $\Pr(QG|R)$: (1) the number of triple patterns and (2) the number of free variables, i.e. variables in a triple pattern that are not bound to any input resource. Given that $\forall(s_i, p_i, o_i) : \Pr(s_i, p_i, o_i) \leq 1$, a low number of triple patterns increases the relevance probability of QG. Thus, our approach aims at generating small query graphs to maximize the relevance probability. Regarding the second parameter, more free variables increase the uncertainty and consequently cause a decrease in $\Pr(QG|R)$. As a result of these considerations, we devise an algorithm that minimizes the number of both the number of free variables and the number of triple patterns in QG. Note that each triple pattern, the subject s_i (resp. object o_i) should be included in the domain (resp. range) of the predicate p_i or be a variable. Otherwise, we assume the relevance probability of the given triple pattern to be zero:

$$(s_i \notin domain(p_i)) \vee (o_i \notin range(p_i)) \Rightarrow \Pr(s_i, p_i, o_i) = 0.$$

Forward Chaining. One of the prerequisites of our approach is the inference of implicit knowledge on the types of resources as well as domain and range information of the properties. We define the *comprehensive type* (CT) of a resource r as the set of all super-classes of explicitly stated classes of r (i.e., those classes associated with r via the `rdf:type` property in the knowledge base). The comprehensive type of a resource can be easily computed using forward chaining on the `rdf:type` and `rdfs:subClassOf` statements in the knowledge base. We can apply the same approach to properties to obtain maximal knowledge on their domain and range. We call the extended domain and range of a property *p comprehensive domain* (CD_p) and *comprehensive range* (CR_p). We reduce the task of finding the *comprehensive properties* ($CP_{r-r'}$) which link two resources r and r' to find properties p such that the comprehensive domain (resp. comprehensive range) of p intersects with the comprehensive type of r resp r' or vice-versa. We call the set OP_r (resp. IP_r) of all properties that can originate from (resp. end with) a resource r the set of outgoing (resp. incoming) properties of r.

3.5.2 Approach

To construct possible query graphs, we generate in a first step an *incomplete query graph* $IQG(R) = (V'', E'')$ such that the vertices V'' (resp. edges E'') are either equal or subset of the vertices (resp. edges) of the final query graph $V'' \subseteq V'$ (resp. $E'' \subseteq E'$). In fact, an incomplete query graph (IQG) contains a set of disjoint sub-graphs, i.e. there is no vertex or edge in common between the sub-graphs: $IQG = \{g_i(v_i, e_i) | \forall g_i \neq g_j : v_i \cap v_j = \emptyset \wedge e_i \cap e_j = \emptyset\}$. An IQG connects a maximal number of the resources detected beforehand in all possible combinations.

The IQG is the input for the second step of our approach, which transforms the possibly incomplete query graphs into a set of final query graphs QG. Note that for the second step, we use an extension of the minimum spanning tree method that takes subgraphs (and not sets of nodes) as input and generates a minimal spanning graph as output. Since in the second step, the minimum spanning tree does not add any extra intermediate node (except nodes connected by `owl:sameAs` links), it eliminates both the need of keeping an index over the neighbourhood of nodes, and of using exploration for finding paths between nodes.

Generation of IQGs After identifying a corresponding set of resources $R = \{r_1, r_2, ...r_m\}$ for the input query, we can construct vertices V' and primary edges of the query graph $E'' \subseteq E'$ in an initial step. Each resource r is processed as follows: (1) If r is an instance, CT of this vertex is equivalent to $CT(r)$ and the label of this vertex is r. (2) If r is a class, CT of this vertex just contains r and the label of this vertex is a new variable.

After the generation of the vertices for all resources that are instances or classes, the remaining resources (i.e., the properties) generate an edge and zero (when connecting existing vertices), one (when connecting an existing with a new vertex) or two vertices. This step uses the sets of incoming and outgoing properties as computed by the forward chaining. For each resource r representing a property we proceed as follows:

- If there is a pair of vertices (v, v') such that r belongs to the intersection of the set of outgoing properties of v and the set of incoming properties of v' (i.e. $r \in OP_v \cap IP_{v'}$), we generate an edge between v and v' and label it with r. Note that in case several pairs (v, v') satisfy this condition, an IQG is generated for each pair.

- Else, if there is a vertex v fulfilling the condition $r \in OP_v$, then we generate a new vertex u with the CT_u being equal to CR_r and an edge labeled with the r between those vertices (v, u). Also, if the condition $r \in IP_v$ for v holds, a new vertex w is generated with CT_w being equal to CD_r as well as an edge between v and w labeled with r.

- If none of the above holds, two vertices are generated, one with CT equal to CD_r and another one with CT equal to CR_r. Also, an edge between these two vertices with label r is created.

This policy for generating vertices keeps the number of free variables at a minimum. Note that whenever a property is connected to a vertex, the associated CT of that vertex is updated to the intersection of the previous CT and CD_p (CR_p respectively) of the property. Also, there may be different options for inserting a property between vertices. In this case, we construct an individual IQG for each possible option. If the output of this step generates an IQG that contains one single graph, we can terminate as there is no need for further edges and nodes.

Example 3.14 We look at the query: `What are the side effects of drugs used for Tuberculosis?`. Assume the resource disambiguation process has identified the following resources:

```
1. diseasome:possibleDrug     (type property)
   CD={diseasome:disease},    CR={drugbank:drugs}
2. diseasome:1154             (type instance)
   CT={diseasome:disease}
3. sider:sideEffect           (type property)
   CD={sider:drug},           CR={sider:sideeffect}
```

After running the IQGs generation, since we have only one resource with the type `class` or `instance`, just one vertex is generated. Thereafter, since only the domain of `possibleDrug` intersects with the CT of the node `1154`, we generate: (1) a new vertex labeled $?v0$ with the CT being equal to
CR =`possibleDrug`, and (2) an edge labeled `possibleDrug` from `1154` to $?v0$. Since, there is no matched node for the property `sideEffect` we generate: (1) a new vertex labeled $?v1$ with the CT being equal to `sider:drug`, (2) a new vertex labeled $?v2$ with the CT being equal to `sider:sideeffect`, (3) an edge labeled `sideEffect` from $?v1$ to $?v2$. Figure 3.5 shows the constructed IQG, which contains two disjoint graphs.

Connecting Sub-graphs of an IQG Since the query graph QG must be a connected graph, we need to connect the disjoint sub-graphs in each of the IQGs. The core idea of our algorithm utilizes the *Minimum Spanning Tree* (MST) approach, which builds a tree over a given graph connecting all the

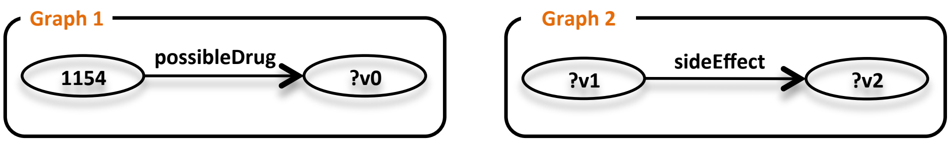


Figure 3.5: *IQG* for the Example 14.

vertices. We use the idea behind Prim's algorithm [66], which starts with all vertices and subsequently incrementally includes edges. However, instead of connecting vertices we connect individual disjoint sub-graphs. Hence, we try to find a minimum set of edges (i.e., properties) to span a set of disjoint graphs so as to obtain a connected graph. Therewith, we can generate a query graph that spans all vertices while keeping the number of vertices and edges at a minimum. Since a single graph may have many different spanning trees, there may be several query graphs that correspond to each *IQG*. We generate all different spanning graphs because each one may represent a specific interpretation of the user query.

To connect two disjoint graphs we need to obtain edges that qualify for connecting a vertex in one graph with a suitable vertex in the other graph. We obtain these properties by computing the set of comprehensive properties *CP* (cf. subsection 3.5.1) for each combination of two vertices from different sub-graphs. Note that if two vertices are from different datasets, we have to traverse `owl:sameAs` links to compute a comprehensive set of properties. This step is crucial for constructing a federated query over interlinked data. In order to do so, we first retrieve the direct properties between two vertices `?v0 ?p ?v1.` In case such properties exist, we add an edge between those two vertices to *IQG*. Then, we retrieve the properties connecting two vertices via an `owl:sameAs` link. To do that, we employ two graph patterns: (1) `?v0 owl:sameAs ?x. ?x ?p ?v1.` (2) `?v0 ?p ?x. ?x owl:sameAs ?v1.` The resulting matches to each of these two patterns are added to the *IQG*. Finally, we obtain properties connecting vertices having `owl:sameAs` links according to the following pattern:
`?v0 owl:sameAs ?x. ?x ?p ?y. ?y owl:sameAs ?v1.` Also, matches for this pattern are added to the *IQG*.

For each connection discovered between a pair of vertices (v, v'), a different *IQG* is constructed by adding the found edge connecting those vertices to the original *IQG*. Note that the *IQG* resulting from this process contains less unconnected graphs than the input *IQG*. The time complexity in the worst case is $O(|v|^2)$ (with $|v|$ being the number of vertices).

Example 3.15 To connect two disjoint graphs i.e. Graph 1 and Graph 2 of the *IQG* shown in Example 14, we need to obtain edges that qualify for connecting either the vertex `1154` or $?v0$ to either vertex $?v1$ or $?v2$ in Graph 2. Forward chaining reveals the existence of two `owl:sameAs` connections between two vertices i.e. (1) `1154` and $?v2$, (2) $?v0$ and $?v1$. Therefore, we can construct the first query graph template by adding an edge between `1154` and $?v2$ and the second query graph template by adding an edge between $?v0$ and $?v1$. The two generated query graph templates are depicted in Figure 3.6.

3.6 Evaluation

Experimental Setup The goal of our evaluation was to determine effectiveness and efficiency of (1) the resource disambiguation and (2) the query construction with respect to accuracy and runtime. We employed four different knowledge bases: *DBpedia* as one individual knowledge base and the three interlinked knowledge bases *Drugbank*, *Sider* and *Diseasome*. We measured the effectiveness of our resource disambiguation approach using the *Mean Reciprocal Rank* (MRR). For each query $q_i \in Q$ in the

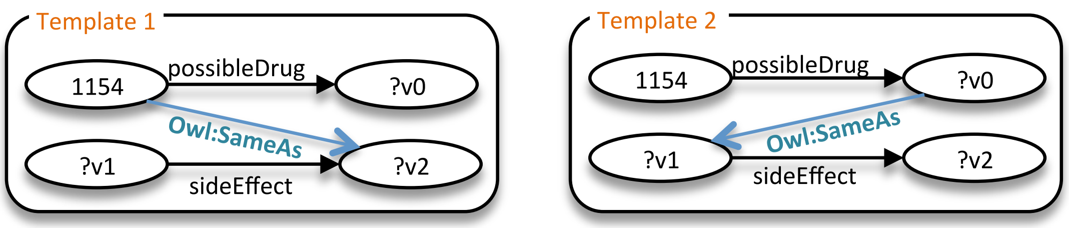


Figure 3.6: Generated query graph templates.

benchmark, we compare the rank r_i assigned by our disambiguation method to the correct m-tuple of resources: $MRR(\mathcal{A}) = \frac{1}{|Q|} \sum\limits_{q_i} \frac{1}{r_i}$. Moreover, we measured the accuracy of the query construction in terms of precision and recall being defined as follows:

$$Recall = \frac{|correct\ resources\ returned\ by\ generated\ query|}{|resources\ in\ gold\ standard\ answer|}$$

$$Precision = \frac{|correct\ resources\ returned\ by\ generated\ query|}{|all\ resources\ returned\ by\ generated\ query|}$$

The query construction is initiated with the top-2 tuples returned by the disambiguation. For testing the statistical significance of our results, we used a Wilcoxon signed ranked test with a significance level of 95%.

3.6.1 Accuracy Benchmark over Interlinked Knowledge Bases

To the best of our knowledge, no benchmark for federated queries over Linked Data has been created so far. Thus, we created a benchmark consisting of 25 queries on the three interlinked datasets Drugbank, Sider and Diseasome for the purposes of our evaluation[9]. The benchmark was created by three independent SPARQL experts, which provided us with (1) a natural-language query and (2) the equivalent conjunctive SPARQL query. We selected these three datasets because they are a fragment of the well interlinked biomedical fraction of the Linked Open Data Cloud[10] and thus represent an ideal case for the future structure of Linked Data sources. The detailed results of our evaluation are shown in Figure 3.7. We ran our SINA with and without OWL inferencing during the state space construction.

When ran without inferencing, our approach was able to correctly disambiguate 23 out of 25 (i.e. 92%) of the resources contained in the queries. For Q9 (resp. Q25), the correct disambiguation was only ranked third (resp. fifth). In the other two cases (i.e. Q10 and Q12), our approach simply failed to retrieve the correct disambiguation. This was due to the path between `Doxil` and `Bextra` not being found for Q10 as well as the mapping from `disease` to `side effect` not being used in Q12. Overall, we achieve an MRR of 86.1% without inferencing.

The MRR was 2% lower (not statistically significant) when including OWL inferencing due to the best resource disambiguation being ranked at the second position for three queries that were disambiguated correctly without inferencing (Q5, Q7 and Q20). This was simply due to the state space being larger and leading to higher transition probabilities for the selected resources. With respect to precision and recall achieved with and without reasoning, there were also no statistically significant differences between the

[9] The benchmark queries are available at `http://wiki.aksw.org/Projects/lodquery`
[10] For example, 859 `owl:sameAs` links exists between the 924 instances of drugs in Sider and the 4,772 instances of drugs Drugbank

two approaches. The approach without reasoning achieved a precision of 0.91 and a recall of 0.88 while using reasoning led to precision (resp. recall) values of 0.95 (resp. 0.90). Overall the pros and cons of using inferencing are clearly illustrated in the results of our experiments. On Q12, our approach is unable to construct a query without reasoning due to the missing equivalence between the terms `disease` and `side effect`. This equivalence is made available by the inference engine, thus making the construction of the SPARQL query possible. On the downside, adding supplementary information through inferencing alters the ranking of queries and can thus lead to poorer recall values as in the case of Q20.

3.6.2 Accuracy Benchmark over DBpedia

DBpedia [23] the large knowledge base extracted from Wikipedia is an ideal test case with respect to size. There is no standard evaluation benchmark for keyword search over RDF data yet. However, there are the *QALD-1*, *QALD-2* and *QALD-3* benchmarks[11] tailored towards comparing question answering systems for natural language queries[12] [67]. We employed the *QALD-3* test dataset (in order to compare with the recent systems participated in the campaign) and the *QALD-1* dataset (in order to test Sina more) for evaluation (note that *QALD-2* dataset was employed for bootstrapping). Basically, training datasets were used for tuning and debugging Sina. The QALD-3 test dataset (and the QALD-1 benchmark) consist of 100 (respectively 50) questions in natural language that were also formulated as SPARQL queries. The questions are of different levels of complexity. Generally, the reasons for failures are as follows:

1. *Complex questions.* Questions containing quantifiers, comparatives and superlatives.
2. *Coverage.* Questions requiring information from beyond DBpedia ontology, i.e., from `YAGO` or `FOAF`.
3. *Query expansion.* Examples are the keywords *"wife"*, which should be matched to *"spouse"* and *"daughter"* to *"child"*.
4. *Query cleaning.* An example is the question *"Through which countries does the Yenisei river flow?"* The keyword *flow* is not a stop word but does not have any matched resource in the corresponding SPARQL query of the benchmark; and should therefore be ignored.

Since query expansion and cleaning might result in more noisy input for the model, we did not address these in this work[13]. With respect to the *QALD-3* test dataset, we take all 100 questions in English natural language without changes in the original question into account. Accordingly, Sina can correctly answer 32 questions with precision 0.32, recall 0.32, F-measure 0.32 and of these 32 questions, average MRR is 0.87. Thus, it outperforms the most state-of-the-art question answering systems[14]. Note that our approach is limited to conjunctive queries. Regarding the failures, 14 questions were beyond DBpedia coverage; 43 queries had query expansion or query cleaning issues; 7 questions were complex questions. From the remaining questions, in two cases matching to the appropriate resource failed, e.g. due to using abbreviation. In the other two cases, the constructed query does not fulfill the information need of the question. The detailed results of this evaluation are shown in Figure 3.8.

In the sequel, we elaborate on the result of the evaluation using *QALD-1* benchmark. We excluded 7 complex questions and 13 questions requiring information beyond DBpedia, i.e., from `YAGO` and

[11] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`

[12] Sina did not participate in the running campaign due to the late preparation, but we used these benchmarks for own evaluation.

[13] A careful extension of our approach and analysis of the results will be required.

[14] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/3/qald3_results.pdf`

`FOAF`. From the remaining questions 14 require query expansion or query cleaning to map keywords to resources correctly. In order to keep these 14 questions while reducing the effect of external parameters (i.e. expansion and cleaning) in our evaluation analysis, we slightly modified the expression of these 14 questions in such a way that expansion and cleaning is not required anymore. For instance, the query *"Through which countries does the Yenisei river flow?"* was slightly rephrased as `countries of the Yenisei river`.

Figure 3.9 shows the results of the accuracy study for each of the 30 questions used in our evaluation. Only the expression of questions printed with pink background has been modified compared to QALD-1. Out of the 30 questions, our disambiguation method precisely identifies resources for 27 questions with an *MRR* higher than 96%. Also, the query graph construction method correctly constructs the query graphs retrieving answers for 25 questions exactly as required by the benchmark with precision 0.5, recall 0.5 and F-measure 0.5.

In the following, we discuss the reasons of failures. In the case of Q30, the failure is caused by data quality (i.e., inconsistent domain and range information in DBpedia). Although suitable resources have been identified i.e. `dbp:creator` and `dbr:Goofy`, generating a suitable SPARQL query failed due to a conflict between the domain of the property `dbp:creator` and the type of the entity `dbr:Goofy`. According to the DBpedia schema the domain of the property `dbp:creator` is `dbo:Work` but the type of the entity `dbr:Goofy` has been declared as `dbo:Person`. In other words, there is no intersection between *CT* of `dbr:Goofy` and *OP* of `dbp:creator`. In DBpedia, `dbr:Goofy` is with the type `dbo:Person` wrongly placed as the subject of a triple with property `dbp:creator`. In the case of the Q37, the constructed query is not matched to the information need of the question. The constructed query contains the following triples: `1. dbr:Bill-Clinton dbp:child ?c.`
`2. dbr:Bill-Clinton  dbp:spouse ?s.` Our approach fails to identify appropriate resources for Q1, Q2 and Q48. This failure is mainly due to the difficulty of recognizing the correct data granularity and consequently identifying the right resources. The question `'When did Germany join the EU?'` needs query expansion. In fact, the segment `'join the EU'` should be mapped to the property `dbp:accessioneudate`. It is very unlikely that a user expresses a query close to the label of that property. In the case of Q1, the segment `'computer software'` should be mapped to a literal, whereas currently the only kind of literal included for looking up is `rdfs:label`. Improving this is part of our extension agenda. Q2 is very complex. For the given segment `'located'` the corresponding SPARQL query contains three triple patterns with the properties `dbo:location`, `dbp:location` and `dbp:locationCountry`. Since we assume that each of the known segments in the benchmark could be mapped to exactly one resource in the SPARQL query, our approach is not yet able to handle such questions. Q30 does not aim at retrieving any resources (it uses `ask` instead of `select`). Although our method successfully identifies correct resources and triple patterns, this question was not considered a correct query.

3.6.3 Runtime Benchmark

Although performance was not (yet) the primary focus of our work, we also wanted to provide evidence that our approach can be used for real-time querying. In the sequential version of Sina, all the requests to the knowledge bases are performed sequentially. Our intuition was that the runtime can be optimized by parallelizing those requests. In other to speedup runtime, we thus implemented parallelization over three components, i.e., *segment validation*, *resource retrieval* and *query construction*. We evaluated the runtime of our approach on the life-science as well as on QALD-1 benchmark queries. All experiments were carried out on a Windows 7 machine with an Intel Core M 620 processor, 6GB of RAM and a 230GB SSD. Table 3.3 shows the average runtime of Sina over DBpedia and the life science datasets (with and

#K	DBpedia			Life Science			Life Science with Inf.		
	Seq	*Par*	*%Gain*	*Seq*	*Par*	*%Gain*	*Seq*	*Par*	*%Gain*
2	57.3	17.5	69.3	3.7	3.4	3.4	4.4	3.8	12.0
3	38.3	9.3	75.4	13.4	8.5	36.0	13.5	8.3	38.8
4	30.3	12.5	58.6	68.9	54.9	21.3	104.5	86.4	17.3
5	38.9	18.5	52.3	66.3	59.0	10.9	128.8	112.0	13.0
6	-	-	-	493.1	452.8	8.1	94.7	61.4	35.1
7	39.6	20.5	48.0	-	-	-	-	-	-

Table 3.3: SINA performance in seconds in sequential mode (*Seq*) and with parallelization (*Par*) for different number of keywords (K) and datasets.

without inferencing during state space construction) for three runs. Through employing parallelization, we achieved a total performance gain of 60% for DBpedia, 16.8% for life science without inferencing and 23.3% for life science with inferencing.

Although most life science queries without inferencing take less time than with inferencing their performance gain is larger. This is due to the fact that the time saved in both versions is almost the same. For instance, the average performance gain for six keyword queries in the life science scenario with and without inferencing is approximately **40** and **33.3** seconds.

The observed results of runtime show that number of input keywords is not correlated to the achieved gain or query execution time. There are several explanations: (1) The number of keywords does not directly affect the number of requests sent to the knowledge base. In fact, the number of initial requests sent to the knowledge base for resource retrieval is correlated to the number of valid segments (not input keywords) while the number of valid segments is not depended on the number of input keywords. For instance, for two keywords *A* and *B*, we may have three valid segments as (A, B, AB) while for three keywords A, B and C we may have only two valid segments as (A, BC). (2) The size of the state space is not related to the size of input keywords. For instance, a keyword *A* can retrieve more resources than a keyword *B*, which means that a query containing keyword *A* will demand more time than a query containing keyword *B*. (3) There are also external factors that affect the query execution runtime such as the knowledge base indexing strategy.

3.7 Related Work

We analyze semantic search approaches in five dimensions, i.e., input query format, disambiguation, expansion, data distribution and query transformation. With respect to the first dimension, there are two common types of input query, i.e., natural language query and keyword query. There is a contradiction in usability studies of these two types of input queries. While [68] shows that users prefer using natural language queries to keywords, [69] presents that students prefer keyword query. Second dimension is using a disambiguation approach which selects the best interpretation of the input query. Third dimension is query expansion like taking into account synonyms in order to improve retrieval performance. Fourth dimension is related to the number of the underlying knowledge bases, whether the search engine runs on either a single knowledge base or multiple interlinked knowledge bases. The last dimension refers on how to transform the input query to a formal query. Several approaches have been developed for this transformation, i.e., document-centric, entity-centric and question-answering approaches.

Most of these approaches are adaptations of document retrieval approaches. For instance, *Swoogle* [28], *Watson* [29], *Sindice* [26] stick to the document-centric paradigm. Entity-centric approaches (e.g. *Sig.Ma* [27], *Falcons* [40], *SWSE* [39]) have recently emerged. However, the basis for all these services

Systems	Input query	Disam.	Expan.	Data dis.	Transformation
Sina	NL & Keyword	✓	-	Multiple	Question answering
Sindice	Keyword	-	-	Multiple	Document retrieval
Sigma	Keyword	-	-	Multiple	Entity retrieval
Swoogle	Keyword	-	-	Multiple	Document retrieval
PowerAqua	NL	✓	✓	Multiple	Question answering
TBSL	NL	✓	✓	Single	Question answering

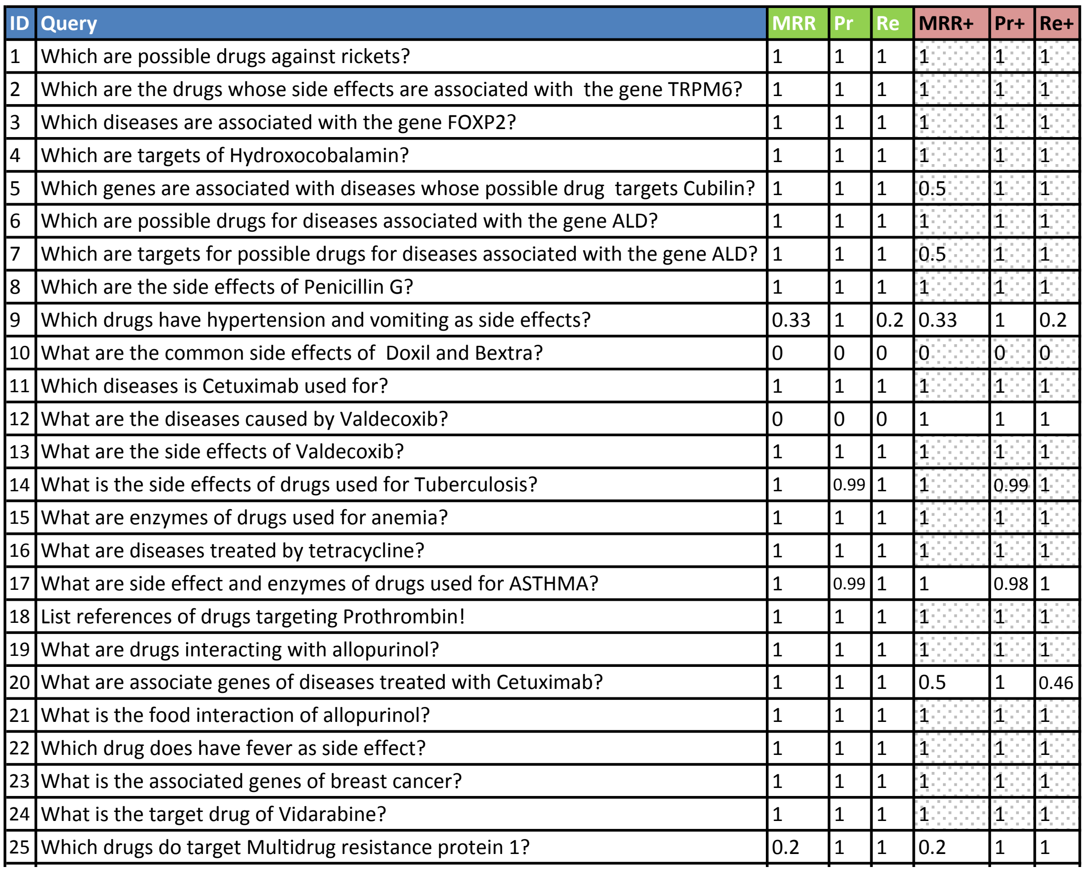
Table 3.4: Comparison of Semantic Web search engines in five dimensions.

are keyword indexing and retrieval relying on the matching user keywords and indexed terms. Examples of question answering systems are *PowerAqua* [41] and *OntoNL* [31]. *PowerAqua* can automatically combine information from multiple knowledge bases at runtime. The input is a natural language query and the output is a list of relevant entities. PowerAqua lacks a deep linguistic analysis and can not handle complex queries. *Pythia* [42] is a question answering system that employs deep linguistic analysis. It can handle linguistically complex questions, but is highly dependent on a manually created lexicon. Therefore, it fails with datasets for which the lexicon was not designed. Pythia was recently used as kernel for *TBSL* [62], a more flexible question-answering system that combines Pythia's linguistic analysis and the *BOA framework* [45] for detecting properties to natural language patterns. Exploring schema from anchor points bound to input keywords is another approach discussed in [48]. Querying Linked datasets is addressed with the work mainly treat both the data and queries as bags of words [40, 70]. [71] presents a hybrid solution for querying linked datasets. It runs the input query against one particular dataset regarding the structure of data, then for candidate answers, it finds and ranks the linked entities from other datasets. Table 3.4 compares six Semantic Web search engines in five dimensions. Our approach is a prior work as it queries all the datasets at hand and then according to the structure of the data, it makes a federated query. Furthermore, our approach is independent of any linguistic analysis and does not fail when the input query is an incomplete sentence.

Segmentation and disambiguation are inherent challenges of keyword-based search. Keyword queries are usually short and lead to significant keyword ambiguity [72]. Segmentation has been studied extensively in the natural language processing (NLP) literature (e.g., [73]). NLP techniques for chunking such as part-of-speech tagging or name entity recognition cannot achieve high performance when applied to query segmentation. [74] addresses the segmentation problem as well as spelling correction and employs a dynamic programming algorithm based on a scoring function for segmentation and cleaning. An unsupervised approach to query segmentation in Web search is described in [75]. [76] is a supervised method based on Conditional Random Fields (CRF) whose parameters are learned from query logs. For detecting named entities, [77] uses query log data and Latent Dirichlet Allocation. In addition to query logs, various external resources such as Web pages, search result snippets, Wikipedia titles and a history of the user activities have been used [78–81]. Still, the most common approach is using the context for disambiguation [82–84]. In this work, resource disambiguation is based on the structure of the knowledge at hand as well as semantic relations between the candidate resources mapped to the keywords of the input query.

3.8 Discussion and Conclusion

The result of evaluation shows the effectiveness as well as scalability of this approach. In the current implementation forward chaining is carried out on the fly. Consequently, the runtime can be further significantly increased by pre-processing the knowledge base, adding all statements that can be generated via forward chaining and constructing an index for the label information. After implementing further performance optimizations (e.g. caching computed values such as resource distances), we expect our implementation to require less than 10s for up to 5 keywords. Note that a main assumption of this work is that some schema information is available for the underlying knowledge base and resources are typed according to the schema. Regarding the disambiguation, the superiority of our model is related to the transition probabilities. We achieved a fair balance between the qualification of states for transiting by reflecting the popularity and distance in the hub and authority values and setting a transition probability to the unknown entity state (depending on the hub value).

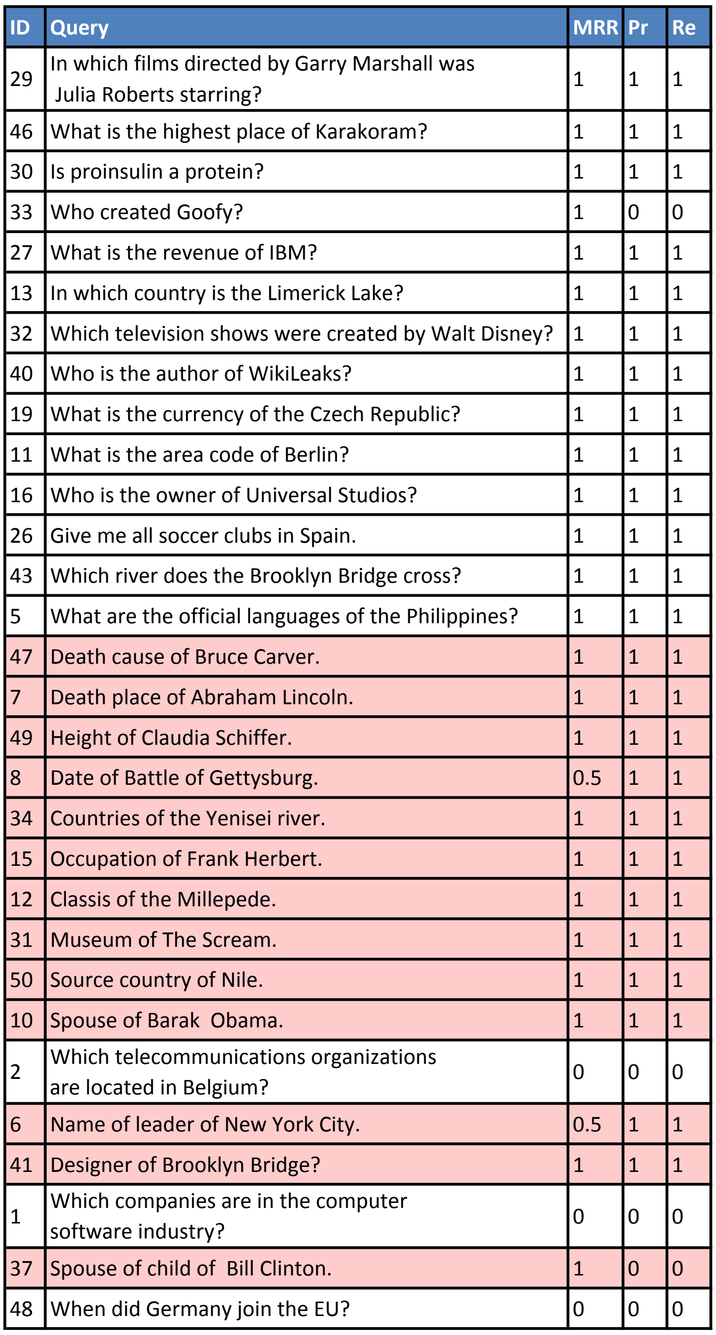


ID	Query	MRR	Pr	Re	MRR+	Pr+	Re+
1	Which are possible drugs against rickets?	1	1	1	1	1	1
2	Which are the drugs whose side effects are associated with the gene TRPM6?	1	1	1	1	1	1
3	Which diseases are associated with the gene FOXP2?	1	1	1	1	1	1
4	Which are targets of Hydroxocobalamin?	1	1	1	1	1	1
5	Which genes are associated with diseases whose possible drug targets Cubilin?	1	1	1	0.5	1	1
6	Which are possible drugs for diseases associated with the gene ALD?	1	1	1	1	1	1
7	Which are targets for possible drugs for diseases associated with the gene ALD?	1	1	1	0.5	1	1
8	Which are the side effects of Penicillin G?	1	1	1	1	1	1
9	Which drugs have hypertension and vomiting as side effects?	0.33	1	0.2	0.33	1	0.2
10	What are the common side effects of Doxil and Bextra?	0	0	0	0	0	0
11	Which diseases is Cetuximab used for?	1	1	1	1	1	1
12	What are the diseases caused by Valdecoxib?	0	0	0	1	1	1
13	What are the side effects of Valdecoxib?	1	1	1	1	1	1
14	What is the side effects of drugs used for Tuberculosis?	1	0.99	1	1	0.99	1
15	What are enzymes of drugs used for anemia?	1	1	1	1	1	1
16	What are diseases treated by tetracycline?	1	1	1	1	1	1
17	What are side effect and enzymes of drugs used for ASTHMA?	1	0.99	1	1	0.98	1
18	List references of drugs targeting Prothrombin!	1	1	1	1	1	1
19	What are drugs interacting with allopurinol?	1	1	1	1	1	1
20	What are associate genes of diseases treated with Cetuximab?	1	1	1	0.5	1	0.46
21	What is the food interaction of allopurinol?	1	1	1	1	1	1
22	Which drug does have fever as side effect?	1	1	1	1	1	1
23	What is the associated genes of breast cancer?	1	1	1	1	1	1
24	What is the target drug of Vidarabine?	1	1	1	1	1	1
25	Which drugs do target Multidrug resistance protein 1?	0.2	1	1	0.2	1	1

Figure 3.7: MRR, precision and recall for the life science benchmark.

ID	Query	MRR	Pr	Re
2	Who was the successor of John F. Kennedy?	0.5	1	1
15	What is the longest river?	1	1	1
21	What is the capital of Canada?	0.5	1	1
22	Who is the governor of Wyoming?	1	1	1
24	Who was the father of Queen Elizabeth II?	1	1	1
26	How many official languages are spoken on the Seychelles?	1	1	1
30	What is the birth name of Angela Merkel?	1	1	1
33	Give me all Australian nonprofit organizations.	1	1	1
35	Who developed Minecraft?	1	1	1
39	Give me all companies in Munich.	1	1	1
40	List all games by GMT.	1	1	1
41	Who founded Intel?	0.33	1	1
43	Give me all breeds of the German Shepherd dog.	1	1	1
54	What are the nicknames of San Francisco?	1	1	1
56	When were the Hells Angels founded?	1	1	1
57	Give me the Apollo 14 astronauts.	1	1	1
58	What is the time zone of Salt Lake City?	1	1	1
60	Give me a list of all lakes in Denmark.	1	1	1
61	How many space missions have there been?	0.2	1	1
62	Did Socrates influence Aristotle?	1	1	1
63	Give me all Argentine films.	1	1	1
64	Give me all launch pads operated by NASA.	1	1	1
65	Which instruments did John Lennon play?	1	1	1
73	How many children did Benjamin Franklin have?	1	1	1
76	List the children of Margaret Thatcher.	1	1	1
78	Was Margaret Thatcher a chemist?	1	1	1
81	Which books by Kerouac were published by Viking Press?	0.2	1	1
86	What is the largest city in Australia?	1	1	1
87	Who composed the music for Harold and Maude?	1	1	1
88	Which films starring Clint Eastwood did he direct himself?	0.25	1	1
90	Where is the residence of the prime minister of Spain?	1	1	1
100	Who produces Orangina?	1	1	1

Figure 3.8: MRR, precision and recall for QALD-3 questions on DBpedia.



ID	Query	MRR	Pr	Re
29	In which films directed by Garry Marshall was Julia Roberts starring?	1	1	1
46	What is the highest place of Karakoram?	1	1	1
30	Is proinsulin a protein?	1	1	1
33	Who created Goofy?	1	0	0
27	What is the revenue of IBM?	1	1	1
13	In which country is the Limerick Lake?	1	1	1
32	Which television shows were created by Walt Disney?	1	1	1
40	Who is the author of WikiLeaks?	1	1	1
19	What is the currency of the Czech Republic?	1	1	1
11	What is the area code of Berlin?	1	1	1
16	Who is the owner of Universal Studios?	1	1	1
26	Give me all soccer clubs in Spain.	1	1	1
43	Which river does the Brooklyn Bridge cross?	1	1	1
5	What are the official languages of the Philippines?	1	1	1
47	Death cause of Bruce Carver.	1	1	1
7	Death place of Abraham Lincoln.	1	1	1
49	Height of Claudia Schiffer.	1	1	1
8	Date of Battle of Gettysburg.	0.5	1	1
34	Countries of the Yenisei river.	1	1	1
15	Occupation of Frank Herbert.	1	1	1
12	Classis of the Millepede.	1	1	1
31	Museum of The Scream.	1	1	1
50	Source country of Nile.	1	1	1
10	Spouse of Barak Obama.	1	1	1
2	Which telecommunications organizations are located in Belgium?	0	0	0
6	Name of leader of New York City.	0.5	1	1
41	Designer of Brooklyn Bridge?	1	1	1
1	Which companies are in the computer software industry?	0	0	0
37	Spouse of child of Bill Clinton.	1	0	0
48	When did Germany join the EU?	0	0	0

Figure 3.9: MRR, precision and recall for QALD-1 questions on DBpedia.

CHAPTER 4

Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features[1]

Abstract: Effective search in structured information based on textual user input is of high importance in thousands of applications. Query expansion methods augment the original query of a user with alternative query elements with similar meaning to increase the chance of retrieving appropriate resources. In this work, we introduce a number of new query expansion features based on semantic and linguistic inferencing over Linked Open Data. We evaluate the effectiveness of each feature individually as well as their combinations employing several machine learning approaches. The evaluation is carried out on a training dataset extracted from the QALD question answering benchmark. Furthermore, we propose an optimized linear combination of linguistic and lightweight semantic features in order to predict the usefulness of each expansion candidate. Our experimental study shows a considerable improvement in precision and recall over baseline approaches.

4.1 Introduction

With the growth of the Linked Open Data cloud, a vast amount of structured information was made publicly available. Querying that huge amount of data in an intuitive way is challenging. SPARQL, the standard query language of the semantic web, requires exact knowledge of the vocabulary and is not accessible by laypersons. Several tools and algorithms have been developed that make use of semantic technologies and background knowledge [85], such as *TBSL* [62], *SINA* [61] and Wolfram|Alpha[2]. Those tools suffer from a mismatch between query formulation and the underlying knowledge base structure that is known as the *vocabulary problem* [86]. For instance, using DBpedia as knowledge base, the query “Who is married to Barack Obama?” could fail, because the desired property in DBpedia is labeled “spouse” and there is no property labeled “married to”.

Automatic query expansion (AQE) is a tried and tested method in web search for tackling the vocabulary problem by adding related words to the search query and thus increase the likelihood that appropriate documents are contained in the result. The query is expanded with features such as synonyms, e.g. “spouse” and “married to” in the example above, or *hyponym-hypernym* relations, e.g. “red” and

[1] Corresponding publication is: Saeedeh Shekarpour, Konrad Höffner, Jens Lehmann, and Sören Auer, “Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features”, *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, 2013 pp. 191–197

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

"color". We investigate, which methods semantic search engines can use to overcome the vocabulary problem and how effective AQE with the traditional linguistic features is in this regard. Semantic search engines can use the graph structure of RDF and follow interlinks between datasets. We employ this to generate additional expansion features such as the labels of sub- and superclasses of resources. The underlying research question is whether interlinked data and vocabularies provide features which can be taken into account for query expansion and how effective those new semantic query expansion features are in comparison to traditional linguistic ones.

We do this by using machine learning methods to generate a linear combination of linguistic and semantic query expansion features with the aim of maximizing the F_1-score and efficiency on different benchmark datasets. Our results allow developers of new search engines to integrate AQE with good results without spending much time on its design. This is important, since query expansion is usually not considered in isolation, but rather as one step in a pipeline for question answering or keyword search systems.

Our core contributions are as follows:

- Definition of several semantic features for query expansion.
- Creation of benchmark test sets for query expansion.
- Combining semantic and linguistic features in a linear classifier.
- An analysis of the effect of each feature on the classifier as well as other benefits and drawbacks of employing semantic features for query expansion.

The paper is structured as follows: In section 4.2, the overall approach is described, in particular the definition of features and the construction of the linear classifier. Section 4.3 provides the experiment on the QALD-1, QALD-2 and QALD-3 test sets and presents the results we obtained. In the related work in section 5.5, we discuss in which settings AQE is used. Finally, we conclude and give pointers to future work.

4.2 Approach

In document retrieval, many query expansion techniques are based on information contained in the top-ranked retrieved documents in response to the original user query, e.g. [87]. Similarly, our approach is based on performing an initial retrieval of resources according to the original keyword query. Thereafter, further resources are derived by leveraging the initially retrieved ones. Overall, the proposed process depicted in Figure 4.1 is divided into three main steps. In the first step, all words closely related to the original keyword are extracted based on two types of features – linguistic and semantic. In the second step, various introduced linguistic and semantic features are weighted using learning approaches. In the third step, we assign a relevance score to the set of the related words. Using this score we prune the related word set to achieve a balance between *precision* and *recall*.

4.2.1 Extracting and Preprocessing of Data using Semantic and Linguistic Features

For the given input keyword k, we define the set of all words related to the keyword k as $X_k = \{x_1, x_2, ..., x_n\}$. The set X_k is defined as the union of the two sets LE_k and SE_k. LE_k (resp. SE_k) is constructed as the collection of all words obtained through linguistic features (resp. semantic). Linguistic features extracted from WordNet are:

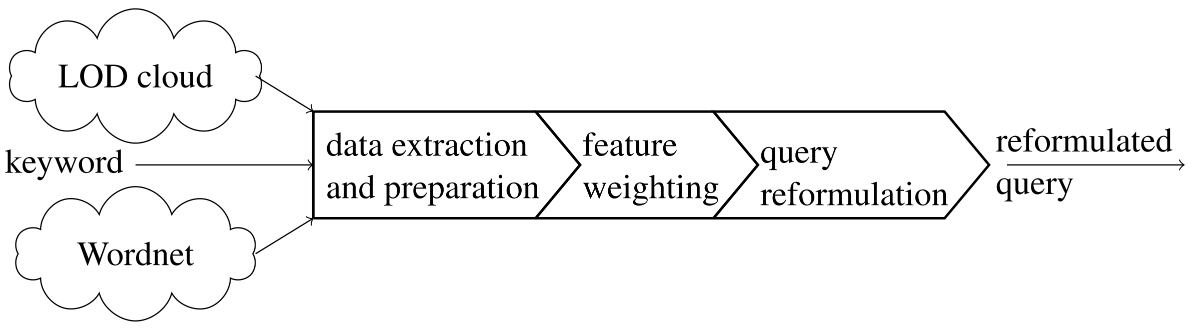


Figure 4.1: AQE Pipeline.

- *synonyms:* words having a similar meanings to the input keyword k.
- *hyponyms:* words representing a specialization of the input keyword k.
- *hypernyms:* words representing a generalization of the input keyword k.

The set SE comprises all words semantically derived from the input keyword k using Linked Data. To form this set, we match the input keyword k against the *rdfs:label* property of all resources available as Linked Open Data[3]. It returns the set $AP_k = \{r_1, r_2, ..., r_n\}$ as $AP_k \subset (C \cup I \cup P)$ where C, I and P are the sets of classes, instances and properties contained in the knowledge base respectively, whose labels contain k as a sub-string or are equivalent to k. For each $r_i \in AP_k$, we derive the resources semantically related to r_i by employing a number of semantic features. These semantic features are defined as the following semantic relations:

- *sameAs:* deriving resources having the same identity as the input resource using *owl:sameAs*.
- *seeAlso:* deriving resources that provide more information about the input resource using *rdfs:seeAlso*.
- *class/property equivalence:* deriving classes or properties providing related descriptions for the input resource using *owl:equivalentClass* and *owl:equivalentProperty*.
- *superclass/-property:* deriving all super classes/properties of the input resource by following the *rdfs:subClassOf* or
  rdfs:subPropertyOf property paths originating from the input resource.
- *subclass/-property:* deriving all sub resources of the input resource r_i by following the *rdfs:subClassOf* or
  rdfs:subPropertyOf property paths ending with the input resource.
- *broader concepts:* deriving broader concepts related to the input resource r_i using the SKOS vocabulary properties *skos:broader* and *skos:broadMatch*.
- *narrower concepts:* deriving narrower concepts related to the input resource r_i using *skos:narrower* and *skos:narrowMatch*.
- *related concepts:* deriving related concepts to the input resource r_i using *skos:closeMatch*, *skos:mappingRelation* and *skos:exactMatch*.

[3] via `http://lod.openlinksw.com/sparql`

Note that on a given r_i only those semantic features are applicable which are consistent with its associated type. For instance, super/sub class/property relations are solely applicable to resources of type *class* or *property*.

For each $r_i \in AP_k$, we derive all the related resources employing the above semantic features. Then, for each derived resource r', we add all the English labels of that resource to the the set SE_k. Therefore, SE_k contains the labels of all semantically derived resources. As mentioned before, the set of all related words of the input keyword k is defined as $X_k = LE_k \cup SE_k$. After extracting the set X_k of related words, we run the following preprocessing methods for each $x_i \in X_k$:

1. *Tokenization:* extraction of individual words, ignoring punctuation and case.
2. *Stop word removal:* removal of common words such as articles and prepositions.
3. *Word lemmatisation:* determining the lemma of the word.

Vector space of a word

A single word $x_i \in X_k$ may be derived via different features. For example, as can be observed in Figure 4.2, the word "motion picture" and "film" is derived by *synonym*, *sameAs* and *equivalent* relations. Thus, for each derived word x_i, we define a vector space representing the derived features resulting in including that word. Suppose that totally we have n linguistic and semantic features. Each $x_i \in X_k$ is associated with a vector of size n as $V_{x_i} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$. Each α_i represents the presence or absence of the word x_i in the list of the words derived via the feature f_i. For instance, if we assume that f_1 is dedicated to the *synonym* feature, the value 1 for α_1 in the V_{x_i} means that x_i is included in the list of the words derived by the *synonym* feature. There are features and those features generate a set of expansion words.

4.2.2 Feature Selection and Feature Weighting

In order to distinguish how effective each feature is and to remove ineffective features, we employ a weighting schema ws for computing the weights of the features as $ws : f_i \in F \rightarrow w_i$. Note that F is the set of all features taken into account. There are numerous feature weighting methods to assign weight to features like information gain [88], weights from a linear classifier [89], odds ratio, etc. Herein, we consider two well-known weighting schemas.

Information Gain (IG)

Information gain is often used (see section 5.5) to decide which of the features are the most relevant. We define the information gain (IG) of a feature as:

$$\mathrm{IG}(f_i) = \sum_{\substack{c\in\{+,-\}\\ f_i\in\{present,absent\}}} \Pr(c, f_i) \ln \frac{\Pr(c, f_i)}{\Pr(f_i)\Pr(c)}$$

In our approach, we used the ID3 decision tree algorithm with information gain.

Feature weights from linear classifiers

Linear classifiers, such as for example SVMs, calculate predictions by associating the weight w_i to the feature f_i. Features whose w_i is close to 0 have a small influence on the predictions. Therefore, we can assume that they are not very important for query expansion.

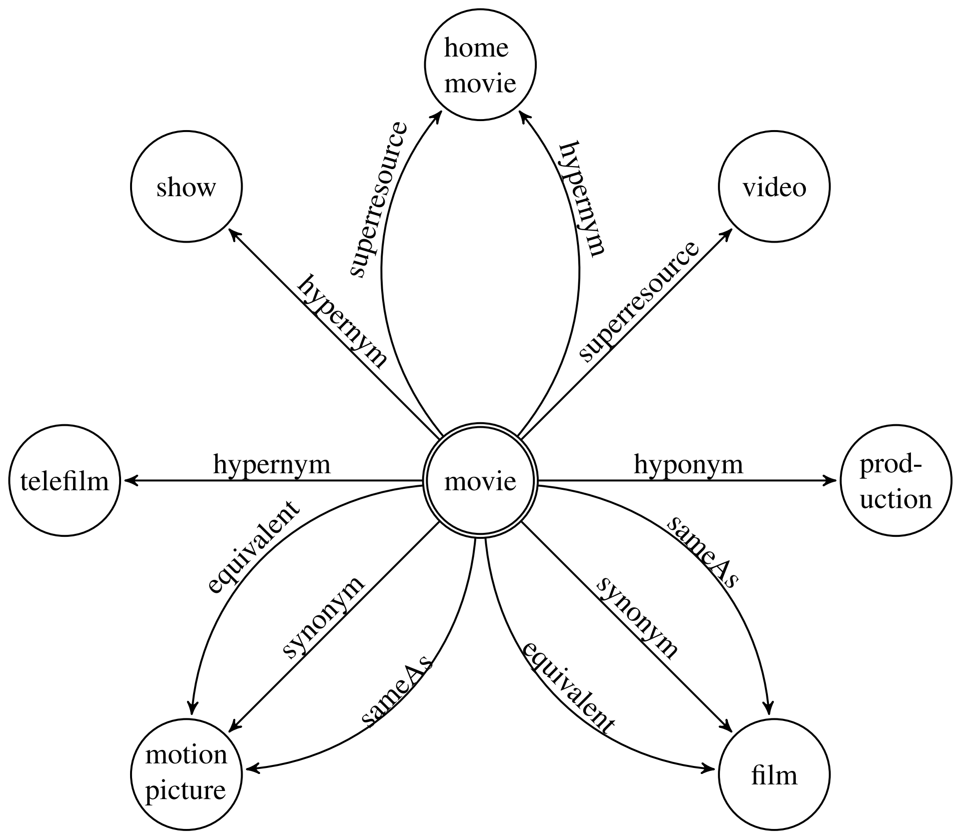


Figure 4.2: Exemplary expansion graph of the word *movie* using semantic features.

4.2.3 Setting the Classifier Threshold

As a last step, we set the threshold for the classifiers above. To do this, we compute the relevance score value score(x_i) for each word $x_i \in X_k$. Naturally, this is done by combining the feature vector $V_{x_i} = [\alpha_1, \alpha_2, \dots, \alpha_n]$ and the feature weight vector $W = [w_1, w_2, \dots, w_n]$ as follows:

$$\text{score}(x_i) = \sum_{i=1:n} \alpha_i w_i$$

.

We can then compute a subset Y_k of X_k by selecting all words, which are above a threshold θ:

$$Y_k = \{y | y \in X_k \land \text{score}(y) \geq \theta\}$$

.

This reduces the set of query expansion candidates X_k to a set Y_k containing only the words in X_k above threshold θ. Since we use a linear weighted combination of features, words which exhibit one or more highly relevant features are more likely to be included in the query expansion set Y_k. Thus, the threshold can be used to control the trade-off between precision and recall. A high threshold means higher precision, but lower recall whereas a low threshold improves recall at the cost of precision.

To sum up, Y_k is the output of the AQE system and provides words semantically related to the input keyword k. Retrieval of resources from the underlying knowledge base is based on the match of a given keyword with the *rdfs:label* of resources. Thus, this expansion increases the likelihood of recognizing more relevant resources. Because in addition to resources matching their *rdfs:label* to the input keyword k, we take into account resources having match of their *rdfs:label* with the words contained in Y. Subsequently, this causes an increase in *recall*. On the contrary, it may result in loss of *precision* by including resources which may be irrelevant. A severe loss in *precision* significantly hurts retrieval performance. Therefore, we investigate a moderate tradeoff between speed and accuracy, although the final result highly depends on requirements of the search service(precision is more important or recall). Herein, the set Y includes all $x_i \in X$ with a high relevance likelihood and excludes those with a low likelihood.

4.3 Experiment and Result

4.3.1 Experimental Setup

The goal of our evaluation is to determine (1) How effective linguistic as well as semantic query expansion features perform and (2) How well a linear weighted combination of features performs. To the best of our knowledge, no benchmark has been created so far for query expansion tasks over Linked Data. Thus, we created one benchmark dataset. This dataset called QALD benchmark contains 37 keyword queries obtained from the *QALD-1* and *QALD-2* benchmarks[4]. The *QALD-1* and *QALD-2* benchmarks are essentially tailored towards comparing question answering systems based on natural language queries. We extracted all those keywords contained in the natural language queries requiring expansion for matching to the target knowledge base resource.

An example are the keywords "wife" and "husband" which should be matched to *dbpedia-owl:spouse*. Note that in the following all experiments are done on this dataset except the last experiment.

Features	**derived words**	**matches**
synonym	503	23
hyponym	2703	10
hypernym	657	14
sameAs	2332	12
seeAlso	49	2
equivalent	2	0
super class/property	267	4
sub class/property	2166	4
broader concepts	0	0
narrower concepts	0	0
related concepts	0	0

Table 4.1: Number of the words derived by each feature and their associated matches.

4.3.2 Results

Generally, when applying expansion methods, there is a risk of yielding a large set of irrelevant words, which can have a negative impact on further processing steps. For this reason, we were interested in

[4] `http://www.sc.cit-ec.uni-bielefeld.de/qald-`n for $n = 1, 2$.

measuring the effectiveness of all linguistic as well as semantic features individually in order to decide which of those are effective in query expansion. Table 4.1 shows the statistics over the number of derived words and the number of matches per feature. Interestingly, *sameAs* has even more matches than *synonym*, but also leads to a higher number of derived words. The *hyponym* and *sub class/property* return a huge number of derived words while the number of matches are very low. The features *hypernym* and *super class/property* in comparison to the rest of the features result in a considerable number of matches. The features *broader concepts*, *narrower concepts* and *related concepts* provide a very small amount of derived words and zero number of matches. Thus, we exclude the *skos* features in the later experiments.

In continuation, we investigate the accuracy of each individual feature. Thus, we employ a *SVM* classifier and individually make an evaluation over the accuracy of each feature. This experiment was done on the dataset with 10 fold cross validation. Table 4.4 presents the results of this study. In this table, the *precision*, *recall* and *F-Measure* for the positive (+), negative (-) and all (total) examples are shown. In addition, six separate evaluations are carried out over a subset of features which are semantically close to each other e.g. *hyponym+sub class/property*.

In the following, we only mention the most prominent observations of this study. The features *hyponym*, *super class/property* and *sub class/property* have the highest value for *F-Measure*. The *precision* of *sameAs* is the same as for *synonym*. The feature *equivalent* has a high *precision* although the *recall* is very low. The *precision* of the *sub class/property* and *hyponym* is high for negative examples. At last, the combined features always behave better than the individual features.

Feature	**GR**	**SVM**	**IG**
synonym	0.92	0.4	0.3
hyponym	0.4	0.81	0.49
hypernym	1	0.4	0.67
sameAs	0.4	0.8	0.49
seeAlso	0.4	1.36	0.3
equivalent	0.3	0.49	0.3
super class/property	0.45	1.45	0.7
sub class/property	1.5	0.67	1.12

Table 4.2: Computed weights of the features using the schemas *Gain Ratio (GR)*, *Support Vector Machines (SVM)* and *Information Gain (IG)*.

The second goal of our experimental study is how well a linear weighted combination of features can predict the relevant words? To do that, firstly we employed two weighting schemas as described in the approach, i.e. information gain (IG) and the weighting of the linear classifier *SVM*. Secondly, these computed weights are used for reformulating the input keyword.

Table 4.2 shows the weights computed by *SVM* schemas. The feature *super class/property* is ranked as the highest distinguishing feature. The computed value of *Hyponym* is relatively high but this feature has a negative influence on all examples. Interestingly, in *SVM* schema *sameAs* and *seeAlso* as well as *synonym* are acquired the equal values. This may result in that *sameAs* and *seeAlso* can be a comparable alternative for *synonym*. Furthermore, *subproperty* and *subclass* are excluded in this schema.

Thereafter, we scored all the derived words according to the beforehand computed weights. We set up two different settings. In each setting, respectively, only linguistic features and only semantic features are taken into account. A separate evaluation is carried out for each setting with respect to the computed weights *SVM*.

Table 4.3 shows the results of this study. Interestingly, the setting with only semantic features result in an accuracy at least as high as the setting with only linguistic features. This observation is an important finding of this paper that semantic features appear to be competitive with linguistic features.

Features	Weighting	P	Recall	F-Score
Linguistic	SVM	0.73	0.65	0.62
Semantic	SVM	0.68	0.63	0.6
Linguistic	Decision Tree / IG	0.588	0.579	0.568
Semantic	Decision Tree / IG	0.755	0.684	0.661

Table 4.3: Accuracy results of predicting the relevant derived words.

4.4 Related Work

Automatic Query Expansion (AQE) is a vibrant research field and there are many approaches that differ in the choice and combination of techniques.

4.4.1 Design Choices for Query Expansion

In the following, we describe the most important choices of data sources, candidate feature extraction methods and representations, feature selection methods and expanded query representations (cf. [90] for a detailed survey).

Data sources are organized collections of words or documents. The choice of the data source is crucial because it influences the size of the vocabulary as well as the available relations; and thus possible expansion features. Furthermore, a data source is often restricted to a certain domain and thus constitutes a certain context for the search terms. For example, corpora extracted from newspapers yield good results when expanding search queries related to news but are generally less suited for scientific topics. Thus a search query with the keyword “fields” intended for electromagnetic fields could be expanded to “football fields” and yield even worse results then the original query.

Popular choices for data sources are text corpora, synsets, hyponyms and hypernyms, anchor texts, search engine query logs or top ranked documents. Our approach uses both WordNet as a source for synonyms, hyponyms and hypernyms as well as the LOD cloud to obtain labels of related classes or properties, such as equivalent, sub- and super-resources (cf. section 4.2). WordNet is used frequently but it suffers from two main problems [90]:

1. There is a lack of proper nouns which we tackle by using the LOD cloud including DBpedia which contains mainly instances.

2. The large amount of ambiguous terms leads to a disambiguation problem. However, this does not manifest itself in the relatively small models of the benchmarks we used (cf. section 4.3). Disambiguation is not the focus of this work but may need to be addressed separately when our approach is used in larger domains.

Features	**P**	**Recall**	**F-Score**	
synonym	0.44	0.68	0.54	−
	0.33	0.15	0.21	+
	0.39	0.42	0.37	total
hyponym	0.875	0.368	0.519	−
	0.6	0.947	0.735	+
	0.737	0.658	0.627	total
hypernym	0.545	0.947	0.692	−
	0.8	0.211	0.333	+
	0.673	0.579	0.513	total
sameAs	0.524	0.579	0.55	−
	0.529	0.474	0.5	+
	0.527	0.526	0.525	total
seeAlso	0.471	0.842	0.604	−
	0.25	0.053	0.087	+
	0.36	0.447	0.345	total
equivalent	0.48	0.89	0.63	−
	0.33	0.053	0.091	+
	0.41	0.47	0.36	total
super class/property	0.594	1	0.745	−
	1	0.316	0.48	+
	0.797	0.658	0.613	total
sub class/property	0.48	0.89	0.63	−
	0.33	0.05	0.09	+
	0.52	0.41	0.47	total
sameAs, seeAlso, equivalent	0.5	0.579	0.53	−
	0.5	0.42	0.45	+
	0.5	0.5	0.49	total
synonym, sameAs, seeAlso, equivalent	0.47	0.579	0.52	−
	0.46	0.36	0.41	+
	0.47	0.47	0.46	total
hyponym, subresource	0.875	0.368	0.519	−
	0.6	0.947	0.735	+
	0.737	0.658	0.627	total
hypernym, superresource	0.594	1	0.745	−
	1	0.316	0.48	+
	0.797	0.658	0.613	total

Table 4.4: Separate evaluations of the precision, recall and f-score of each individual feature.

Feature selection [91] consists of two parts: (1) feature weighting assigns a scoring to each feature and (2) the feature selection criterion determines which of the features to retain based on the weights. Some common feature selection methods are *mutual information*, *information gain*, *divergence from randomness* and *relevance models*. A framework for feature weighting and selection methods is presented in [91]. The authors compare different feature ranking schemes and (although not the primary focus of the article) show that SVMs achieve the highest F_1-score of the examined methods. We use information gain and SVMs separately and compare the results. To learn the feature rankings we use the data mining software *Weka* [92].

Expanded query representation can take the form of a Boolean or structured query, vectors of concept types or unweighted terms and many others. Because our query is a set of keywords, our extended query is an extended set of keywords and thus consists of unweighted terms.

4.4.2 Semantic Search and Question Answering Engines

While AQE is prevalent in traditional web search engines, the semantic search engines we examine in the following either do not address the vocabulary problem or tackle it in a different way.

Table 4.5 shows how the participants of the QALD/ILD 2012 workshop and selected other approaches tackle the vocabulary problem. Interestingly, two of the three considered approaches did not use any kind of query expansion, relying instead only on exact string matching between the query keyword and the label of the associated resource. *Alexandria* [93] uses *Freebase* to include synonyms and different surface forms.

MHE[5] combines query expansion and entity recognition by using textual references to a concept and extracting Wikipedia anchor texts of links. For example, when looking at the following link:

```
<a href="http://en.wikipedia.org/
wiki/United_Nations">UN</a>
```

Here the word "UN" is mapped to the concept *United_Nations*. This approach takes advantage of a large amount of hand-made mappings that emerge as a byproduct. However, this approach is only viable for Wikipedia-DBpedia or other text corpora whose links are mapped to resources.

Engine	Method
TBSL [62]	WordNet synonyms and BOA pattern library [94]
PowerAqua [41]	WordNet synonyms and hypernyms, `owl:sameAs`
Eager [95]	resources of the same type using Wikipedia categories
Alexandria [93]	alternative names (synonyms and different surface forms) from Freebase [96]
SemSeK [97]	no AQE
QAKiS [98]	no AQE
MHE	Wikipedia anchor texts from links pointing to the concept

Table 4.5: Prevalence of AQE in RDF based search or question answering engines.

Eager [95] expands a set of resources with resources of the same type using DBpedia and Wikipedia categories (instead of linguistic and semantic features in our case). Eager extracts implicit category and synonym information from abstracts and redirect information, determines additional categories from the

[5] `http://ups.savba.sk/~marek`

DBpedia category hierarchy and then extracts additional resources which have the same categories in common.

PowerAqua [41] is an ontology-based system that answers natural language queries and uses WordNet synonyms and hypernyms as well as resources related with the `owl:sameAs` property.

A prerequisite of feature-based Query Expansion is, that the data about the features used, i.e. the pairs contained in the relations, is available. Relation equivalency (*owl:equivalentProperty*) links in particular are often not, or not completely, defined for a knowledge base. There is however an approach for mining equivalent relations from Linked Data, that relies on three measures of equivalency: *triple overlap*, *subject agreement* and *cardinality ratio*. [99]

An approach similar to ours is [100], however it relies on supervised learning and uses only semantic expansion features instead of a combination of both semantic and linguistic ones.

4.5 Conclusions

Semantic search is one of the most important applications for demonstrating the value of the semantic web to real users. In the last years, a number of approaches for semantic search have been introduced. However, other than in traditional search, the effect of query expansion has not yet been studied. With semantically structured knowledge, we can also employ additional semantic query expansion features. In this work, we performed a comprehensive study of semantic query expansion. We compared the effectiveness of linguistic and semantic query expansion as well as their combination. Based on a query expansion benchmark we created, our results suggest that semantic features are at least as effective as linguistic ones and the intelligent combination of both brings a boost in precision and recall.

CHAPTER 5

Query Reformulation on RDF Knowledge Bases using Hidden Markov Models[1]

Abstract: Textual querying is the most popular and simple way to retrieve information. Query expansion is a way to reformulate the input query in order to increase the chance of retrieving appropriate information. A common way is to derive words from linguistic resources (e.g. WordNet). However, taking all words derived from the input keywords into account significantly increases retrieval cost. We introduce a novel method for automatic query expansion with respect to background knowledge. Our method uses a Hidden Markov Model to determine the most suitable derived words from linguistic resources. We employ the co-occurrence of words in the background knowledge for connecting derived words. The evaluation is carried out on a test query set extracted from the QALD question answering benchmark. Our experimental study shows the feasibility and high accuracy of the method.

5.1 Introduction

There is increasingly much structured data being published on the Web. So far, more than 31 billion triples[2]) are publicly available. Yet, retrieving information is a challenge mainly due to the need of having accurate knowledge of the used vocabularies. Most of the schema-aware search systems, for example, question answering systems such as *TBSL* [62] and *SINA* [61], suffer from a *vocabulary mismatch* between the input query keywords and the underlying knowledge base schema [86]. For instance, using DBpedia as knowledge base, the query “Who is wife of Barack Obama?” could fail, because the desired property in DBpedia is labeled “spouse”.

Automatic query expansion is a long-standing research topic in information retrieval. It is a way of reformulating the input query in order to overcome the vocabulary mismatch problem. In case of a vocabulary mismatch, schema-aware search systems are unable to retrieve data. Therefore, query expansion can be a crucial step in question answering or keyword search pipeline. A naive way for automatic query expansion is adding words derived from linguistic resources. In this regard, expansions are *synonyms*, *hyponyms* and *hypernyms* of the input keywords. In practice, this naive approach fails due to high retrieval cost and substantially decreasing precision.

[1] Corresponding publication is: Saeedeh Shekarpour and Sören Auer, “Query Reformulation on RDF Knowledge Bases using Hidden Markov Models”, *Submitted to the Eighth International Conference on Web Search and Web Data Mining, WSDM 2015*, 2015

[2] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/` (March 1th, 2014)

Traditional information retrieval approaches cannot exploit the internal structure of data due to their use of bag-of-words semantics. For searching information on the Data Web we need similar informational retrieval concepts like words co-occurence which takes the internal structure of the data into account. An RDF knowledge base can be viewed as a directed, labeled graph $G_i = (V_i, E_i)$ where V_i is a set of nodes comprising all entities and literal property values, and E_i is a set of directed edges, i.e. the set of all properties.

In order to address the vocabulary mismatch problem, we employ a *Hidden Markov Model* (HMM) to obtain the optimal tuple of words. We test different bootstrapping methods for the HMM parameters using various distributions as well as an algorithm based on *Hyperlink-Induced Topic Search* (HITS). Our proposed functions for HMM parameters produce the best results for expansion.

Our main contributions are as follows:

- We define the concept of triple-based co-occurrence of words in RDF knowledge bases.
- We extend the Hidden Markov Model approach for choosing the most likely derived words.
- We create a benchmark test set for query expansion.
- We analyze the effect of this approach for both – queries requiring expansion and those which do not.

The paper is structured as follows: In the following section, we present the problem at hand in more detail and some of the notations and concepts used in this work. Section 5.3 presents the proposed reformulation method in detail. Our evaluation results are presented in Section 5.4 while related work is reviewed in Section 5.5. Finally, we conclude and give pointers to future work.

5.2 Problem and Preliminaries

In this work we focus on user-supplied queries in natural language, which we transform into an ordered sets of keywords through tokenizing and stop-word removal. Our input query thus is an n-tuple of keywords, i.e. $Q = (k_1, k_2, ..., k_n)$.

Definition 12 (Segment)**.** *For a given query $Q = (k_1, k_2, ..., k_n)$, the segment $S_{(i,j)}$ is the sequence of keywords from start position i to end position j, i.e., $S_{(i,j)} = (k_i, k_{i+1}, ..., k_j)$.*

The segment set is the list of all segments $S_{(i,j)}$ $(1 \leq i, j \leq n)$ derived for the input query Q with respect to the order of keywords. Since the number of keywords is in most queries less than six [3], retrieving the list of all segments is not computationally expensive.

The desired output of our approach is a ranked list of reformulated queries. A reformulated query formally is defined as follows:

Definition 13 (Reformulated query)**.** *For an n-tuple query $Q = (k_1, k_2, ..., k_n)$, a reformulated query is an m-tuple of keywords $R(Q) = (k'_1, k'_2, ..., k'_m)$ where each k'_i either matches a keyword k_j in Q or was linguistically derived from a segment $S_{(l,j)}$.*

For each segment s_i, we derive related words via linguistic features. Linguistic features extracted from *WordNet* [101] are:

- **Synonyms:** words having the same or a very similar meaning to the input segment s_i.

[3] http://www.keyworddiscovery.com/keyword-stats.html?date=2012-08-01

- **Hyponyms:** words representing a specialization of the input segment s_i.
- **Hypernyms:** words representing a generalization of the input segment s_i.

Definition 14 (Expansion set)**.** *For the given query Q, we define the expansion set as the union of all the following items:*

1. ***Original words:*** *segments extracted from the input query with respect to the order.*
2. ***Lemmatized words:*** *words determined as the lemma of the extracted segments.*
3. ***Linguistic words:*** *words derived via applying linguistic features over the extracted segments.*

5.3 Reformulating Query Using Hidden Markov Model

In this section, we describe how we use a Hidden Markov Model (HMM) for reformulating the input query. First, we introduce the notation of HMM parameters and then we detail how we bootstrap the parameters of our HMM for reformulating.

Hidden Markov Models: Formally, a Hidden Markov Model (HMM) is a quintuple $\lambda = (X, Y, A, B, \pi)$ where:

- X is a finite set of states. In our case, X equals to the expansion set associated with the input n-tuple of keywords.
- Y denotes the set of observations. Herein, Y equals to the set of all segments derived from the input n-tuple of keywords.
- $A : X \times X \rightarrow [0, 1]$ is the transition matrix. Each entry a_{ij} is the transition probability $Pr(S_j|S_i)$ from state i to state j.
- $B : X \times Y \rightarrow [0, 1]$ represents the emission matrix. Each entry $b_{ih} = Pr(h|S_i)$ is the probability of emitting the symbol h from state i.
- $\pi : X \rightarrow [0, 1]$ denotes the initial probability of states.

Commonly, supervised learning is employed for estimating the Hidden Markov Model parameters. We rely on *bootstrapping*, a technique used to estimate an unknown probability distribution function. Specifically, we bootstrap the parameters of our HMM by using statistical information as well as parameter sensitivity evaluation for the emission probability and more importantly the topology of the RDF graph for the transition probability. The results of the evaluation show that by using these bootstrapped parameters, we achieve a high Mean Reciprocal Rank (MRR) above 69% (as discussed in Section 5.4).

Constructing the State Space: A-priori, the state space is populated with as many states as the total number of words in the the *expansion set*. Note that the *expansion set* is formed based on the input n-tuple of keywords. In each state, the observable item is a segment that the associated word originated from. For instance, the word `spouse` is derived from the segment `wife`, so `wife` is emitted from the state `spouse`. Based on this construction of the state space, we are able to detect more likely derived words through the sequence of observable input keywords. The reformulated query is then determined through the most likely sequence of states.

Relations between States: There is a transition between two states, if the associated words are co-occurring. For RDF knowledge bases, we define co-occurrence of words in terms of *triple-based co-occurrence*.

Definition 15 (Triple-based Co-occurrence)**.** *In a given triple $t = (s, p, o)$, two words w_1 and w_2 are co-occurring, if they appear in the labels (`rdfs:label`) of at least two resources (i.e., (s, p), (s, o) or (o, p)). The following* SPARQL *query checks the co-occurrence of w_1 and w_2 in the* subject-predicate *position.*

```
ASK WHERE { {
  ?s  ?p          ?o  .
  ?s  rdfs:label  ?sl .
  ?p  rdfs:label  ?pl .
  FILTER(regex(?sl,"word1") AND
       regex(?pl,"word2"))
} UNION {
  ?s  ?p          ?o  .
  ?s  rdfs:label  ?sl .
  ?p  rdfs:label  ?pl .
  FILTER(regex(?sl,"word2") AND
         regex(?pl,"word1"))
}
FILTER(langMatches(lang(?sl),"en") AND
       langMatches(lang(?pl),"en")) }
```

5.3.1 Bootstrapping the Model Parameters

Our bootstrapping approach for the model parameters A and π is based on the HITS algorithm and relations between words in the knowledge base. The rationale is that the relatedness of two resources can be defined in terms of two parameters: the co-occurrence between the two words and the popularity of each of the words. The co-occurrence between two words is based on triple-based co-occurrence. The popularity of a word is simply the co-occurrence degree of the words available in the state space. We use the HITS algorithm for transforming these two values to hub and authority values (as detailed below). An analysis of the bootstrapping shows significant improvement of accuracy. In the following, we first introduce the *HITS* algorithm, since it is employed within the functions for computing the two HMM parameters A and π. Then, we discuss the distribution functions proposed for each parameter. Finally, we compare our bootstrapping method with other well-known distribution functions.

Hub and Authority of States. *Hyperlink-Induced Topic Search* (HITS) is a link analysis algorithm that was developed originally for ranking Web pages [63]. It assigns a hub and an authority value to each Web page. The *hub value* estimates the value of links to other pages and the *authority value* estimates the value of the content on a page. Hub and authority values are mutually interdependent and computed in a series of iterations. In each iteration the authority value is updated to the sum of the hub scores of each referring page; and the hub value is updated to the sum of the authority scores of each referring page. After each iteration, hub and authority values are normalized. This normalization process causes these values to converge eventually.

For each two states S_i and S_j in the state space, we add an edge if he two corresponding words co-occurs in one triple. The authority of a state can now be computed by: $auth(S_j) = \sum\limits_{S_i} hub(S_i)$. The hub value of a state is given by $hub(S_j) = \sum\limits_{S_i} auth(S_i)$. These definitions of hub and authority for states are the foundation for computing the transition and initial probabilities in the HMM.

Transition Probability. To compute the transition probability between two states, we take the connectivity of the whole space state into account. The transition probability of state S_j following state S_i is denoted as $a_{ij} = Pr(S_j|S_i)$. Note that the condition $\sum_{S_i} Pr(S_j|S_i) = 1$ holds. The transition probability from the state S_i to the state S_j is computed by:

$$a_{ij} = Pr(S_j|S_i) = \frac{auth(S_j)}{\sum_{\forall a_{ik}>0} auth(S_k)} \times hub(S_i)$$

Here, the probabilities from state S_i to the neighbouring states are uniformly distributed based on the authority values. Consequently, states with higher authority values are more probable to be met.

Initial Probability. The initial probability $\pi(S_i)$ is the probability that the model assigns to the initial state S_i at the beginning. The initial probabilities fulfill the condition $\sum_{\forall S_i} \pi(S_i) = 1$. We denote states for which the first keyword is observable by $InitialStates$. The initial states are defined as follows:

$$\pi(S_i) = \frac{auth(S_i) + hub(S_i)}{\sum_{\forall S_j \in InitialStates} (auth(S_j) + hub(S_j))}$$

In fact, $\pi(S_i)$ of an initial state is uniformly distributed on both hub and authority values.

Emission Probability. For computing the emission probability of the state S_i and the emitted segment h, we consider the origin type of the word associated to the state S_i. Words associated with the states originated from one of following:

1. *Original words:* segments extracted from the input query with respect to the original order.
2. *Lemmatized words:* words representing lemmas of the extracted segments.
3. *Linguistic words:* words obtained by through linguistic relations from the extracted segments.

We assign the probability θ if the word has as origin either *original words* or *lemmatized words*.

$$b_{ih} = Pr(h|S_i) = \theta$$

We assign the probability η if the word is a *linguistic word*.

$$b_{ih} = Pr(h|S_i) = \eta$$

Intuitively, θ should be larger than η. A statistical analysis with our query corpus confirms this assumption. Accordingly, around 82% of the words do not have a vocabulary mismatch problem. Hence, taking either the original words or lemmatised words into account suffices to a large extent. Only around 12% of the words have a vocabulary mismatch problem. However, we can not solely rely on this statistics. We consider the difference γ between θ and η and perform a parameter sensitivity evaluation on γ.

Viterbi Algorithm for the K-best Set of Hidden States. The optimal path through the HMM for a given sequence (i.e. input query keywords) generates a reformulated query. The *Viterbi algorithm* or *Viterbi path* [64] is a dynamic programming approach for finding the optimal path through a HMM for a given input sequence. It discovers the most likely sequence of underlying hidden states that might have generated a given sequence of observations. This discovered path has the maximum joint emission and transition probability of the involved states. The sub-paths of this most likely path also have the maximum probability for the respective sub-sequence of observations. The naive version of this algorithm just keeps track of the most likely path. We extended this algorithm using a tree data structure to store all possible paths generating the observed query keywords. Thus, our implementation can provide a ranked list of all paths generating the observation sequence with the corresponding probability.

Example 5.16 Let us consider the input query: `altitude of Everest`. The observation list is shown in the first column of Table 5.1. After constructing and pruning state space, it contains the words listed in the second column of Table 5.1 (only a subset of state space is presented). By running the *Viterbi algorithm*, we have a ranked list of the chosen words that the sequence `altitude of Everest` is observable through. In the following, we show the top-4 most likely paths along with their associated probability.

```
0.02451:   altitude,    Everest.
0.01681:   elevation,   Everest.
0.01145:   length,      Everest.
0.01145:   height,      Everest.
```

Observation	**States**	**Origin Type**
Everest	`Everest`	original keyword
altitude	`altitude`	original keyword
	`elevation`	synonym
	`height`	synonym
	`ceiling`	hyponym
	`level`	hyponym
	`distance`	hypernym
altitude Everest	`altitude Everest`	original keyword

Table 5.1: A subset of state space along with the origin type, list of all observations for the given query `altitude Everest`.

5.4 Evaluation

In general, when applying expansion methods, there is a high risk of yielding a large set of irrelevant words, which will have a negative impact on the runtime and the accuracy of the question answering system. For this reason, the goal of our evaluation is to determine: (1) How effective is our method with regard to a correct reformulation of queries which have vocabulary mismatch problem? (2) How robust is the method for queries which do not have vocabulary mismatch problem? To the best of our knowledge, no benchmark has been created so far for query expansion tasks over Linked Data. Thus, we created a benchmark dataset, which contains 20 keyword queries obtained from the *QALD-1* and *QALD-2* benchmarks[4]. The *QALD-1* and *QALD-2* benchmarks are essentially tailored for comparing question answering systems based on natural language queries. Out of the overall 20 queries 10 have a vocabulary mismatch problem and require to be reformulated. Table 5.2 lists the entire set of queries. In addition, the last column of this table shows the statistics over the number of derived words per query. As it can be seen, for a low number of input keywords, relatively high number of derived words can be taken into account.

Our experimental study is divided into two parts. First, we perform an evaluation of the bootstrapping parameters of the Hidden Markov Model. We dedicate 10 initial queries from the benchmark for this purpose. Second, we evaluate the overall accuracy using the remaining queries. In this step, we employ the optimal parameter setting learned during the first step.

Since the output of our approach is a ranked list, the metric employed for evaluation is *rank R* and *cumulative rank CR*. If the desired output is placed in the position *i*, the dedicated rank is *i*. Moreover, it

[4] `http://www.sc.cit-ec.uni-bielefeld.de/qald-`n for $n = 1, 2$.

ID	**Query**	**Mismatch word**	**Match word**	**# Derived words**
Q1	profession bandleader	profession	occupation	47
Q2	Barak Obama wife	wife	spouse	48
Q3	Is Natalie Portman an actress?	actress	actor	139
Q4	Lawrence of Arabia conflict	conflict	battle	114
Q5	children of Benjamin Franklin	children	child	41
Q6	capital of Canada	-	-	52
Q7	companies in Munich	-	-	11
Q8	governor of Texas	-	-	25
Q9	official languages of the Philippines	-	-	108
Q10	Who founded Intel?	-	-	6
Q11	movies with Tom Cruise	movie	film	77
Q12	husband of Amanda Palmer wife	husband	spouse	32
Q13	altitude of Everest	altitude	elevation	16
Q14	companies in California	company	organisation	117
Q15	writer of Wikipedia	writer	author	966
Q16	soccer clubs in Spain	-	-	1 9
Q17	employees of Google	-	-	10
Q18	successor of John F. Kennedy	-	-	77
Q19	nicknames of San Francisco	-	-	15
Q20	Statue of Liberty built	-	-	43

Table 5.2: List of the queries in our benchmark along with the number of the derived words.

is probable that several items have the same rank, thus, we define cumulative rank in the position p as: $CR = \sum_{i=1}^{p} n_i$. The goal is to reach a minimum for R and CR.

As mentioned in the previous section, γ is the difference between the emission probabilities θ and η. $\gamma = 0$ means that we do not differentiate between original words and derived words. We measured the accuracy of the approach by assigning different values to γ. Figure 5.2 and Figure 5.3 shows the results of evaluation. Figure 5.2 shows rank and cumulative rank for queries required to be reformulated and Figure 5.3 shows the results for queries which do not. Obviously, assigning a positive value to γ prioritizes original words. The optimum value is $\gamma = 0.3$.

We bootstrapped transition probability on uniform distribution once with applying HITS algorithm and once without. Figure 5.1 compares the results of this bootstrapping. On average, the cumulative rank (CR) without applying HITS is higher. It can be observed, that by applying HITS, a model can produce more distinguishable probability values. Applying HITS is slightly more effective.

Figure 5.4 represents rank (R) and cumulative rank (CR) for test queries from the benchmark. The first five queries require to be reformulated and the rest do not. Although the size of state space is relatively high, the desired reformulated query appears in top-4 items. An important observation is that this approach is quite robust in case of queries that do not have vocabulary mismatch problem. In fact, including a high number of derived words does not affect the priority of original words.

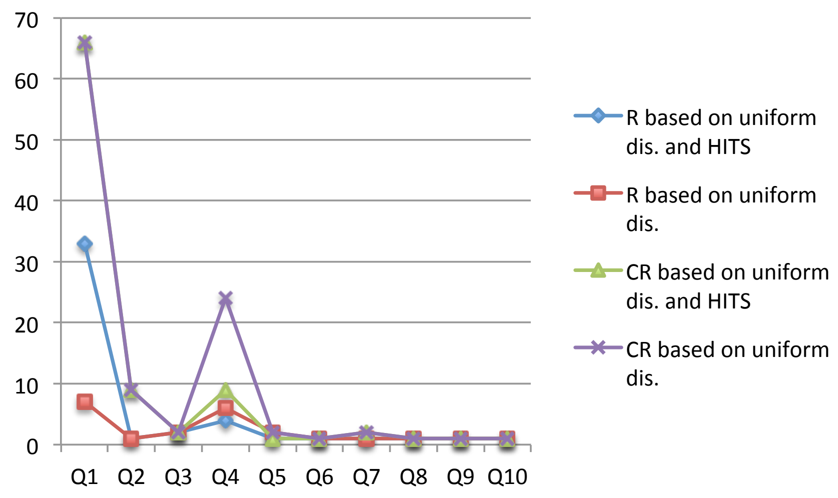


Figure 5.1: Rank (R) and Cumulative Rank (CR) with uniform distribution with and without applying HITS algorithm.

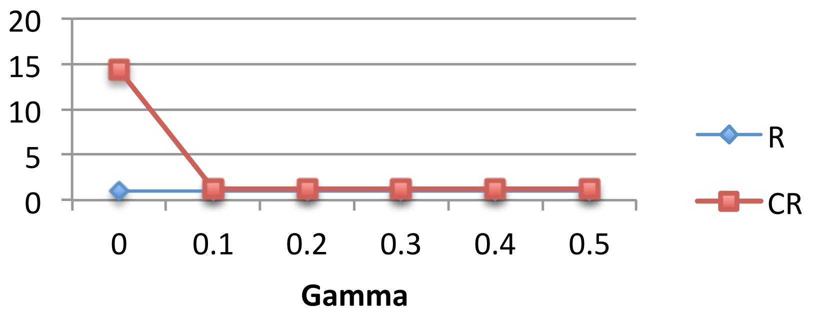


Figure 5.2: Rank (R) and Cumulative Rank (CR) for different values of γ for queries that require expansion.

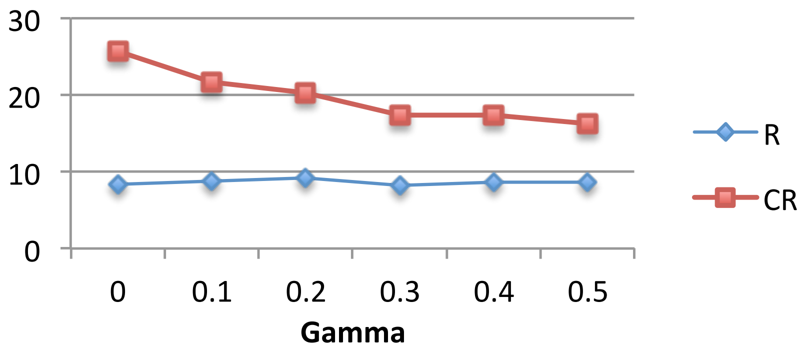


Figure 5.3: Rank (R) and Cumulative Rank (CR) for different values of γ for queries that do not require expansion.

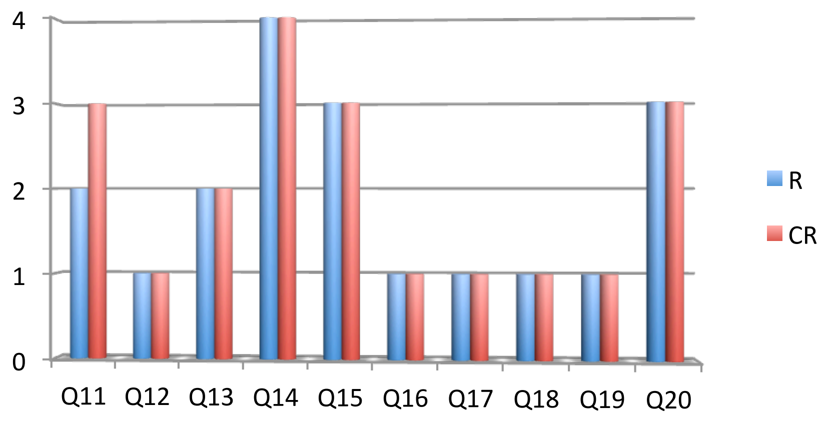


Figure 5.4: Rank (R) and Cumulative rank (CR) for the test queries.

5.5 Related Work

Automatic Query Expansion (aqe) has been focus of researchers for a long time. It aims at improving retrieval efficiency. Here we divide available works into two parts. The first part discusses a load of approaches employed for query expansion on Web of Documents. The second part presents state of art of query expansion on Semantic Web.

5.5.1 Query Expansion on Web of Documents

Various approaches differ in the choice of data sources and features. Data source is a collections of words or documents. The choice of the data source influences the size of the vocabulary (expansion terms) as well as the available features. Furthermore, a data source is often restricted to a certain domain and thus constitutes a certain context for the search terms. Common choices for data sources are text corpora, WordNet synsets, hyponyms and hypernyms, anchor texts, search engine query logs or top ranked documents. A popular way of choosing data source is Pseudo Relevance Feedback (PRF) based query expansion. In this method, top-n retrieved documents are assumed to be relevant and employed as data source. [102] as one of the early works expands queries based on word similarity using cosine coefficient. [103, 104] give the theoretical basis for detecting similarity based on co-occurence.

Feature selection [91] consists of two parts: (1) feature weighting assigns a scoring to each feature and (2) the feature selection criterion determines which of the features to retain based on the weights. Some common feature selection methods are *mutual information*, *information gain*, *divergence from randomness* and *relevance models*. A framework for feature weighting and selection methods is presented in [91]. The authors compare different feature ranking schemes and show that **SVM** achieves the highest F_1-score of the examined methods.

[90] presents a comprehensive survey of aqe in information retrieval and detail a large amount of candidate feature extraction methods.

5.5.2 Query Expansion on Semantic Web

Since Semantic Web is publishing structured data, expansion methods can be enhanced by taking the structure of data into account. A very important choice is defining and employing new features.

For instance, our previous work [105] introduces a number of new query expansion features based on semantic and linguistic inferencing over Linked Open Data. It evaluates the effectiveness of each feature individually as well as their combinations employing several machine learning approaches. An approach similar to ours is [100], however it relies on supervised learning and uses only semantic expansion features instead of a combination of both semantic and linguistic ones. A prerequisite of feature-based Query Expansion is, that the data about the features used, i.e. the pairs contained in the relations, is available. Relation equivalency (*owl:equivalentProperty*) links in particular are often not, or not completely, defined for a knowledge base. There is however an approach for mining equivalent relations from Linked Data, that relies on three measures of equivalency: *triple overlap*, *subject agreement* and *cardinality ratio*. [99]

However, While **aqe** is prevalent in traditional web search engines, the current semantic search engines either do not address the vocabulary problem or tackle it in a different way. SemSeK [97], SINA [61], QAKiS [98] are semantic search engines which still do not use any kind of query expansion and rely instead only on exact string matching between the query keyword and the label of the associated resource. *Alexandria* [93] uses *Freebase* to include synonyms and different surface forms. *MHE*[5] combines query expansion and entity recognition by using textual references to a concept and extracting Wikipedia anchor texts of links. This approach takes advantage of a large amount of hand-made mappings that emerge as a byproduct. However, this approach is only viable for Wikipedia-DBpedia or other text corpora whose links are mapped to resources. *Eager* [95] expands a set of resources with resources of the same type using DBpedia and Wikipedia categories (instead of linguistic and semantic features in our case). Eager extracts implicit category and synonym information from abstracts and redirect information, determines additional categories from the DBpedia category hierarchy and then extracts additional resources which have the same categories in common. *PowerAqua* [41] is an ontology-based system that answers natural language queries and uses WordNet synonyms and hypernyms as well as resources related with the `owl:sameAs` property.

The presented approach here benefits structure of data for expanding query. With respect to the available methods fiting to Semantic Web, it is a pioneer.

5.6 Conclusion

In this paper, we presented a method for automatic query expansion. It employs a Hidden Markov Model for generating a suitable reformulated query. The relations between states in this model are formed using triple-based co-occurrence of words in the RDF knowledge base. With this approach we are able to exclude less likely derived words. The result of the evaluation confirms the feasibility and high accuracy of this model. We plan to extend this work in different directions. Up to now, all computations are carried out on the fly. We plan to create an index especially for co-occurrence of words in order to reduce the computational load during runtime. Another possible extension is applying a supervised learning approach for learning Hidden Markov Model parameters. As a next step, we plan to extend the size of our benchmark and perform further evaluations. We are also interested in applying more features (e.g. obtained from Wikipedia) for constructing the expansion set.

[5] `http://ups.savba.sk/~marek`

Appendix

APPENDIX A

Keyword-Driven Resource Disambiguation over RDF Knowledge Bases[1]

In this appendix, we attach our publication *Keyword-Driven Resource Disambiguation over RDF Knowledge Bases* [9] which provides more experimental study on query segmentation and resource disambiguation.

[1] Published as: Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "Keyword-Driven Resource Disambiguation over RDF Knowledge Bases", *Semantic Technology, Second Joint International Conference, JIST 2012, Nara, Japan, December 2-4, 2012. Proceedings*, Springer, 2012 pp. 159–174

Keyword-Driven Resource Disambiguation over RDF Knowledge Bases

Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer

Department of Computer Science, University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany
`{lastname}@informatik.uni-leipzig.de`

Abstract. Keyword search is the most popular way to access information. In this paper we introduce a novel approach for determining the correct resources for user-supplied queries based on a hidden Markov model. In our approach the user-supplied query is modeled as the observed data and the background knowledge is used for parameter estimation. We leverage the semantic relationships between resources for computing the parameter estimations. In this approach, query segmentation and resource disambiguation are mutually tightly interwoven. First, an initial set of potential segments is obtained leveraging the underlying knowledge base; then, the final correct set of segments is determined after the most likely resource mapping was computed. While linguistic analysis (e.g. named entity, multi-word unit recognition and POS-tagging) fail in the case of keyword-based queries, we will show that our statistical approach is robust with regard to query expression variance. Our experimental results reveal very promising results.

1 Introduction

The Data Web currently amounts to more than 31 billion triples[1] and contains a wealth of information on a large number of different domains. Yet, accessing this wealth of structured data *remains* a key challenge for lay users. The same problem emerged in the last decade when users faced the huge amount of information available of the Web. Keyword search has been employed by popular Web search engines to provide access to this information in a user-friendly, low-barrier manner. However, keyword search in structured data raises two main difficulties: First, the *right segments of data items* that occur in the keyword queries have to be identified. For example, the query *'Who produced films starring Natalie Portman'* can be segmented to (*'produce'*, *'film'*, *'star'*, *'Natalie Portman'*) or (*'produce'*, *'film star'*, *'Natalie'*, *'Portman'*). Note that the first segmentation is more likely to lead to a query that contain the results intended for by the user. Second, these segments have to be disambiguated and mapped to the right resources. Note that the resource ambiguity problem is of increasing importance

[1] See `http://www4.wiwiss.fu-berlin.de/lodcloud/state/` (May 23th, 2012).

as the size of knowledge bases on the Linked Data Web grows steadily. Considering the previous example[2], the segment *'film'* is ambiguous because it may refer to the class `dbo:Film` (the class of all movies in DBpedia) or to the properties `dbo:film` or `dbp:film` (which relates festivals and the films shown during these festivals). In this paper, we present an automatic query segmentation and resource disambiguation approach leveraging background knowledge. Note that we do not rely on training data for the parameter estimation. Instead, we leverage the semantic relationships between data items for this purpose. While linguistic methods like named entity, multi-word unit recognition and POS-tagging fail in the case of an incomplete sentences (e.g. for keyword-based queries), we will show that our statistical approach is robust with regard to query expression variance. This article is organized as follows: We review related work in Section 2. In Section 3 we present formal definitions laying the foundation for our work. In the section 4 our approach is discussed in detail. For a comparison with natural language processing (NLP) approaches section 5 introduces an NLP approach for segmenting query. Section 6 presents experimental results. In the last section, we close with a discussion and an outlook on potential future work.

2 Related Work

Our approach is mainly related to two areas of research: text and query segmentation and entity disambiguation. Text segmentation has been studied extensively in the natural language processing (NLP) literature, and includes tasks such as noun phrase chunking where the task is to recognize the chunks that consist of noun phrases (see e.g., [16]). Yet, such approaches cannot be applied to query segmentation since queries are short and composed of keywords. Consequently, NLP techniques for chunking such as part-of-speech tagging [4] or name entity recognition [7,5] cannot achieve high performance when applied to query segmentation. Segmentation methods for document-based Information Retrieval can be categorized into statistical and non-statistical algorithms. As an example of none statistical methods, [15] addresses the segmentation problem as well as spelling correction. Each keyword in a given query is first expanded to a set of similar tokens in the database. Then, a dynamic programming algorithm is used to search for the segmentation based on a scoring function. The statistical methods fall into two groups, namely supervised and unsupervised methods. For example, the work presented in [19] proposes an unsupervised approach to query segmentation in Web search. Yet this technique can not be easily applied to structured data. Supervised statistical approaches are used more commonly. For example, [25] presents a principled statistical model based on Conditional Random Fields (CRF) whose parameters are learned from query logs. For detecting named entities, [9] uses query log data and Latent Dirichlet Allocation. In addition to query logs, various external resources such as Webpages, search

[2] The underlying knowledge base and schema used throughout the paper for examples and evaluation is DBpedia 3.7 dataset and ontology.

```
Data: $q$: n-tuple of keywords, knowledge base
Result: SegmentSet: Set of segments
SegmentSet=new list of segments;
start=1;
while $start <= n$ do
    $i = start$;
    while $S_{(start,i)}$ is valid do
        $SegmentSet.add(S_{(start,i)})$;
        i++;
    end
    start++;
end
```

Algorithm 1. Naive algorithm for determining all valid segments taking the order of keywords into account

result snippets and Wikipedia titles have been used [17,20,3]. Current segmentation algorithms are not applicable to our segmentation problem for several reasons. First, because they mostly are not intended for search on structured data, it is not guaranteed that the segments they retrieve are actually part of the underlying knowledge base. Another problem with these segmentation algorithms is that they ignore the semantic relationships between segments of a segmentation. Thus, they are likely to return sub-optimal segmentations.

An important challenge in Web search as well as in Linked Data Search is entity disambiguation. Keyword queries are usually short and inherently lead to significant keyword ambiguity as one query word may represent different information needs for different users [21]. There are different ways for tackling this challenge; firstly, query clustering [6,23,2] applies unsupervised machine learning to cluster similar queries. The basic insight here is that it has been observed that users with similar information needs click on a similar set of pages, even though the queries they pose may vary. Other approaches apply query disambiguation, which tries to find the most appropriate sense of each keyword. To achieve this goal, one way is involving the user in selecting the correct sense [11,10,24]. Another technique for disambiguation is personalized search by using a history of the user activities to tailor the best choice for disambiguation [1,18,26]. Still, the most common approach is using context for disambiguation [14,8,13]. Albeit context has been defined vaguely (with various definitions), herein we define context as information surrounding the given query which can be employed for augmenting search results. In this work, resource disambiguation is based on a different type of context: we employ the structure of the knowledge at hand as well as semantic relations between the candidate resources mapped to the possible segmentations of the input query.

3 Formal Specification

RDF data is modeled as a directed, labeled graph $G = (V, E)$ where V is a set of nodes i.e. the union of entities and property values, and E is a set of directed edges i.e. the union of object properties and data value properties.

The user-supplied query can be either a complete or incomplete sentence. However, after removing the stop words, typically set of keywords remains. The order in which keywords appear in the original query is partially significant. Our approach can map adjacent keywords to a joint resource. However, once a mapping from keywords to resources is established the order of the resources does not affect the SPARQL query construction anymore. This is a reasonable assumption, since users will write strongly related keywords together, while the order of only loosely related keywords or keyword segments may vary. The input query is formally defined as an n-tuple of keyword, i.e. $Q = (k_1, k_2, ..., k_n)$. We aim to transform the input keywords into a suitable set of entity identifiers, i.e. resources $R = \{r_1, r_2...r_m\}$. In order to accomplish this task the input keywords have to be grouped together as segments and for each segment a suitable resource should be determined.

Definition 1 (Segment and Segmentation). *For a given query Q, a segment $S_{(i,j)}$ is a sequence of keywords from start position i to end position j which is denoted as $S_{(i,j)} = (k_i, k_{i+1}, ..., k_j)$. A query segmentation is an m-tuple of segments $SG_q = (S_{(0,i)}, S_{(i+1,j)}, ..., S_{(m,n)})$ where the segments do not overlap with each other and arranged in a continuous order, i.e. for two continuous segments $S_x, S_{x+1} : Start(S_{x+1}) = End(S_x) + 1$. The concatenation of segments belonging to a segmentation forms the corresponding input query Q.*

Definition 2 (Resource Disambiguation). *Let the segmentation $SG' = (S^1_{(0,i)}, S^2_{(i+1,j)}, ..., S^x_{(m,n)})$ be a suitable segmentation for the given query Q. Each segment is mapped to multiple candidate resources from the underlying knowledge base, i.e. $S^i \rightarrow R^i = \{r_1, r_2...r_h\}$. The aim of disambiguation is to choose an x-tuple of resources from the Cartesian product of sets of candidate resources $(r_1, r_2, ..., r_x) \in \{R^1 \times R^2 \times ...R^x\}$ for which each r_i has two important properties. First, it is among the highest ranked candidates for the corresponding segment with respect to the similarity as well as popularity and second it shares a semantic relationship with other resources in the x-tuple.*

When considering the order of keywords, the number of segmentations for a query Q consisting of n keywords is $2^{(n-1)}$. However, not all these segmentations contain valid segments. A valid segment is a segment for which at least one matching resource can be found in the underlying knowledge base. Thus, the number of segmentations is reduced by excluding those containing invalid segments. Algorithm 1 is an extension of the greedy approach presented in [25]. This naive approach finds all valid segments when considering the order of keywords. It starts with the first keyword in the given query as first segment, then it includes the next keyword into the current segment as a new segment and checks whether adding the new keyword would make the new segment no longer valid. We repeat this process until we reach the end of the query. As a running example, lets assume the input query is *'Give me all video games published by Mean Hamster Software'*. Table 1 shows the set of valid segments based on naive algorithm along with some samples of the candidate resources. Note that the suitable segmentation is *('video games', 'published', 'Mean Hamster Software')*.

Table 1. Generated segments and samples of candidate resources for a given query

Segments	Samples of Candidate Resources
video	1. dbp:video
video game	1. dbo:VideoGame
game	1. dbo:Game 2. dbo:games 3. dbp:game 4. dbr:Game 5. dbr:Game_On
publish	1. dbo:publisher 2. dbp:publish 3. dbr:Publishing
mean	1. dbo:meaning 2. dbp:meaning 3. dbr:Mean 4. dbo:dean
mean hamster	1. dbr:Mean_Hamster_Software
mean hamster software	1. dbr:Mean_Hamster_Software
hamster	1. dbr:Hamster
software	1. dbo:Software 2. dbp:software

Resource Disambiguation Using a Ranked List of Cartesian Product Tuples: A naive approach for finding the correct $x - tuple$ of resources is using a ranked list of tuples from the Cartesian product of sets of candidate resources $\{R^1 \times R^2 \times ...R^n\}$. The n-tuples from the Cartesian product are simply sorted based on the aggregated relevance score (e.g. similarity and popularity) of all contained resources.

4 Query Segmentation and Resource Disambiguation Using Hidden Markov Models

In this section we describe how hidden Markov models are used for query segmentation and resource disambiguation. First we introduce the concept of hidden Markov models and then we detail how we define the parameters of a hidden Markov model for solving the query segmentation and entity disambiguation problem.

4.1 Hidden Markov Models

The Markov model is a stochastic model containing a set of states. The process of moving from one state to another state generates a sequence of states. The probability of entering each state only depends on the previous state. This memoryless property of the model is called *Markov property.* Many real-world processes can be modeled by Markov models. A hidden Markov model is an extension of the Markov model, which allows the observation symbols to be emitted from each state with a finite probability. The main difference is that by looking at the observation sequence we cannot say exactly what state sequence has produced these observations; thus, the state sequence is *hidden.* However, the probability of producing the sequence by the model can be calculated as well as which state sequence was most likely to have produced the observations.

A hidden Markov model (HMM) is a quintuple $\lambda = (X, Y, A, B, \pi)$ where:

- X is a finite set of states, Y denotes the set of observed symbols;
- $A : X \times X \to \mathbb{R}$ is the transition matrix that each entry $a_{ij} = Pr(S_j|S_i)$ shows the transition probability from state i to state j;

– $B : X \times Y \rightarrow \mathbb{R}$ represents the emission matrix, in which each entry $b_{ih} = Pr(h|S_i)$ is associated with the probability of emitting the symbol h from state i;
– π denoting the initial probability of states $\pi_i = Pr(S_i)$.

4.2 State Space and Observation Space

State Space. A state represents a knowledge base entity. Each entity has an associated *rdfs:label* which we use to label the states. The actual number of states X is potentially high because it contains theoretically all RDF resources, i.e. $X = V \cup E$. However, in practice we limit the state space by excluding irrelevant states. A relevant state is defined as a state for which a valid segment can be observed. In other words, a valid segment is observed in an state if the probability of emitting that segment is higher than a certain threshold θ. The probability of emitting a segment from a state is computed based on a similarity scoring which we describe in the section 4.3. Therefore, the state space of the model is pruned and contains just a subset of resources of the knowledge base, i.e. $X \subset V \cup E$. In addition to these candidate states, we add an **unknown entity state** to the set of states. The *unknown entity* (UE) state comprises all entities, which are not available (anymore) in the pruned state space. The *observation space* is the set of all valid segments found in the input user query (using e.g. the Algorithm 1). It is formally is defined as $O = \{o | \text{o is a valid segment}\}$.

4.3 Emission Probability

Both the labels of states and the segments contain sets of words. For computing the emission probability of the state i and the emitted segment h, we compare the similarity of the label of state i with the segment h in two levels, namely string-similarity level and set-similarity level: (1) The *set-similarity level* measures the difference between the label and the segment in terms of the number of words using the *Jaccard similarity.* (2) The *string-similarity level* measures the string similarity of each word in the segment with the most similar word in the label using the *Levenshtein distance.* Our similarity scoring method is now a combination of these two metrics. Consider the segment $h = (k_i, k_{i+1}, ..., k_j)$ and the words from the label l divided into a set of keywords M and stopwords N, i.e. $l = M \cup N$. The total similarity score between keywords of a segment and a label is then computed as follows:

$$b_{ih} = Pr(h|S_i) = \frac{\sum_{t=i}^{j} argmax_{\forall m_i \in M}(\sigma(m_i, k_t))}{|M \cup h| + 0.1 * |N|}$$

This formula is essentially an extension of the *Jaccard similarity coefficient.* The difference is that in the numerator, instead of using the cardinality of intersections the sum of the string-similarity score of the intersections is computed. As in the Jaccard similarity, the denominator comprises the cardinality of the union of two sets (keywords and stopwords). The difference is that the number of stopwords have been down-weighted by the factor 0.1 to reduce their influence (since they do not convey much meaningful information).

4.4 Hub and Authority of States

Hyperlink-Induced Topic Search (HITS) is a link analysis algorithm for ranking Web pages [12]. Authority and hub values are defined in terms of one another and computed in a series of iterations. In each iteration, hub and authority values are normalized. This normalization process causes these values to converge eventually. Since RDF data is graph-structured data and entities are linked together, we employed a weighted version of the HITS algorithm in order to assign different popularity values to the states in the state space. For each state we assign a hub value and an authority value. A good hub state is one that points to many good authority states and a good authority state is one that is pointed to from many good hub states. Before discussing the HITS computations, we define the edges between the states in the HMM. For each two states i and j in the state space, we add an edge if there is a path in the knowledge base between the two corresponding resources of maximum length k. Note, that we also take property resources into account when computing the path length.The path length between resources in the knowledge base is assigned as weight to the edge between corresponding states. We use a weighted version of the HITS algorithm to take the distance between states into account. The authority of a state is computed as: For all $S_i \in S$ which point to $S_j : auth_{S_j} = \sum_{\forall i} w_{i,j} * hub_{S_i}$ And the hub value of a state is computed as: For all $S_i \in S$ which are pointed to by $S_j : hub_{S_j} = \sum_{\forall i} w_{i,j} * auth_{S_i}$ The weight $w_{i,j}$ is defined as $w_{i,j} = k - pathLength(i, j)$, where $pathLength(i, j)$ is the length of the path between i and j. These definitions of hub and authority for states are the foundation for computing the transition probability in the underlying hidden Markov model.

4.5 Transition Probability

As mentioned in the previous section, each edge between two states shows the shortest path between them with the length less or equal to k-hop. The edges are weighted by the length of the path. Transition probability shows the probability of going from state i to state j. For computing the transition probability, we take into account the connectivity of the whole of space state as well as the weight of the edge between two states. The transition probability values decrease with the distance of the states, e.g. transitions between entities in the same triple have higher probability than transitions between entities in triples connected through extra intermediate entities. In addition to the edges recognized as the shortest path between entities, there is an edge between each state and the *Unknown Entities* state. The transition probability of state j following state i denoted as $a_{ij} = Pr(S_j|S_i)$. For each state i the condition $\sum\limits_{\forall S_j} Pr(S_j|S_i) = 1$ should be held. The transition probability from the state i to *Unknown Entity* (UE)

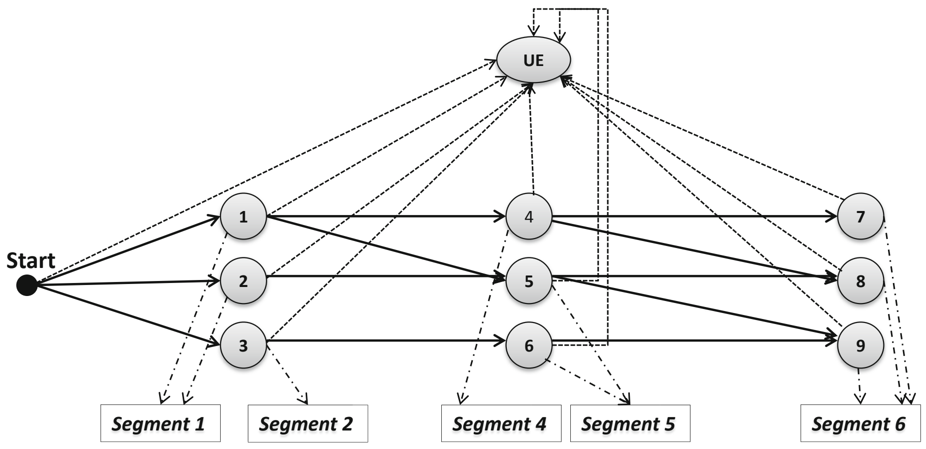


Fig. 1. Trellis representation of Hidden Markov Model

state is defined as: $a_{iUE} = Pr(UE|S_i) = 1 - hub_{S_i}$ And means a good hub has less probability to go to UE state. Thereafter, the transition probability from the state i to state j is computed as: $a_{ij} = Pr(S_j|S_i) = \frac{auth_{S_j}}{\sum\limits_{\forall a_{ik}>0} auth_{S_k}} * hub_{S_i}$. Here, the edges with the low distance value and higher authority values are more probable to be met.

4.6 Initial Probability

The initial probability π_{S_i} is the probability that the model assigns to the initial state i in the beginning. The initial probabilities fulfill the condition $\sum\limits_{\forall S_i} \pi_{S_i} = 1$. We denote states for which the first keyword is observable by $InitialStates$. The initial states are defined as follows:

$$\pi_{S_i} = \frac{auth_{S_i} + hub_{S_i}}{\sum\limits_{\forall S_j \in InitialStates} (auth_{S_j} + hub_{S_j})}$$

In fact, π_{S_i} of an initial state depends on both hub and authority values. Figure 1 illustrates an instantiated hidden markov model. The set of hidden states are represented by circles. The state UE refers to the absent resources in the model and other hidden states are relevant resources. Each segment box represents a possible observation. The arrows show a transition from one state to another state and the dashed arrows shows an emitted observation associated with a specific state.

4.7 Viterbi Algorithm for the K-Best Set of Hidden States

The optimal path through the Markov model for a given sequence (i.e. input query keywords) reveals disambiguated resources forming a correct segmentation. The *Viterbi algorithm* or *Viterbi path* is a dynamic programming approach for finding the optimal path through the markov model for a given sequence. It discovers the most likely sequence of underlying hidden states that might have generated a given sequence of observations. This discovered path has the maximum joint emission and transition probability of involved states. The sub paths of this most likely path also have the maximum probability for the respective sub sequence of observations. The naive version of this algorithm just keeps track of the most likely path. We extended this algorithm using a tree data structure to store all possible paths generating the observed query keywords. Therefore, in our implementation we provide a ranked list of all paths generating the observation sequence with the corresponding probability. After running the Viterbi algorithm for our running example, the disambiguated resources are: {*dbo:VideoGame, dbo:publisher, dbr:Mean-Hamster-Software*} and consequently the reduced set of valid segments is: {*VideoGam, publisher, Mean-Hamster-Software*} .

5 Query Segmentation Using Natural Language Processing

Natural language processing (NLP) techniques are commonly used for text segmentation. Here, we use a combination of named entity and multi-word unit recognition services as well as POS-tagging for segmenting the input-query. In the following, we discuss this approach in more detail.

Detection of Segments: Formally, the detection of segments aims to transform the set of keywords $K = \{k_1, .., k_n\}$ into a set of segments $\mathcal{T} = \{t_1, ..., t_m\}$ where each k_i is a substring of exactly one $t_j \in \mathcal{T}$. Several approaches have already been developed for this purpose, each with its own drawbacks: Semantic lookup services (e.g., *OpenCalais*[3] and *Yahoo! SeoBook*[4] as used in the current implementation) allow to extract *named entities* (NEs) and *multi-word units* (MWUs) from query strings. While these approaches work well for long queries such as *"Films directed by Garry Marshall starring Julia Roberts"*, they fail to discover noun phrases such as "highest place" in the query *"Highest place of Karakoram"*. We remedy this drawback by combining lookup services and a simple noun phrase detector based on POS tags. This detector first applies a POS tagger to the query. Then, it returns all sequences of keywords whose POS tags abide by the following right-linear grammar:

1. $S \rightarrow adj\ A$ 2. $S \rightarrow nn\ B$ 3. $A \rightarrow B$
4. $B \rightarrow nn$ 5. $B \rightarrow nn\ B$

where S is the start symbol, A and B are non-terminal symbols and nn (noun) as well as adj (adj) are terminal symbols. The compilation of segments is carried as follows: We send the input K to the NE and MWU detection services as well as to the noun phrase detector. Let $\mathcal{N}$ be the set of NEs, $\mathcal{M}$ the set of MWUs

[3] http://viewer.opencalais.com/
[4] http://tools.seobook.com/yahoo-keywords/

and $\mathcal{P}$ the set of noun phrases returned by the system. These three sets are merged to a set of labels $\mathcal{L} = (\mathcal{N} \oplus \mathcal{M}) \oplus \mathcal{P}$, where $\oplus$ is defined as follows:

$$A \oplus B = A \cup B \backslash \{b \in B | \exists a \in A \ overlap(a, b)\} \quad (1)$$

where $overlap(a, b)$ is true if the strings a and b overlap. The operation $\oplus$ adds the longest elements of B to A that do not overlap with A. Note that this operation is not symmetrical and prefers elements of the set A over those of the set B. For example, "`river which Brooklyn Bridge crosses`" leads to $\mathcal{N}$ = {"`Brooklyn Bridge`"}, M = {"`Brooklyn`" , "`Brooklyn Bridge`"} and $\mathcal{P}$ = {"`Brooklyn Bridge`"}. Thus, $\mathcal{L} = (\mathcal{N} \oplus \mathcal{M}) \oplus \mathcal{P}$ = {"`Brooklyn Bridge`"}. The final set of segments $\mathcal{T}$ is computed by retrieving the set of all single keywords that were not covered by the approaches above and that do not occur in a list of stopwords. Thus, for the query above, $\mathcal{T}$ = {"Brooklyn Bridge", "river", "cross"}.

6 Evaluation

The goal of our experiments was to measure the accuracy of resource disambiguation approaches for generating adequate SPARQL queries. Thus, the main question behind our evaluation was as follows: Given a keyword-based query(KQ) or a natural-language query (NL) and the equivalent SPARQL query, how well do the resources computed by our approaches resemble the gold standard. It is important to point out that a single erroneous segment or resource can lead to the generation of a wrong SPARQL query. Thus, our criterion for measuring the correctness of segmentations and disambiguations was that *all of the recognized segments* as well as *all of the detected resources* had to match the gold standard.

Experimental Setup. So far, no benchmark for query segmentation and resource disambiguation has been proposed in literature. Thus, we created such a benchmark from the DBpedia fragment of the question answering benchmark *QALD-2*[5]. The QALD-2 benchmark data consists of 100 training and 100 test questions in natural-language that are transformed into SPARQL queries. In addition, it contains a manually created keyword-based representation of each of the natural-language questions. The benchmark assumed the generic query generation steps for question answering: First, the correct segments have to be computed and mapped to the correct resources. Then a correct SPARQL query has to be inferred by joining the different resources with supplementary resources or literals. As we are solely concerned with the first step in this paper, we selected 50 queries from the QALD-2 benchmark (25 from the test and 25 from the training data sets) that were such that each of the known segments in the benchmark could be mapped to exactly one resource in the SPARQL query and vice-versa. Therewith, we could derive the correct segment to resource mapping directly from the benchmark[6]. Queries that we discarded include *"Give me all soccer*

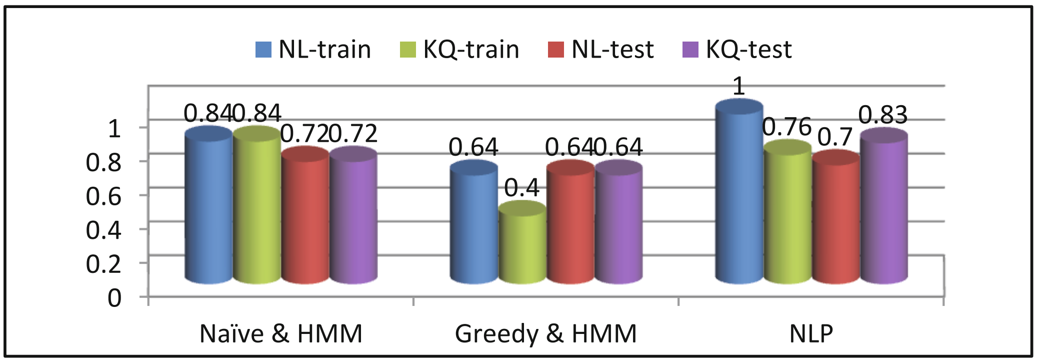
[5] `http://www.sc.cit-ec.uni-bielefeld.de/qald-2`

[6] The queries and result of the evaluation and source code is available for download at `http://aksw.org/Projects/lodquery`

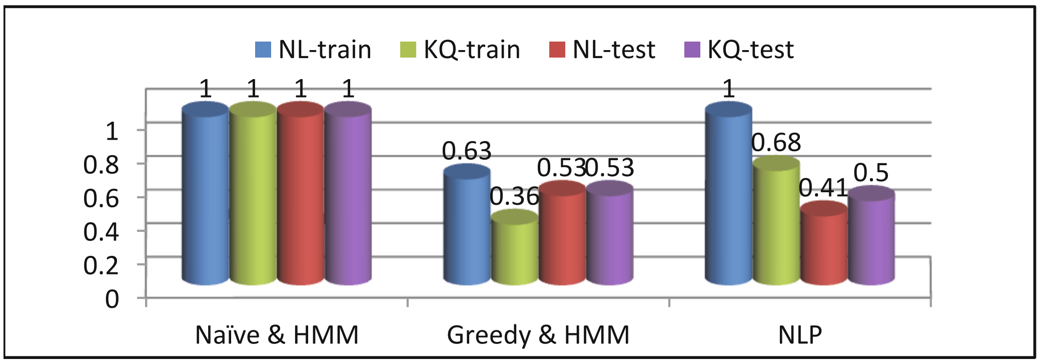
clubs in Spain", which corresponds to a SPARQL query containing the resources {`dbo:ground, dbo:SoccerClub, dbr:Spain` }. The reason for discarding this particular query was that the resource `dbo:ground` did not have any match in the list of keywords. Note that we also discarded queries requiring schema information beyond DBpedia schema. Furthermore, 6 queries out of the 25 queries from the training data set [7] and 10 queries out of 25 queries from the test data set [8] required a query expansion to map the keywords to resources. For instance, the keyword *"wife"* should be matched with *"spouse"* or *"daughter"* to *"child"*. Given that the approaches at hand generate and score several possible segmentations (resp. resource disambiguation), we opted for measuring the *mean reciprocal rank* MRR [22] for both the query segmentation and the resource disambiguation tasks. For each query $q_i \in Q$ in the benchmark, we compare the rank r_i assigned by different algorithms to the correct segmentation and to the resource disambiguation: $MRR(\mathcal{A}) = \frac{1}{|Q|}\sum\limits_{q_i}\frac{1}{r_i}$. Note that if the correct segmentation (resp. resource disambiguation) was not found, the reciprocal rank is assigned the value 0.

Results. We evaluated our hidden Markov model for resource disambiguation by combining it with the naive (Naive & HMM) and the greedy segmentation (Greedy & HMM) approaches for segmentation. We use the natural language processing (NLP) approach as a baseline in the segmentation stage. For the resource disambiguation stage, we combine ranked Cartesian product (RCP) with the natural language processing (NLP & RCP) and manually injected the correct segmentation (RCP) as the baseline. Note that we refrained from using any query expansion method. The segmentation results are shown in Figure 2. The MRR are computed once with the queries that required expansion and once without. Figure 2(a), including queries requiring expansion, are slightly in favor of NLP, which achieves on overage a 4.25% higher MRR than Naive+HMM and a 24.25% higher MRR than Greedy+HMM. In particular, NLP achieves optimal scores when presented with the natural-language representation of the queries from the "train" data set. Naive+HMM clearly outperforms Greedy+HMM in all settings. The main reason for NLP outperforming Naive+HMM with respect to the segmentation lies in the fact that Naive+HMM and Greedy+HMM are dependent on matching segments from the query to resources in the knowledge base (i.e. segmentation and resource disambiguation are interwoven). Thus, when no resource is found for a segment (esp. for queries requiring expansion) the HMM prefers an erroneous segmentation, while NLP works independent from the disambiguation phase. However, as it can be observed NLP depends on the query expression. Figure 2(b) more clearly highlights the accuracy of different approaches. Here, the MRR without queries requiring expansion is shown. Naive+HMM perfectly segments both natural language and keyword-based queries. The superiority of intertwining segmentation and disambiguation in Naive+HMM is clearly shown by our disambiguation results in the second stage in Figure 3. In this stage,

[7] Query IDs: 3, 6, 14, 43, 50, 93.
[8] Query IDs: 3, 20, 28, 32, 38, 42, 46, 53, 57, 67.



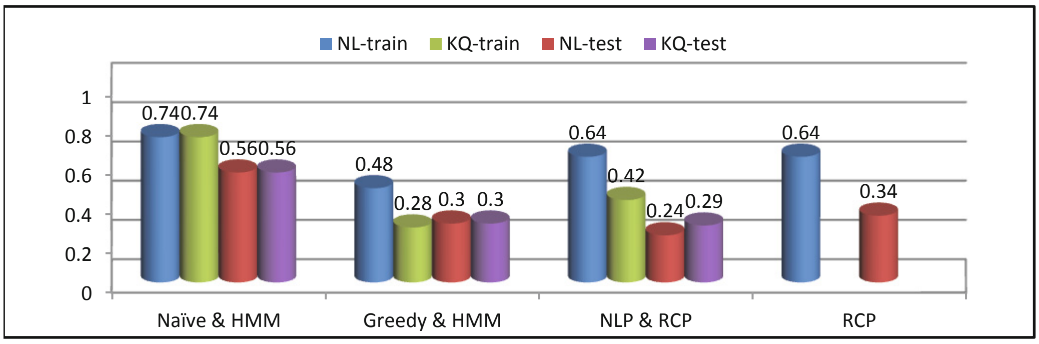
(a) Queries that require query expansion are included.



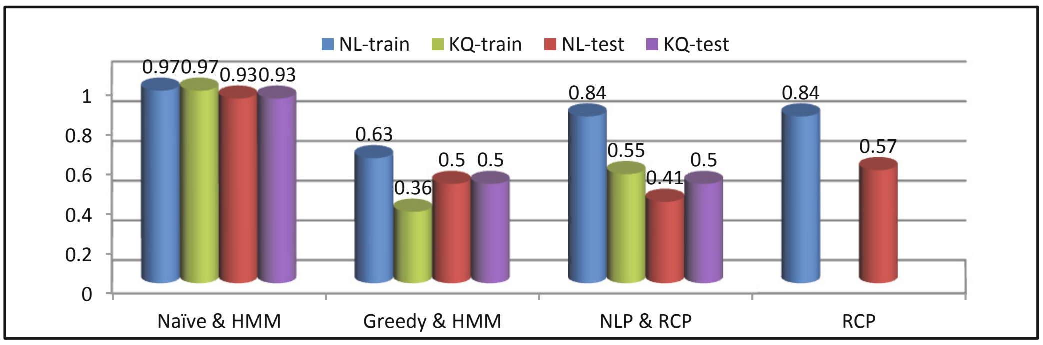
(b) Queries that require query expansion are not included.

Fig. 2. Mean reciprocal rank of query segmentation (first stage)

Naive+HMM outperforms Greedy+HMM, NLP+RCP and RCP in all four experimental settings. Figure 3(a) shows on average 24% higher MRR, although queries requiring expansion are included. In the absence of the queries that required an expansion (Figure 3(b)), Naive+HMM on average by 38% superior to all other approaches and 25% superior to RCP. Note that RCP relies on correct segmentation which in reality is not always a valid assumption. Generally, Naive+HMM being superior to Greedy+HMM can be expected, since the naive approach for segmentation generates more segments from which the HMM can choose. Naive+HMM outperforming RCP (resp. NLP+RCP) is mostly related to RCP (resp. NLP+RCP) often failing to assign the highest rank to the correct disambiguation. One important feature of our approach is, as the evaluation confirms, the robustness with regard to the query expression variance. As shown in Figure 3, Naive+HMM achieves the same MRR on natural-language and the keyword-based representation of queries on both – the train and the test – datasets. Overall, Naive+HMM significantly outperforms our baseline Greedy+HNM as well as state-of-the-art techniques based on NLP. Figure 4 shows the mean of MRR for different values of the threshold θ applied for



(a) Queries that require query expansion are included.



(b) Queries that require query expansion are not included.

Fig. 3. Mean reciprocal rank of resource disambiguation (second stage)

punning the state space. As it can be observed the optimal value of θ is in the range $[0.6, 0.7]$. A high value of θ prevents including some relevant resources and a low value causes a load of irrelevant resources. We set θ to 0.7.

The success of our model relies on transition probabilities which are based on the connectivity of both the source and target node (hub score of source and sink authority) as well as taking into account the connectivity (authority) of all sink states. Especially, employing the HITS algorithm leads to distributing a normalized connectivity degree across the state space. To compare the proposed method for bootstrapping transition probability, we tested two other methods (i.e., normal and Zipfian distribution). Assume the random variable X is defined as the weighted sum of the normalized length of the path (distance) between two states and normalized connectivity degree: $X = \alpha * distance + (1 - \alpha) * (1 - connectivityDegree)$. We bootstrapped the transition probability based on the normal and Zipfian distribution for the variable X. Table 2 shows the MRR of the HMM based on different methods (i.e., normal distribution, Zipfian and the proposed method) employed for bootstrapping the transition probability. The results achieved with the first two methods only led to a low accuracy. The proposed method is superior to the other two methods in all settings.

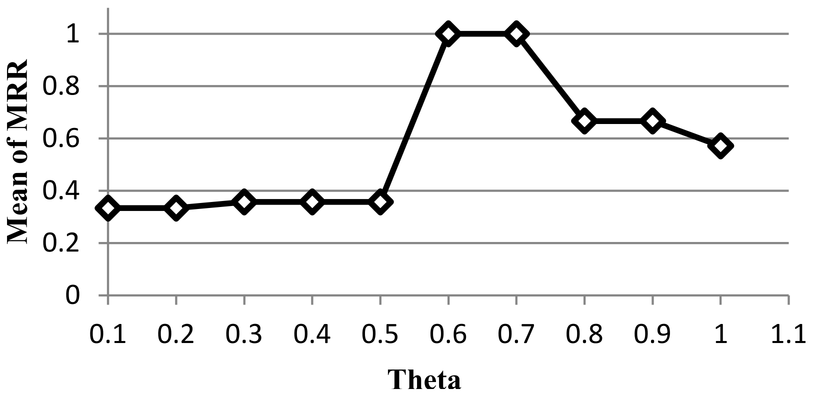


Fig. 4. Mean MRR for different values of θ

Table 2. MRR based on different methods employed in transition probability for 10 queries from train dataset

Query ID	12	15	19	22	23	25	31	33	34	35
Zipf $\alpha = 1$	0.3	0.5	0.25	0.2	0.05	0.05	0.2	0.2	0.2	0.5
Zipf $\alpha = 0.75$	0.3	0.5	0.25	0.2	0.05	0.05	0.16	0.2	0.2	0.5
Zipf $\alpha = 0.5$	0.3	0.5	0.25	0.2	0.05	0.05	0.16	0.2	0.2	0.5
Zipf $\alpha = 0.25$	0.3	0.5	0.25	0.2	0.045	0.05	0.16	0.2	0.2	0.5
Zipf $\alpha = 0$	0.3	0.5	0.25	0.2	0.045	0.05	0.16	0.2	0.2	0.5
The proposed method	1	1	1	1	1	1	1	1	1	1
Normal $\alpha = 1$	1	0.5	0.07	0.25	0.034	0.041	0.3	0.1	1	0.16
Normal $\alpha = 0.75$	1	0.5	0.07	0.3	0.034	0.041	0.125	0.1	1	0.25
Normal $\alpha = 0.5$	1	0.5	0.07	0.3	0.041	0.052	0.14	0.1	1	0.25
Normal $\alpha = 0.25$	1	0.5	0.07	0.3	0.058	0.58	0.2	0.125	1	0.2
Normal $\alpha = 0$	1	0.5	0.1	0.5	0.083	0.045	0.5	0.5	0.5	0.5

7 Discussion and Future Work

We explored different methods for bootstrapping the parameters (i.e. different distributions tested e.g., normal, Zipf) of the HMM. The results achieved with these methods only led to a very low accuracy. The success of our model relies on transition probabilities which are based on the connectivity of both the source and target node (hub score of source and sink authority) as well as taking into account the connectivity (authority) of all sink states. Employing the HITS algorithm leads to distributing a normalized connectivity degree across the state space. More importantly, note that considering a transition probability to the unknown entity state is crucial, since it arranges states with the same emitted segments in a descending order based on their hub scores. Most previous work has been based on finding a path between two candidate entities. For future, we aim to realize a search engine for the Data Web, which is as easy to use as search engines for the Document Web, but allows to create complex queries and returns comprehensive structured query results[9]. A first area of improvements is related to using dictionary knowledge such as hypernyms, hyponyms or co-hyponyms. Query expansion might also, however, result in a more noisy input for

[9] A prototype of our progress in this regard is available at `http://sina.aksw.org`

our model. Thus, a careful extension of our approach and analysis of the results will be required. In addition, we will extend our approach with a query cleaning algorithm. The input query might contain some keywords which semantically are not related to the rest of keywords. Since user usually is looking for information semantically closely related to each other, these unrelated keywords (i.e. noise) should be cleaned.

References

1. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. Technical Report 2003-29 (2003)
2. Beeferman, D., Berger, A.: Agglomerative clustering of a search engine query log. ACM Press (2000)
3. Brenes, D.J., Gayo-Avello, D., Garcia, R.: On the fly query entity decomposition using snippets. CoRR, abs/1005.5516 (2010)
4. Brill, E., Ngai, G.: Man* vs. machine: A case study in base noun phrase learning. ACL (1999)
5. Chieu, H.L., Ng, H.T.: Named entity recognition: A maximum entropy approach using global information. In: Proceedings COLING 2002 (2002)
6. Chuang, S.-L., Chien, L.-F.: Towards automatic generation of query taxonomy: A hierarchical query clustering approach. IEEE Computer Society (2002)
7. Collins, M., Singer, Y.: Unsupervised models for named entity classification. In: SIGDAT Empirical Methods in NLP and Very Large Corpora (1999)
8. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing Search in Context: the Concept Revisited. In: WWW (2001)
9. Guo, J., Xu, G., Cheng, X., Li, H.: Named entity recognition in query. ACM (2009)
10. Joachims, T., Granka, L.A., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. In: SIGIR. ACM (2005)
11. Kelly, D., Teevan, J.: Implicit feedback for inferring user preference: a bibliography. SIGIR Forum 37(2), 18–28 (2003)
12. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM 46(5) (1999)
13. Kraft, R., Chang, C.C., Maghoul, F., Kumar, R.: Searching with context. In: WWW 2006: 15th Int. Conf. on World Wide Web. ACM (2006)
14. Lawrence, S.: Context in web search. IEEE Data Eng. Bull. 23(3), 25–32 (2000)
15. Pu, K.Q., Yu, X.: Keyword query cleaning. PVLDB 1(1), 909–920 (2008)
16. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. CoRR (1995)
17. Risvik, K.M., Mikolajewski, T., Boros, P.: Query segmentation for web search (2003)
18. Shepitsen, A., Gemmell, J., Mobasher, B., Burke, R.: Personalized recommendation in social tagging systems using hierarchical clustering. ACM (2008)
19. Tan, B., Peng, F.: Unsupervised query segmentation using generative language models and wikipedia. In: WWW. ACM (2008)
20. Tan, B., Peng, F.: Unsupervised query segmentation using generative language models and wikipedia. ACM (2008)
21. Uzuner, A., Katz, B., Yuret, D.: Word sense disambiguation for information retrieval. AAAI Press/The MIT Press (1999)

22. Vorhees, E.: The trec-8 question answering track report. In: Proceedings of TREC-8 (1999)
23. Wen, J.-R., Nie, J.-Y., Zhang, H.-J.: Query Clustering Using User Logs. ACM Transactions on Information Systems 20(1) (2002)
24. White, R.W., Jose, J.M., van Rijsbergen, C.J., Ruthven, I.: A simulated study of implicit feedback models. In: McDonald, S., Tait, J.I. (eds.) ECIR 2004. LNCS, vol. 2997, pp. 311–326. Springer, Heidelberg (2004)
25. Yu, X., Shi, H.: Query segmentation using conditional random fields. ACM (2009)
26. Zhu, Y., Callan, J., Carbonell, J.G.: The impact of history length on personalized search. ACM (2008)

APPENDIX B

Towards an Efficient RDF Dataset Slicing[1]

In this appendix, we attach our publication *Towards an Efficient RDF Dataset Slicing* [3] in which we have contributed.

[1] Published as: Edgard Marx, Tommaso Soru, Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, and Karin Breitman, “Towards an Efficient RDF Dataset Slicing”, *Int. J. Semantic Computing* 7.4 (2013) p. 455

International Journal of Semantic Computing
© World Scientific Publishing Company

TOWARDS AN EFFICIENT RDF DATASET SLICING

EDGARD MARX, TOMMASO SORU, SAEEDEH SHEKARPOUR, SÖREN AUER,
AXEL-CYRILLE NGONGA NGOMO

AKSW, Department of Computer Science, University of Leipzig
Augustusplatz 10, 04109 Leipzig, Germany
{*marx,tsoru,shekarpour,auer,ngonga*}*@informatik.uni-leipzig.de*
http://aksw.org

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

In the last years an increasing number of structured data was published on the Web as Linked Open Data (LOD). Despite recent advances, consuming and using Linked Open Data within an organization is still a substantial challenge. Many of the LOD datasets are quite large and despite progress in RDF data management their loading and querying within a triple store is extremely time-consuming and resource-demanding. To overcome this consumption obstacle, we propose a process inspired by the classical Extract-Transform-Load (ETL) paradigm. In this article, we focus particularly on the selection and extraction steps of this process. We devise a fragment of SPARQL dubbed SliceSPARQL, which enables the selection of well-defined slices of datasets fulfilling typical information needs. SliceSPARQL supports graph patterns for which each connected subgraph pattern involves a maximum of one variable or IRI in its join conditions. This restriction guarantees the efficient processing of the query against a sequential dataset dump stream. Furthermore, we evaluate our slicing approach on three different optimization strategies. Results show that dataset slices can be generated an order of magnitude faster than by using the conventional approach of loading the whole dataset into a triple store.

Keywords: RDF Dataset Slicing; SPARQL; Graph processing.

1. Introduction

In the last years an increasing number of structured data was published on the Web as Linked Open Data (LOD). Despite recent advances, consuming and using Linked Open Data within an organization is still a substantial challenge. Many of the LOD datasets are quite large and despite progress in RDF data management their loading and querying within a triple store is extremely time consuming and resource demanding. Examples of such datasets are *DBpedia* (version 3.8)[a] and *LinkedGeoData*[b], which encompass more than 1 billion triples each. Loading these

[a]`http://dbpedia.org`
[b]`http://linkedgeodata.org`, version of May 3rd, 2013

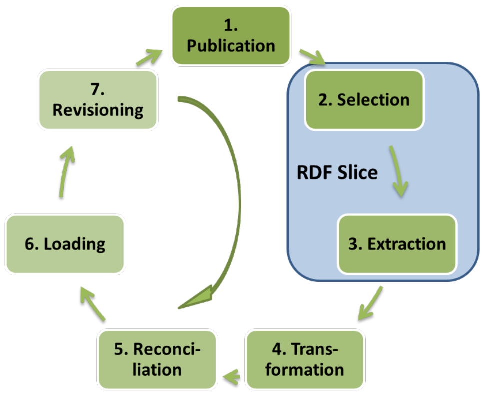


Fig. 1. Linked Open Data consumption process.

datasets into a triple store requires substantial amounts of resources and time (e.g. 8 hours for DBpedia and 100 hours for LinkedGeoData on standard server hardware). Rapid prototyping, experimentation and agile development of semantic applications is currently effectively prevented this way. However, many users are not interested in the whole dataset, but in a very specific part of it. A search engine specialized on entertainment topics, for example, might aim to enrich its search results with facts on actors and movies from DBpedia. A location-based service might want to provide information on points-of-interest in the neighborhood of the users location from LinkedGeoData. In both scenarios, only a tiny fraction of the respective knowledge bases is required: From DBpedia, we need in the first case essentially all instances from the classes `Actor` (2,431 instances) and `Film` (71,715 instances). For the location-based service scenario, we can omit all nodes, ways and relations from LinkedGeoData, which do not belong to the class `point-of-interest` or any of its sub-classes. In that case, 98% of LinkedGeoData can be purged, thus lowering resource requirements and increasing query performance by several orders of magnitude. If we enable users to efficiently extract slices from knowledge bases, which comprise exactly the information they require, building special-purpose Semantic Web applications will become significantly more efficient.

Figure 1 depicts a conceptual LOD consumption process. The process is inspired by the classical Extract-Transform-Load (ETL) process known from data warehousing. However, other than ETL the LOD consumption considers both new dataset versions being published *and* revisions being applied to internally used (parts of) these datasets. The steps are:

(1) *Publication* is a prerequisite for the remaining consumption steps and comprises the publication of an RDF dataset by a data publisher, mostly as a dataset dump or SPARQL endpoint.
(2) *Selection* comprises the definition and specification of a relevant fragment of a

dataset, which is envisioned to be used internally by a consuming organization.
(3) *Extraction* processes the dataset dump and extracts the relevant fragment.
(4) *Transformation* comprises the mapping and mapping execution of the extracted data structure to match organization internal data structures.
(5) *Reconciliation* applies revisions made by the organization to earlier versions of the dataset to the actual version.
(6) *Loading* makes the dataset available for internal services and applications, for example, by means of a SPARQL endpoint.
(7) *Revisioning* allows the organization to apply (manual) changes to the dataset, such as deleting instances changing properties etc. Revisions applied to a certain version of the dataset should be persistent and be automatically reapplied (after an update of the dataset in the respective reconciliation step.

In this article, we focus particularly on the selection and extraction steps. We devise a fragment of SPARQL dubbed SliceSPARQL, which enables the selection of well-defined slices of datasets fulfilling typical information needs. SliceSPARQL supports graph patterns for which each connected subgraph pattern involves a maximum of one variable or IRI in its join conditions. This restriction guarantees the efficient processing of the query against a sequential dataset dump stream. As a result our evaluation shows that dataset slices can be generated an order of magnitude faster than by using the conventional approach of loading the whole dataset into a triple store and retrieving the slice by executing the query against the triple store's SPARQL endpoint.

The remainder of the paper is organized as follows. section 2 briefly introduces some important formalisms such as the RDF data model and SPARQL algebra. section 3 discusses our approach for Linked Data graph selection and extraction. section 6 presents a comprehensive field study comparing conventional and SliceSPARQL Linked Data selection and extraction using a testbed involving a variety of different selections and datasets. section 7 discusses related works. Finally, section 8 concludes with an outlook on future work.

2. Background

According to the current state of the LOD cloud[c], 295 datasets containing more than 31 billion triples have been published [?].

Commonly the selection of subsets of RDF is performed using the SPARQL query language[d]. The SPARQL RDF query language can be used to express queries across diverse data sources. It is composed by three main parts: 1. Query Forms, 2. WHERE clause as well as 3. Solution Sequence and Modifiers (SSM).

The *Query Forms* contains variables that will appear in a solution result. It can be used to select all or a subset of the variables bound in a pattern match.

[c]Retrieved 10th November 2013.
[d]`http://www.w3.org/TR/sparql11-query`

Query Forms are designed to form result sets or RDF graphs. There are the four different select query forms SELECT, CONSTRUCT, ASK and DESCRIBE. The SELECT query form is the most common one and is used to return rows of variable bindings. CONSTRUCT allows to create a new RDF graph or modify the existing one through substituting variables in a graph templates for each solution. ASK returns a Boolean value indicating whether the graph contains a match or not. Finally, DESCRIBE is used to return all triples about the resources matching the query.

The *WHERE clause* is composed by a *Graph Pattern* and some constraints helpers such as *FILTER. OPTIONAL* was designed for situations where there is a necessity to select also some RDF term that is not bound in some BGP. Filters are used to restrict a set of matched RDF terms to a subset where the filter expression evaluates to TRUE. The triple patterns in a BGP could be or not connected by a join condition. BGPs are a composition of one or more triple patterns that contains only variables or both, variables and constants. By rule, a triple pattern cannot be composed only by constants. BGPs are used to select RDF terms from a certain data subgraph and are composed by one or more triple patterns which contains only variables or both, variables and constants. The selected RDF terms are those of the matching subgraphs that was mapped to variables. Please refer to Definition 2 for better understanding.

The use of query Forms and WHERE clauses generates an unordered set of solutions. *Solution Sequence and Modifiers (SSM)* can be applied to this set to generate another sequence or select a portion of the result set. The SSM is composed by six modifiers: ORDER, PROJECTION, DISTINCT, REDUCED, OFFSET and LIMIT. The subsequent formalization of RDF and core SPARQL is closely following [?].

Definition 1. (RDF definition) Assume there are pairwise disjoint infinite sets I, B, and L (IRIs, blank nodes, and RDF literals, respectively). A triple $(v_s, v_p, v_o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF *triple.* In this tuple, v_s is the *subject*, v_p the *predicate* and v_p the *object.* We set $T = I \cup B \cup L$ and call T's elements RDF terms.

In the following, the same notion is applied to *triple patterns* and *triple maps.* An RDF *graph* is a set of RDF triples (also called RDF dataset, or simply a dataset). Additionally, we assume the existence of an infinite set V of variables with $V \cap T = \emptyset$. The W3C recommendation SPARQL is a query language for RDF. By using *graph patterns*, information can be retrieved from SPARQL-enabled RDF stores. This retrieved information can be further modified by a query's solution modifiers, such as sorting or ordering of the query result. Finally the presentation of the query result is determined by the *query type*, return either a set of triples, a table or a Boolean value. The graph pattern of a query is the base concept of SPARQL and as it defines the part of the RDF graph used for generating the query result, therefore *graph patterns* are the focus of this discussion. We use the same graph pattern syntax definition as [?].

Definition 2. (SPARQL Basic Graph Pattern syntax) The syntax of a SPARQL Basic Graph Pattern expression is defined recursively as follows:

(1) A tuple from $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a graph pattern (a *triple pattern*).
(2) The expressions (P_1 AND P_2), (P_1 OPT P_2) and (P_1 UNION P_2) are graph patterns, if P_1 and P_2 are graph patterns.
(3) The expression (P FILTER R) is a graph pattern, if P is a graph pattern and R is a SPARQL constraint.

SPARQL constraints are composed of functions and logical expressions, and are supposed to evaluate to Boolean values. Additionally, we assume that the query pattern is well-defined according to [?]. Table 1 categorizes the types of joins in graph patterns consisting of two triple patterns. For example, the join type SS means that the two triple patterns have the same subject, while SO means that the subject of the first triple pattern is the same as the object of the second triple pattern.

Although SPARQL allows a variety of types of selections, an empirical study over real queries [?] shows that the most frequent *Query Forms* executed against DBpedia and SWDF[e] is SELECT, comprising 96.9% and 99.7% respectively while ASK, CONSTRUCT and DESCRIBE are scarcely used. This study also states that the most common triple patterns found only have a variable at the object position (DBpedia 66.35%; SWDF 47.79%). The authors also conclude that most queries are simple, i.e., 66.41% of DBpedia queries and 97.25% of SWDF just contain a single triple pattern. Another important finding is that joins are typically of the types SS (∼60%), SO (∼35%) and OO (∼4.5%). The study also shows that most of queries (99.97%) have a star-shaped graph pattern, and the chains in 98% of the queries have length one, with the longest path having a length of five.

Regarding published datasets in LOD cloud, in our study we find out the existence of tree types of sorting files. According to the sort type, files could be classified as (1) instance segmented, (1) sorted or (3) unsorted. In (1) instance segmented sort the triples concerning one subject are grouped together, in sequence. Sorted (2) files are those where the subject of the triples fallow the lexical order. For instance, in the lexical order between 10 and 9, 10 preceed 9. Thereafter the set of files that are sorted belongs to the set of files that are instance segmented but the iverse is not true. Finnaly, unsorted (3) files are those that do not belongs to any of the previous sets.

[e]`http://data.semanticweb.org`

Acronym	**Join Type**	**Graph Patterns**
SS	**subject-subject**	$(s_1, p_1, o_1)(s_1, p_2, o_2)$
PP	**predicate-predicate**	$(s_1, p_1, o_1)(s_2, p_1, o_2)$
OO	**object-object**	$(s_1, p_1, o_1)(s_2, p_2, o_1)$
SO	**subject-object**	$(s_1, p_1, o_1)(s_2, p_2, s_1)$
SP	**subject-predicate**	$(s_1, p_1, o_1)(s_2, s_1, o_2)$
OP	**object-predicate**	$(s_1, p_1, o_1)(s_2, o_1, o_2)$

Table 1. Categorization of join types in graph patterns consisting of two triple patterns.

3. RDF Dataset Slicing

The goal of our slicing approach is to compute portions of a given set of RDF streams that abide by a description provided in a restricted SPARQL vocabulary which we call SliceSPARQL. An overview of the approach is given in Figure 2. We base our slicing approach on matching triple patterns of SliceSPARQL queries sequentially against the data read from the dataset dump file.

Definition 3. (SliceSPARQL) SliceSPARQL is the fragment of SPARQL for which each connected subgraph pattern of the SPARQL graph pattern involves a maximum of one variable or IRI in its join conditions.

The process of dataset slicing as depicted in Figure 2 comprises three stages. In the first stage, the SliceSPARQL query is analyzed in order to recognize the maximally connected subgraph patterns and an associated most restrictive triple pattern. Then, the most restrictive pattern is used to extract the matching join candidate triples from the dataset dump file. In the *second stage*, the datasets are processed again in order to verify which of the join candidates match the remaining triple patterns of the respective SliceSPARQL's maximally connected subgraph pattern.

Definition 4. (Most Restrictive Triple Patterns) For a given triple pattern t, the number of constants contained in t is denoted by t_c. The set of the most restrictive triple patterns of a SliceSPARQL query is the set of triple patterns having maximum t_c.

The type of joins in graph patterns consisting of two triple patterns can be categorized in six categories. Table 1 shows the list of all possible join types.

Definition 5. (Set of join candidates) A graph pattern p matching the triple t is denoted by $p(t)$. Consider all maximally connected subgraph patterns P of a SliceSPARQL query with respect to the join position. For a given graph pattern $p \in P$, the set of the join candidates is the set of all RDF terms in the join position of triples matching p. This set is denoted by C_p and formally is defined as:

$$C_p = \{RDFTerm(t) | t \in D \wedge p(t) \wedge p \in P\}$$

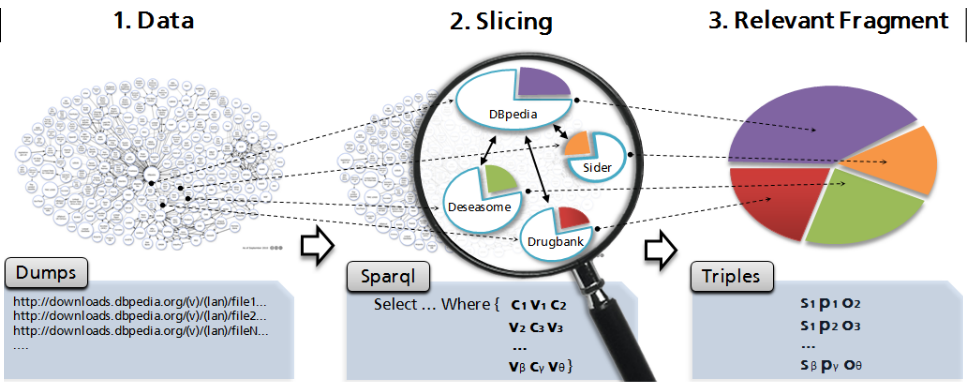

1. Data
2. Slicing
3. Relevant Fragment
DBpedia
Sider
Deseasome
Drugbank
Dumps
http://downloads.dbpedia.org/(v)/(lan)/file1...
http://downloads.dbpedia.org/(v)/(lan)/file2...
http://downloads.dbpedia.org/(v)/(lan)/fileN...
....
Sparql
Select ... Where { c1 v1 c2
v2 c3 v3
...
vβ cγ vθ }
Triples
s1 p1 o2
s1 p2 o3
...
sβ pγ oθ

Fig. 2. Overview of the RDF data slicing approach: (1) dataset dumps are accessed on the Web, (2) the SliceSPARQL patterns are evaluated against the sequential dataset dump stream, (3) relevant fragments from different datasets can be combined into an application specific knowledge base.

Considering the set of all patterns P, the set of join candidate C is the intersection of all C_p.

$$C = \bigcap_{\forall p \in P} C_p$$

Finally, in the *third stage*, we process the dataset once more and select all triples containing an RDF term which matches all patterns in SliceSPARQL.

The two final stages are omitted in any of the following situations: 1. All triple patterns are disjoint; 2. The dataset is segmented by subject and the graph pattern contains only SS-joins; 3. The dataset is sorted and the graph pattern contains only SS, SO or SP join conditions. The first stage can be also omitted if the triple patterns do not contain any variables. In this (rather rare) case the constants used for joining are the join candidates.

We decided to perform the extraction in the three stages for efficiency reasons. Note that the data could be extracted in just a single stage, but doing that could take as long as loading the data into a triple store. The last stage could also be omitted if we select all possible triples already in the second stage. This would save a little bit space once possible join candidates were selected in the first stage, but most of the candidates would not fulfill the triple patterns. For example, in case that we aim to create a slice with data about cities from New York state all world cities would be candidates, but just a small portion are actually relevant. A three-stage process is better suited, since it allows to determine the desired slice more precisely in the first two stages and to extract the required information in a targeted way in the third stage.

Example 1.

Let us look at the selection of drugs from the following example dataset composed out of triples extracted from DBpedia:

```
3 dbr:Aspirin    a                 dbo:Drug .
4 dbr:Aspirin    rdfs:label        "Aspirin"@en .
5 dbr:California rdf:type          dbo:Place .
6 dbr:California dbo:largestCity   dbpedia:Los_Angeles .
```

The following query can be used to select all instances of the DBpedia class `Drug` and all resources referring to them:

```
1 SELECT * WHERE { {
2     ?s    a     dbo:Drug .
3     ?s    ?p    ?o.
4   } UNION {
5     ?s1   a     dbo:Drug .
6     ?o1   ?p1   ?s1. }
7 } }
```

In the first stage the SPARQL query is split into two BGPs (lines 2-3 and 5-6) each containing two triple patterns. The first one has an SS join condition using the variable ?s (SSv). The second one has an SO join condition using the variable ?s1 (SOv). The more restrictive triple pattern is then chosen for each of them. In both cases, the first triple pattern (i.e. line 2 and 5) is the more restrictive one. In the first stage, all triples in the dataset matching one of these restrictive patterns are then selected as join candidates. Note that the triple in line 1 of the given dataset matches both more restrictive patterns:

In the second stage the RDF terms of the join candidates used in a join condition (variables or constants) are then used to evaluate which of the join candidates matches all other triple patterns in each of the BGPs. In our example, the RDF term of our candidate fully matches only one BGP. The RDF term `dbpedia:Aspirin` matches both triples `?s a dbo:Drug` and `?s ?p ?o`.

In the last and final stage all triples in which the RDF term in the join condition fulfills all triple patterns of some BGP are extracted, i.e.:

```
1 dbr:Aspirin  a           dbo:Drug
2 dbr:Aspirin  rdfs:label  "Aspirin"@en.
```

4. Complexity analysis

In order to better understand the time complexity of the approach we look at all types of join in Table 1. An extraction can be performed in one, two or three stages depending on the SliceSPARQL pattern and the representation of the source dataset. The complexity of each of this methods is shown in Table 2 and described in the sequel. Please note that the join candidates are stored in a B-tree structure.

Case 1. Selection in unsorted datasets with any type of joins.

Case 1.1. The time complexity of the process where the join RDF term is a variable (e.g. `Q1, Q3, Q4, Q5` in Table 4) is $O(n \log n)$. We call this the *generic method* because it can be used for unsorted datasets with any type of join. The complexity of the generic process is as follows:

The *first stage* comprises the (1) selection of a most restrictive pattern and the (2) selection of join candidates. The selection of a most restrictive pattern is carried out by retrieving the triple patterns in the graph pattern with the least variables. With t we refer to the number of triple patterns in the graph pattern and as we discussed in section 2 t is small (i.e., $t << n$). After selecting the most restrictive triple pattern, the selection of the join candidates is performed by reading the dataset sequentially. Let n be the number of triples in the dataset. Each triple that matches the most restrictive triple pattern requires an insertion in a B-tree structure, which can be performed in $O(\log m)$where m is the number of elements in the B-tree, i.e. the size of the join candidate dataset. Consequently, the first stage can be described with the follow formula and complexity $t + \sum_{m=1}^{n} logm \approx O(n \log n)$.

The *second stage* consists of reading the target dataset and checking each triple for whether it matches some of the triple patterns. This can be done in $O(t)$. If the triple matches some triple pattern, then an update is necessary. Such an update can be carried out in $O(\log m)$. Taking the size n of the dataset into account, the second stage can be carried out in $\sum_{m=1}^{n} t \times \log m = t \times O(n \log n)$.

The *third stage* performs the actual extraction. For each triple in the dataset it is checked if the triple contains an RDF term from the join candidates matches all triple patterns from some of the graph patterns. Searching in the stored join candidates can be performed in $\log m$. Hence, the time complexity of this stage is $\sum_{m=1}^{n} \log m = O(n \log n)$.

The final complexity of the generic method is the sum of the complexity of each stage, i.e. $t \times O(n \log n) \approx O(n \log n)$.

Case 1.2. The time complexity in case of a constant join RDF term (e.g. Q2 in Table 4) is $O(n)$.

Since the join constant in the graph pattern is already defined, there is no necessity to perform stage one. The slicing takes two stages. (1) First checking if all triple patterns appear in the dataset and a second one selecting the matching triples. As the both stages require $O(n)$ the final complexity is $O(n)$.

Case 2. The extraction from sorted datasets using SO and SP triple patterns where the most restrictive triple pattern has is joined via the object or predicate respectively can be performed in one stage.

Case 2.1. The RDF join term is a constant and the most restrictive triple pattern contains a constant as subject (e.g. Q7 in Table 4).

In the case of the RDF join term being a constant, the extraction can be performed using two binary searches, one to find the target subject and another one to find the target object ($\sum_1^2 \log n = \mathrm{O}(\log n)$).

Case 2.2. The join RDF term is a variable (SOv) (Q8 in Table 4).

One stage through the dataset leads to all triples matching the triple pattern having the join RDF term as object or predicate. However, after finding these triples a binary search then finds the corresponding triple pair. Consequently, the time complexity is $\sum_{n=1}^{n} logn = \mathrm{O}(n \log n)$. Nevertheless, despite *Case 1* and *Case 2* have the same complexity they require a different number of stages (three for *Case 1* and one for *Case 2*).

Some other triple patterns can also be profiled in one stage e.g. SO where the most restrictive pattern has the join RDF term as subject. In that case, choosing the less restrictive triple pattern can also lead to extracting the desired data in one stage. As the process with tree stages selects the more restrictive triple pattern, we can say that the first approach requires $3 * mlogn$ where $m \leq n$. For typical RDF term distributions $3 * m \leq n$. For instance, 99% of properties instances in DBpedia belongs to only 10% of the properties. A similar distribution can be found in the generic-infobox, mapping-base infobox and pagelinks datasets in terms of node indegree [?].

Case 3. Extraction from sorted or subject segmented dataset dumps.

Case 3.1. The complexity with a variable as join RDF term (*SSv*) and the dataset segmented by subject (Q2 in Table 4) is $\mathrm{O}(n)$.

Since the number of triples for a particular subject segment is small enough to fit into memory, there is no need for more than one stage. Loading each segment of the instance in memory and checking the patterns contained in SS can be performed in one stage. The time complexity in this case is $\sum_{m=1}^{n} 5 = \mathrm{O}(n)$.

Case 3.2. The complexity with a constant join RDF term (*SSc*) and the dataset being sorted by subject (Q6 in Table 4) is $\mathrm{O}(\log n)$.

The extraction can be performed with one binary search to find the target subject, i.e. $\sum_1^2 logn = \mathrm{O}(\log n)$.

5. Implementation

To profile the implementation of the approach we use a set of applications developed in Java. To ease the file management we create three applications for the (1) download, (2) sorting and (3) decompression of files respectively. All the tests were

Placeholder	**Join type**	**Unsorted**	**Instance Segmented**	**Sorted**
variable	SS	O($n \log n$)	O(n)	O(n)
	PP	O($n \log n$)	O($n \log n$)	O($n \log n$)
	OO	O($n \log n$)	O($n \log n$)	O($n \log n$)
	SO	O($n \log n$)	O($n \log n$)	O($n \log n$)
	SP	O($n \log n$)	O($n \log n$)	O($n \log n$)
	OP	O($n \log n$)	O($n \log n$)	O($n \log n$)
constant	SS	O(n)	O(n)	O($\log n$)
	PP	O(n)	O(n)	O(n)
	OO	O(n)	O(n)	O(n)
	SO	O(n)	O(n)	O($\log n$)
	SP	O(n)	O(n)	O($\log n$)
	OP	O(n)	O(n)	O(n)

Table 2. Time complexity from different join types.

profiled using *Virtuoso Server*[f], for this propose we implemented a (4) Virtuoso utility application. The Virtuoso application was built using the Virtuoso JDBC Driver[g]. The rationale of using the Virtuoso JDBC Driver was to communicate directly with the Virtuoso instance without the overhead generated by other methods as HTTP clients. The Virtuoso utility allows dropping graphs, loading dump files and profiling queries natively as ISQL client. To load the dump files into Virtuoso the function `ld_dir`[h] was used in order to speed up the loading.

To profile the slicing approach we created another application (5). The slicing approach was developed and main tested with N-Triple files and is due to the SliceSPARQL fragment not compatible with SPARQL 1.1[i]. Thereafter two other versions of the slicing application with different optimization strategies subsection 5.1 were created.

The storage of join candidates was achieved by using a custom file storage which relies on B-trees. The structure of the each stored join candidate is composed by: *Join Term*, *GP position*, *Join Condition* and *Match Vector*. The *GP position* is the position where the GP appears in query e.g. the first GP is zero, second GP is one and the n-th position is n minus one. The *join term* is the term used to join triple patterns in a GP. The *join condition* is one of the join conditions listed in Table 1. The *match vector* is a string with length n where n is the number of triple patterns in BGP. Each position of the *match vector* represents a triple pattern, containing one if the triple pattern matches or zero if not.

5.1. *Performance improvement*

So far we have presented the approach and its basic time complexity. In this section we describe how the time performance can be further improved. Firstly, we explain

[f]`http://virtuoso.openlinksw.com/`
[g]`http://docs.openlinksw.com/virtuoso/VirtuosoDriverJDBC.html`
[h]`http://docs.openlinksw.com/virtuoso/fn_ld_dir.html`
[i]`http://www.w3.org/TR/sparql11-query/`

why the entire process can be carried out with parallel hardware. Later, we show another way to reduce runtime by using a cache in the slicing stage.

5.2. *Parallelization*

In general, two main reasons subsist behind parallelization:

- Computation can be better distributed in multi-core machines;
- Idle time can be exploited to execute other operations.

In stages two and three for unsorted files (`Q1`), described in section 4, our approach performs database queries in order to know whether the URI in a placeholder is one of the candidates selected in stage one. Since the read/write operation in hard drives is not immediate, the processor stays on idle for a certain amount of time waiting for the database to respond [?]. By using parallel computing, the processor can use this time interval to profile another read/write operation on the hard drive. The rule above applies for local streams as well as for remote files.

In our case, as depicted in Figure 3, we will focus on two types of parallel work:

- Parallelization by file, wherein each file – or dataset partition – can be assigned to a different thread;
- Parallelization by blocks, wherein each block – or file partition – can be assigned to a different thread.

The basic assumption is that all the entries stored in one or more N-Triple files are independent from each other and can be processed separately. However, the first type of parallelization might not lead to good results, especially when files have different sizes. Thus, splitting files into blocks of the same size is required to optimize the synchronous job.

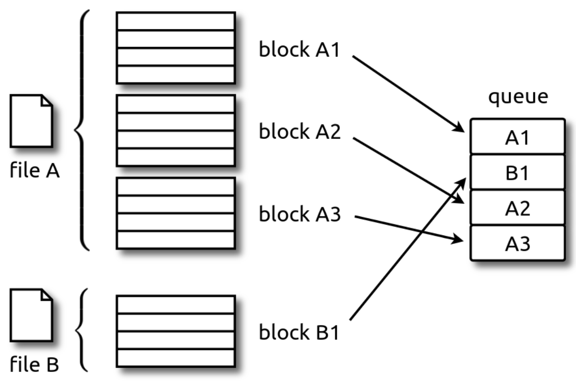


Fig. 3. Each file is divided into blocks which are queued randomly before being processed.

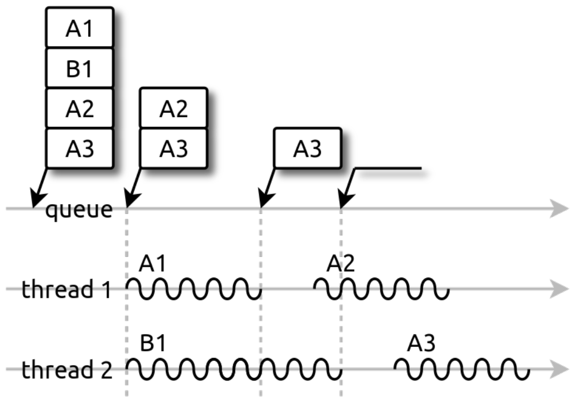


Fig. 4. In this example, four blocks are popped from the queue and processed by two threads.

5.3. *Cache*

The basic idea behind using cache is that we can avoid many database requests by caching the URIs of the triples which have been already fetched. As mentioned before in Definition 2, a query is generated for each triple that matches at least one Basic Graph Pattern. The aim of such query is to know whether the URI in the placeholder is one of the selected candidates. Intuitively, if among the Basic Graph Patterns there exists a triple pattern with no constants, i.e. `?s ?p ?o`, then any triple is a candidate.

5.4. *Domain*

In order to further optimize the execution runtime, a reduction on the working domain can be applied. This is possible because triples involving placeholders require to have object properties. Thus, many database accesses can be avoided by discarding triples that show a literal in the placeholder position. Such situation may occur in three cases, i.e. when the join type is `OO`, `SO` or `OP` (see Table 1).

6. Evaluation

The goal of our evaluation was to determine: (1) How efficient is the slicing approach for various queries? (2) How does the slicing scale for datasets of different size? (3) How does our approach compare to the traditional approach (i.e. loading complete dumps into the triplestore and extracting by querying)? All files generated during the evaluation as well as logs and tables are available online[j].

[j]`http://aksw.org/projects/RDFSlice`

Dataset	**Size**	**Triples**	**Entities**
DBpedia	102 GB	283,928,812	22,857,222
DBpedia-slice	29 GB	125,554,842	13,410,215
DrugBank	98 MB	517,150	19,696
Sider	16 MB	91,569	2,674
Diseasome	12 MB	72,463	8,152

Table 3. Evaluation datasets statistics.

6.1. *Experimental Setup*

We used four interlinked datasets, i.e. Drugbank, Sider, Diseasome and DBpedia Version 3.8 for the evaluation. Table 3 shows the sizes of these datasets. DBpedia-slice refers to a version of DBpedia comprising all DBpedia datasets excluding the *page links undirected* dataset. We selected Drugbank, Sider, Diseasome and DBpedia because they are a fragment of the well interlinked part of Linked Open Data. Especially, DBpedia is an ideal case with respect to size as it is very large. Two students expert to SPARQL and schema of the underlying datasets created eight SPARQL queries shown in Table 3. The provided queries take into account the two most frequent type of join (i.e. `SS` and `SO`) as well as different time complexities. The type of the joins of the associated BGPs and the related dataset are shown. For instance, the query `Q1` running on DBpedia contains eight triple patterns which can be divided to four disjoint BGPs having as join type either subject-subject (SS) or subject-object (SO). We do not take queries into account containing patterns with join types SP, OP and OO, since they seem to be very rare in real SPARQL queries (5%) [?] and have the same complexity as in the SO case. We measured the performance of our slicing approach in terms of `runtime` and `memory consumption`. All experiments were carried out on a Windows 7 machine with an Intel Core M 620 processor and 6GB of RAM and a 230GB SSD.

6.2. *Results*

Figure 5 and Figure 6 shows the `runtime` versus `memory` (being either explored or used) for four parameters of the entire slicing process. These four parameters are:

(1) *Explored graph:* The size of the associated graph of the underlying file which is being explored.
(2) *RAM:* The size of RAM memory which is occupied.
(3) *Slice:* The size of the generated slice from data.
(4) *Disk space:* The size of the disk used to run the application (excluding the slice size).

These parameters are recorded after reading and processing 1 MB and 1 GB for small and large files respectively. Accordingly, the diagrams a-f in Figure 5 represent the above parameters for the Diseasome, Drugbank, Sider, DBpedia and DBpedia-slice datasets, respectively. A general behavior which can be observed in all diagrams is: During the slicing process, the associated dataset dump is analyzed maximum three times. Therefore, the diagram of the explored graph shows three hops except of the 5(f). That is due to DBpedia files being segmented by subject and the the query Q2 containing only subject-subject joins. Thus, one exploration suffices for slicing. With respect to the slice graph, since only in the last pass matches are being found and stored, the size is increasing. Monitoring the last two graphs reveal that a maximum 50 MB of RAM and a very low amount of disk space are occupied. More interestingly, a short while after starting the process, RAM and disk usage are remaining fixed, since in the entire process only join candidates are required to be stored.

With regarding to the optimizations discussed in subsection 5.1, we introduce two evaluations over DBpedia dataset in Figure 6. Both evaluations are using costum domain optimization, but with different strategies. In 6(a) the tool utilize cache to avoid many database queries. In 6(b) both strategies cache and parallelization are used. The results shows that the use of cache achieve 25% of gain while the use of parallelization and cache 13.5%. Further investigations are needed in order to indentify the reasons behind the performace decrease using parallelization.

Table 5 compares total runtime requiring for slicing DBpedia on data being available in triple store and files. Since an advantage of our approach is the extraction directly from files, the load time (i.e. loading data into the triple store) was taken into account. From the five queries (i.e `Q1, Q2, Q6, Q7, Q8`) used in this experiment, four queries (i.e. `Q2, Q6, Q7, Q8`) perform an order of magnitude faster in comparison to the total time computed over triple store (800% faster). Unlike the other queries, the total runtime of `Q1` is very high (still 18% faster than the triple store approach). That is due to the fact that the query is applied on unsorted files and contains triple patterns of the join type SO. Furthermore, Table 6 presents the extraction time in the unsorted version of underlying datasets using the queries i.e. `Q1, Q3, Q4 and Q5`. Although the queries `Q1, Q3 and Q4` have both the `SS` and `SO` join type, the selection time is considerable low because due to the small size of the underlying datasets. In case of DBpedia and DBpedia-slice, the effect of file size is more tangible (70% decrease in the file size leads to a 70% decrease in the extraction time). Table 7 compares total runtime requiring for slicing DBpedia using different optimization strategies.

6.3. *Slicing Interlinked Data*

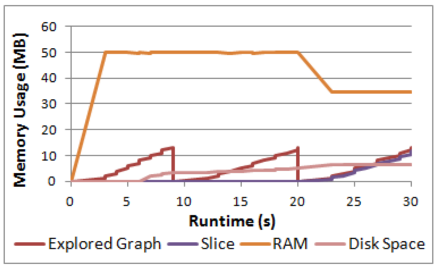
Linked Data enables data to be connected from different sources. Considering the employed datasets, as it can be seen in Figure 7 the classes representing drugs

Query	**Triple Patterns**	**Join**	**Dataset**
Q1	`{?s a dbo:Drug. ?s ?p ?o.} {?s1 a dbo:Drug. ?o1 ?p1 ?s1.} {?s2 a dbo:Disease. ?s2 ?p2 ?o2.} {?s3 a dbo:Disease. ?o3 ?p3 ?s3.}`	**SS+SO**	**DBpedia**
Q2	`{?s a dbo:Drug. ?s ?p ?o.} {?s1 a dbo:Disease. ?s1 ?p1 ?o1.}`	**SS+SS**	**DBpedia**
Q3	`{?s a diseasome:diseases. ?s ?p ?o.} {?s1 a diseasome:diseases. ?s1 ?p1 ?o1.}`	**SS+SO**	**Diseasome**
Q4	`{?s a DrugBank:Drugs. ?s ?p ?o.} {?s1 a DrugBank:Drugs. ?s1 ?p1 ?o1.}`	**SS+SO**	**DrugBank**
Q5	`{?s a Sider:Drugs. ?s ?p ?o.} {?s1 a Sider:Drugs. ?s1 ?p1 ?o1.}`	**SS+SO**	**Sider**
Q6	`{dbr:Cladribine dbo:iupacName ?o. dbr:Cladribine ?p1 ?o1.}`	**SS**	**DBpedia**
Q7	`{dbr:Delirium dbo:wikiPageWikiLink ?o. ?o ?p ?q.}`	**SO**	**DBpedia**
Q8	`{?s1 dbo:lastWin ?o. ?o ?p ?o1.}`	**SO**	**DBpedia**
Q9	`{?s a dbo:Drug. ?s ?p ?o.} {?s1 a dbo:Drug. ?o1 ?p1 ?s1.} {?s2 a dbo:Disease. ?s2 ?p2 ?o2.} {?s3 a dbo:Disease. ?o3 ?p3 ?s3.} {?s4 a dbo:Drug. ?s4 owl:sameAs ?o4. ?o4 ?p5 ?o5. } {?s5 a dbo:Disease. ?s5 owl:sameAs ?o6. ?o6 ?p6 ?o7.}`	**SS+SO**	**DBpedia**

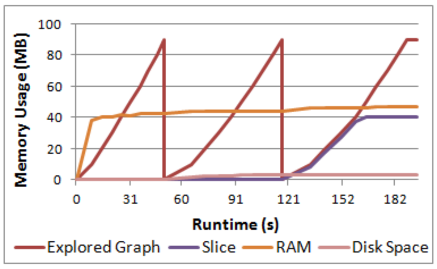
Table 4. Evaluation queries.

in Drugbank , DBpedia and Sider are linked using `owl:sameAs` and diseases from Diseasome are linked to drugs in Drugbank using `possible Drug` and `possible Disease target`. Furthermore, diseases from Diseasome are linked to diseases in DBpedia using `owl:sameAs`. Diseases and side effects between Sider and Diseasome are linked using the `owl:sameAs` property. An interesting and novel aspect of our slicing approach is that it is applicable to interlinked datasets.

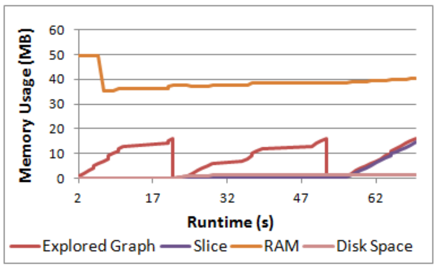
With this respect, `Q9` contains patterns requiring traversal of `owl:sameAs` over the employed interlinked datasets.



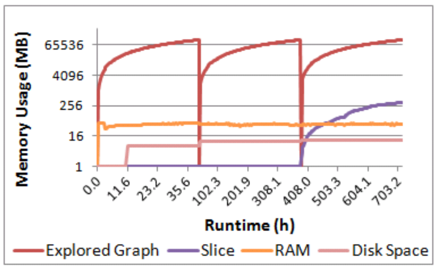
(a) Slicing unsorted dataset *Diseasome* using query Q3.



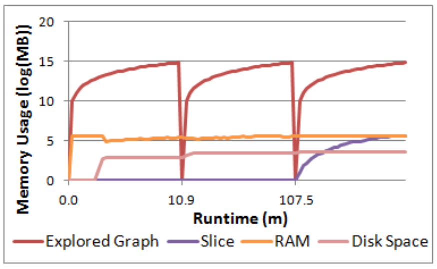
(b) Slicing unsorted dataset *Drugbank* using query Q4.



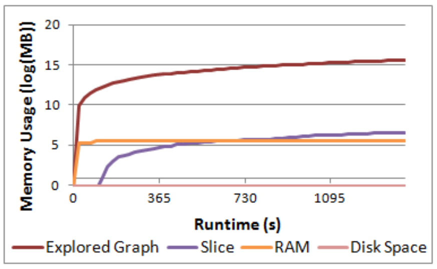
(c) Slicing unsorted dataset *Sider* using query Q5.



(d) Slicing unsorted dataset *DBpedia* using query Q1.



(e) Slicing unsorted dataset *DBpedia-slice* using the Q1.



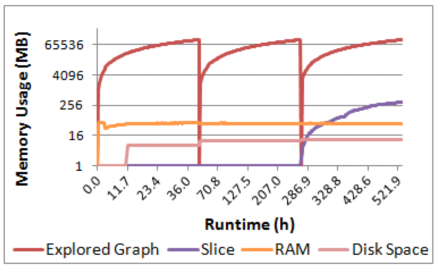
(f) Slicing sorted dastaset *DBpedia* using query Q2.

Fig. 5. Slicing different datasets without optimization with different types of queries in log scale.

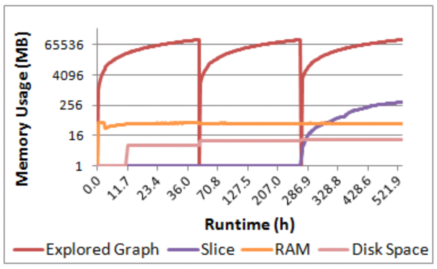
7. Related Work

Our approach and the SliceSPARQL fragment are meant to be part of a broader project dubbed *RDFSlice* [k]. To the best of our knowledge, [?] is the first work specifically targeting RDF data slicing. However, it is related to approaches in the three areas of RDF data a) crawling, b) streaming and c) replication, which we briefly discuss in the following subsections.

[k]http://aksw.org/projects/RDFSlice



(a) Slicing unsorted dataset *DBpedia* using cache.



(b) Slicing unsorted dataset *DBpedia* using cache and parallelization.

Fig. 6. Slicing DBpedia with different optimization strategies using Q1 in log scale.

Approach	**Query**	**Sort**	**Load**	**Extraction**	**Total**	**% Gain**
Triplestore	**Q1**	**-**	**875.6**	**1.8**	**877.4**	18.6
File		**-**	**-**	**713.8**	**713.8**	
Triplestore	**Q2**	**-**	**875.6**	**2.1**	**877.7**	94.8
File		**-**	**-**	**45.9**	**45.9**	
Triplestore	**Q6**	**-**	**875.6**	**0.0**	**875.6**	91.4
File		**75.3**	**-**	**0.0**	**75.3**	
Triplestore	**Q7**	**-**	**875.6**	**0.0**	**875.6**	91.4
File		**75.3**	**-**	**0.0**	**75.3**	
Triplestore	**Q8**	**-**	**875.6**	**2.95**	**878.55**	89.4
File		**75.3**	**-**	**17.38**	**92.68**	

Table 5. Total runtime in minutes requiring for slicing DBpedia against five different queries on data being available in triple store and files without optimization.

7.1. *Crawling*

RDF data crawlers harvest and index RDF content from the Web of Data and Documents. *MultiCrawler* [?] allows to extract information not only from HTML documents but also from structured data on the Web. It focuses on locating relevant links to content in order to extract data. MultiCrawler explores the use of not only

Dataset	**Query**	**Extraction**
DBpedia	Q1	713.8
DBpedia-slice	Q1	202.2
Drugbank	Q4	3.2
Sider	Q5	1.2
Diseasome	Q3	0.5

Table 6. Extraction time in minutes for unsorted datasets of different sizes without optimization.

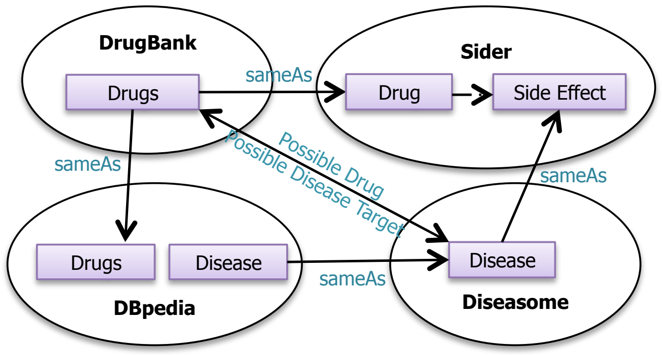


Fig. 7. Schema interlinking for four datasets i.e. DBpedia, DrugBank, Sider, Diseasome.

well-known text indexing but also structured data indexing, by converting all non-structured information found into corresponding structured data. This approach is useful for indexing and finding documents with relevant content, but not to extract fragments of larger datasets. *LDSpider* [?] is a lightweight LOD crawler for integration into Semantic Web applications. LDSpider can be used to traverse LOD content and deliver the extracted data via an API to the application through listeners. The application itself has to select the relevant content. One of LDSpider's main features is the capability to process a variety of Web data formats including Turtle, RDF/XML, Notion 3 and RDFa employing different crawler strategies as breadth-first and load-balancing. *Semantic Web Client library* [?] was developed to crawl the Web of Data in order to facilitate query answering. The rationale is to discover relevant information in the crawled structured data from different sources during query execution time. Similar as crawlers RDFSlice can be used to retrieve and extract relevant RDF data from the Web. However, our RDF data slicing approach focuses on structured data in large files, it does not traverse links and uses specific selection criteria (i.e. SliceSPARQL graph patterns) instead of heuristics.

7.2. *Streaming*

Linked Streaming Data (LSD) is an extension of the RDF data model to support the representation of stream data generated from sensors and social network. They are designed for continuous query data with high rate of change, e. g. once per second.

Strategy	**Query**	**Extraction**	**% Gain**
Cache	Q1	532.3	25.4
Cache and Parallelization	Q1	421.5	41.0

Table 7. Extraction time in minutes for DBpedia dataset using different optimization strategies.

There are already many proposed approaches for RDF Streaming as CQELS [?], Streaming SPARQL [?], C-SPARQL [?], Sparql *stream* [?] and EP-SPARQL [?]. These approaches work in a similar fashion: In LSD approaches, the user defines a time window in which the data will be selected. When the window expires, the data collected is then used to query. The approaches could also use some Linked Data to enrich the information or easily the selection. Differently from the SPARQL Streaming approaches, the Slicing is not designed for querying streaming data. Rather, we aim to extract relevant fragments from large files in the distributed static RDF LOD. Nevertheless, Slicing could also be used as a prefilter in RDF streaming data, helping to select a relevant subset of the data during the time window.

7.3. *Replication*

Although data replication is a well known study in database field to improve performance and availability, there is still a lack in methods and approaches into concerns Linked Data. We also observed that, most of the existing approaches focus in triple stores instead of dump files, their work are also simplified by focusing in managing existing instances rather than create new ones. Some related problems are data selection and synchronization. RDFSync [?] profile data replication between triple stores by decomposing the graphs into smaller Minimum Self-Contained Graphs (MSGs) and comparing their hashes. However, RDFSync does not take into account relevant data replication. Another related work is sparqlPuSH [?]. SparqlPuSH works as a notification message system, notifying the servers registered as listeners about changes into the triple store. SparqlPush could also provide relevant data replication with some restriction by the use of subsection notification of PubSubHubbub protocol[1].

7.4. *Future work*

The number of links in the Linked Data cloud has been growing dramatically since its conception [?], as the 4^{th} Linked Data principle recommends to include links to other URIs for discovering more data [?]. Several Link Discovery frameworks [?, ?] and approaches [?, ?, ?, ?] aim at having the network more and more connected. Hence, we expect the Slicing process to be harder and slower in the future. A solution to this problem can be to reduce the number of considered links, by assigning weights to them or filtering them by type.

Moreover, we will investigate what is the relationship between the number of threads and the execution runtime, according to Göetz's equation for the recommended number of threads:

$$N_{threads} = N_{cpu} \cdot U_{cpu} \cdot \left(1 + \frac{W}{C}\right) \tag{1}$$

[1]`http://code.google.com/p/pubsubhubbub/`

where U is the CPU usage (%) and W/C is the ratio of wait time to compute time [?].

8. Conclusions

In this article we presented an approach facilitating Linked Data consumption by effectively selecting relevant parts and efficiently extracting these from very large RDF datasets. We deem this to be a major step towards simplifying the consumption of large RDF datasets.

We see this work as the first step in a larger research agenda to dramatically improve Linked Data consumption. The presented approach focuses only on two out of six stages of the consumption process. In future, we aim to develop and integrate support for subsequent stages such as transformation, reconciliation, and revisioning. We envision that organizations will thus be empowered to seamlessly integrate LOD data into their internal processes and applications. A particular challenge is the mapping of the LOD data to existing internal information structures and the establishment of a co-evolution between private and public data involving continuous update propagation from LOD sources while preserving revisions applied to prior versions of these datasets.

Acknowledgement

This work was partly supported by CNPq, under the program Ciências Sem Fronteiras, by Instituto de Pesquisa e Desenvolvimento Albert Schirmer (CNPJ 14.120.192/0001-84) and by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

Bibliography

[1] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data", *Journal of Web Semantics Science, Services and Agents on the World Wide Web*, 2014.

[2] Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler, "Generating SPARQL queries Using Templates", *Web Intelligence and Agent Systems Journal* 11.3 (2013) pp. 283–295.

[3] Edgard Marx, Tommaso Soru, Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, and Karin Breitman, "Towards an Efficient RDF Dataset Slicing", *Int. J. Semantic Computing* 7.4 (2013) p. 455.

[4] Sebastian Tramp, Philipp Frischmuth, Timofey Ermilov, Saeedeh Shekarpour, and Sören Auer, "An architecture of a distributed semantic social network", *Semantic Web* 5.1 (2014) pp. 77–95.

[5] Saeedeh Shekarpour and Sören Auer, "Query Reformulation on RDF Knowledge Bases using Hidden Markov Models", *Submitted to the Eighth International Conference on Web Search and Web Data Mining, WSDM 2015*, 2015.

[6] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "Question Answering on Interlinked Data", *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, 2013 pp. 1145–1156.

[7] Saeedeh Shekarpour, Konrad Höffner, Jens Lehmann, and Sören Auer, "Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features", *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, 2013 pp. 191–197.

[8] Edgard Marx, Saeedeh Shekarpour, Sören Auer, and Axel-Cyrille Ngonga Ngomo, "Large-scale RDF Dataset Slicing", *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, 2013.

[9] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "Keyword-Driven Resource Disambiguation over RDF Knowledge Bases", *Semantic Technology, Second Joint International Conference, JIST 2012, Nara, Japan, December 2-4, 2012. Proceedings*, Springer, 2012 pp. 159–174.

[10] Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler, "Keyword-Driven SPARQL Query Generation Leveraging Background Knowledge", *Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2011, Campus Scientifique de la Doua, Lyon, France, August 22-27, 2011*, 2011 pp. 203–210.

[11] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "Query Segmentation and Resource Disambiguation Leveraging Background Knowledge", *Proceedings of WoLE Workshop*, 2012.

[12] Saeedeh Shekarpour, "DC Proposal: Automatically Transforming Keyword Queries to SPARQL on Large-Scale Knowledge Bases", *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part II*, Springer, 2011 pp. 357–364.

[13] Axel-Cyrille Ngonga Ngomo and Sören Auer, "LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data", *Proceedings of IJCAI*, 2011.

[14] Axel-Cyrille Ngonga Ngomo and Klaus Lyko, "EAGLE: Efficient Active Learning of Link Specifications using Genetic Programming", *Proceedings of ESWC*, 2012.

[15] Christian Bizer, Tom Heath, and Tim Berners-Lee, "Linked data-the story so far", *International Journal on Semantic Web and Information Systems (IJSWIS)* 5.3 (2009) pp. 1–22.

[16] Sören Auer, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo, "Introduction to Linked Data and Its Lifecycle on the Web", *Reasoning Web*, 2011 pp. 1–75.

[17] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty, "Building Watson: An Overview of the DeepQA Project", *AI Magazine* 31.3 (2010) pp. 59–79.

[18] Charles Oppenheim and Karen Selby, "Access to information on the World Wide Web for blind and visually impaired people", *Aslib Proceedings* 51.10 (1999) pp. 335–345.

[19] Yi-Fan Yang and Sheue-Ling Hwang, "Specialized Design of Web Search Engine for the Blind People", *Proceedings of the 4th International Conference on Universal Access in Human-computer Interaction: Applications and Services*, Beijing, China: Springer-Verlag, 2007 pp. 997–1005.

[20] Minsuk Lee, James Cimino, Hai Ran Zhu, Carl Sable, Vijay Shanker, John Ely, and Hong Yu, "Beyond information retrievalÑmedical question answering", *AMIA annual symposium proceedings* 2006 (2006) p. 469.

[21] Yuan Ni, Huijia Zhu, Peng Cai, Lei Zhang, Zhaoming Qiu, and Feng Cao, "CliniQA : Highly Reliable Clinical Question Answering System", *Quality of Life through Quality of Information - Proceedings of MIE2012, The XXIVth International Congress of the European Federation for Medical Informatics, Pisa, Italy, August 26-29, 2012*, IOS Press, 2012 pp. 215–219.

[22] Yonggang Cao, Feifan Liu, Pippa Simpson, Lamont D. Antieau, Andrew S. Bennett, James J. Cimino, John W. Ely, and Hong Yu, "AskHERMES: An online question answering system for complex clinical questions", *Journal of Biomedical Informatics* 44.2 (2011) pp. 277–288.

[23] Jens Lehmann, Chris Bizer, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann, “DBpedia - A Crystallization Point for the Web of Data”, *Journal of Web Semantics* 7.3 (2009) pp. 154–165, DOI: `doi:10.1016/j.websem.2009.07.002`, URL: `http://jens-lehmann.org/files/2009/dbpedia_jws.pdf`.

[24] K.I. Goh, M.E. Cusick, D. Valle, B. Childs, M. Vidal, and A.L. Barabási, “Human diseasome: A complex network approach of human diseases”, *Abstract Book of the XXIII IUPAP International Conference on Statistical Physics*, 2007.

[25] David S. Wishart, Craig Knox, Anchi Guo, Savita Shrivastava, Murtaza Hassanali, Paul Stothard, Zhan Chang, and Jennifer Woolsey, “DrugBank: a comprehensive resource for in silico drug discovery and exploration.”, *Nucleic Acids Research* 34.Database-Issue (Feb. 2, 2007).

[26] Giovanni Tummarello, Renaud Delbru, and Eyal Oren, “Sindice.com: weaving the open linked data”, ISWC’07/ASWC’07 (2007).

[27] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker, “Sig.ma: Live views on the Web of Data.”, *J. Web Sem.* 8.4 (2010) pp. 355–364.

[28] Li Ding, Timothy W. Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs, “Swoogle: a search and metadata engine for the semantic web”, *CIKM*, ACM, 2004.

[29] M. D’aquin, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez, and D. Guidi, “Toward a New Generation of Semantic Web Applications”, *Intelligent Systems, IEEE* 23.3 (2008) pp. 20–28.

[30] Vanessa Lopez, Victoria S. Uren, Enrico Motta, and Michele Pasin, “AquaLog: An ontology-driven question answering system for organizational semantic intranets.”, *J. Web Sem.* 5.2 (2007).

[31] Anastasia Karanastasi, Alexandros Zotos, and Stavros Christodoulakis, “The OntoNL Framework for Natural Language Interface Generation and a Domain-Specific Application”, *First International DELOS Conference, Pisa, Italy*, 2007.

[32] Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, and Miroslav Goranov, “KIM - Semantic Annotation Platform”, *Journal of Natural Language Engineering* (2004).

[33] Gideon Zenz, Xuan Zhou, Enrico Minack, Wolf Siberski, and Wolfgang Nejdl, “From Keywords to Semantic Queries – Incremental Query Construction on the Semantic Web”, *Web Semantics* 7.3 (2009) pp. 166–176.

[34] Xiaomin Ning, Hai Jin, Weijia Jia, and Pingpeng Yuan, “Practical and effective IR-style keyword search over semantic web.”, *Inf. Process. Manage.* 45 (2009).

[35] S.M. Beitzel, “On Understanding and Classifying Web Queries”, PhD thesis: Illinois Institute of Technology, 2006.

[36] Fabio Crestani, “Application of Spreading Activation Techniques in Information Retrieval”, *Artif. Intell. Rev.* 11 (1997).

[37] Markus Holi and Eero Hyvönen, “Fuzzy View-Based Semantic Search.”, *ASWC*, vol. 4185, Springer, 2006.

[38] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi de Aragão, “A hybrid approach for searching in the semantic web”, ACM, 2004.

[39] Aidan Hogan, Andreas Harth, JÃ¼rgen Umbrich, Sheila Kinsella, Axel Polleres, and Stefan Decker, “Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine.”, *J. Web Sem.* 9.4 (2011).

[40] Gong Cheng and Yuzhong Qu, “Searching Linked Objects with Falcons: Approach, Implementation and Evaluation.”, *Int. J. Semantic Web Inf. Syst.* 5.3 (Oct. 30, 2009) pp. 49–70.

[41] V. Lopez, Fernandez M., Motta E., and N. Stieler, “PowerAqua: Supporting Users in Querying and Exploring the Semantic Web”, *Journal of Semantic Web*, In press.

[42] Christina Unger and Philipp Cimiano, “Pythia: compositional meaning construction for ontology-based question answering on the semantic web”, *16th Int. Conf. on NLP and IS*, NLDB’11, 2011 pp. 153–160.

[43] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano, “SPARQL Template-Based Question Answering”, *Proceedings of WWW*, 2012.

[44] Daniel Gerber and Axel-Cyrille Ngonga Ngomo, “Bootstrapping the Linked Data Web”, *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*, 2011.

[45] Daniel Gerber and Axel-Cyrille Ngonga Ngomo, “Extracting Multilingual Natural-Language Patterns for RDF Predicates”, *Proceedings of EKAW*, 2012.

[46] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, and Hrishikesh Karambelkar, “Bidirectional Expansion For Keyword Search on Graph Databases”, *VLDB*, 2005.

[47] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, and S. Sudarshan, “Keyword Searching and Browsing in Databases using BANKS”, *ICDE*, 2002.

[48] T. Tran, H. Wang, S. Rudolph, and P. Cimiano, “Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data.”, *ICDE*, 2009.

[49] Yufei Li, Yuan Wang, and Xiaotao Huang, “A Relation-Based Search Engine in Semantic Web.”, *IEEE Trans. Knowl. Data Eng.* (2007).

[50] Guus Schreiber, Alia Amin, Lora Aroyo, Mark van Assem, Victor de Boer, Lynda Hardman, Michiel Hildebrand, Borys Omelayenko, Jacco van Osenbruggen, Anna Tordai, Jan Wielemaker, and Bob Wielinga, “Semantic annotation and search of cultural-heritage collections: The MultimediaN E-Culture demonstrator”, *Journal of Web Semantics* (2008).

[51] Ramanathan V. Guha, Rob McCool, and Eric Miller, “Semantic search”, *WWW*, 2003 pp. 700–709.

[52] Amit Sheth, Boanerges Aleman-Meza, I. Budak Arpinar, Christian Halaschek-Wiener, Cartic Ramakrishnan, Yashodhan Warke Clemens Bertram, David Avant, F. Sena Arpinar, Kemafor Anyanwu, and Krys Kochut,
"Semantic association identification and knowledge discovery for national security applications", *Journal of Database Management* (2005).

[53] Thanh Tran, Tobias Mathäß, and Peter Haase,
"Usability of Keyword-Driven Schema-Agnostic Search.", *ESWC (2)*, vol. 6089, LNCS, Springer, June 8, 2010 pp. 349–364.

[54] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das,
"DBXplorer: A System for Keyword-Based Search over Relational Databases.", *ICDE*, 2002.

[55] Vagelis Hristidis and Yannis Papakonstantinou,
"DISCOVER: Keyword Search in Relational Databases.", *VLDB*, 2002.

[56] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou,
"Efficient IR-Style Keyword Search over Relational Databases.", *VLDB*, 2003.

[57] Fang Liu, Clement T. Yu, Weiyi Meng, and Abdur Chowdhury,
"Effective keyword search in relational databases.", *SIGMOD Conference*, 2006.

[58] Lin Guo, Feng Shao, Chavdar Botev, and Jayavel Shanmugasundaram,
"XRANK: Ranked Keyword Search over XML Documents.", *SIGMOD Conference*, 2003.

[59] Ziyang Liu and Yi Chen,
"Reasoning and identifying relevant matches for XML keyword search.", *PVLDB* (2008).

[60] Liang Jeff Chen and Yannis Papakonstantinou,
"Supporting top-K keyword search in XML databases.", *ICDE*, 2010.

[61] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer,
"Question answering on interlinked data", *WWW*, ed. by Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo A. Baeza-Yates, and Sue B. Moon,
International World Wide Web Conferences Steering Committee / ACM, 2013 pp. 1145–1156, ISBN: 978-1-4503-2035-1.

[62] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano, "Template-based question answering over RDF data", ACM, 2012.

[63] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment", *J. ACM* 46.5 (1999).

[64] Andrew J. Viterbi,
"Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Transactions on Information Theory* IT-13.2 (1967).

[65] E. Vorhees, "The TREC-8 question answering track report", *Proceedings of TREC-8*, 1999.

[66] David R. Cheriton and Robert Endre Tarjan, "Finding Minimum Spanning Trees.", *SIAM J. Comput.* (1976).

[67] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta,
"Evaluating question answering over linked data", *J. Web Sem.* 21 (2013) pp. 3–13.

[68] Esther Kaufmann and Abraham Bernstein,
"How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users?", *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea*, 2008 pp. 281–294.

[69] Monique Reichert, Serge Linckels, Christoph Meinel, and Thomas Engel, "Student's Perception of a Semantic Search Engine", *CELDA*, IADIS, 2005 pp. 139–147.

[70] Haofen Wang, Qiaoling Liu, Thomas Penin, Linyun Fu, Lei Zhang 0007, Thanh Tran, Yong Yu, and Yue Pan, "Semplore: A scalable IR approach to search the Web of Data.", *J. Web Sem.* (2009).

[71] Daniel M. Herzig and Thanh Tran, "Heterogeneous web data search using relevance-based on the fly data integration.", ACM, 2012 pp. 141–150.

[72] A. Uzuner, B. Katz, and D. Yuret, "Word Sense Disambiguation for Information Retrieval.", AAAI Press, 1999.

[73] Lance A. Ramshaw and Mitchell P. Marcus, "Text Chunking using Transformation-Based Learning", *CoRR* (1995).

[74] K. Q. Pu and X. Yu, "Keyword query cleaning.", *PVLDB* 1.1 (Nov. 6, 2008) pp. 909–920.

[75] B. Tan and F. Peng, "Unsupervised query segmentation using generative language models and wikipedia.", *WWW*, ACM, May 13, 2008.

[76] X. Yu and H. Shi, "Query segmentation using conditional random fields.", ACM, Oct. 1, 2009.

[77] J. Guo, G. Xu, X. Cheng, and H. Li, "Named entity recognition in query.", ACM, 2009.

[78] K. M. Risvik, T. Mikolajewski, and P. Boros, "Query Segmentation for Web Search.", 2003.

[79] B. Tan and F. Peng, "Unsupervised query segmentation using generative language models and wikipedia.", ACM, 2008.

[80] D. J. Brenes, D. Gayo-Avello, and R. Garcia, "On the Fly Query Entity Decomposition Using Snippets", *CoRR* abs/1005.5516 (2010).

[81] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke, "Personalized recommendation in social tagging systems using hierarchical clustering", ACM, 2008.

[82] S. Lawrence, "Context in Web Search.", *IEEE Data Eng. Bull.* 23.3 (2000) pp. 25–32.

[83] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing Search in Context: the Concept Revisited", *WWW*, 2001.

[84] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar, "Searching with context", *WWW '06*, ACM, 2006.

[85] R. Guha, Rob McCool, and Eric Miller, "Semantic search", *Proceedings of the 12th international conference on World Wide Web*, WWW '03, Budapest, Hungary: ACM, 2003 pp. 700–709, ISBN: 1-58113-680-3, DOI: `10.1145/775152.775250`, URL: `http://doi.acm.org/10.1145/775152.775250`.

[86] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The Vocabulary Problem in Human-System Communication", *COMMUNICATIONS OF THE ACM* 30.11 (1987) pp. 964–971.

[87] Kevyn Collins-Thompson, "Reducing the risk of query expansion via robust constrained optimization.", *CIKM*, ACM, Nov. 17, 2009.

[88] Houtao Deng, George C. Runger, and Eugene Tuv, "Bias of Importance Measures for Multi-valued Attributes and Solutions.", *ICANN (2)*, vol. 6792, Springer, 2011 pp. 293–300.

[89] Dunja Mladenic, Janez Brank, Marko Grobelnik, and Natasa Milic-Frayling, "Feature selection using linear classifier weights: interaction with classification models.", *In Proceedings of the 27th Annual International ACM SIGIR Conference (SIGIR2004*, ACM, Feb. 10, 2006.

[90] Claudio Carpineto and Giovanni Romano, "A Survey of Automatic Query Expansion in Information Retrieval", *ACM Comput. Surv.* 44.1 (2012) 1:1–1:50, ISSN: 0360-0300, DOI: 10.1145/2071389.2071390, URL: http://doi.acm.org/10.1145/2071389.2071390.

[91] Shoushan Li, Rui Xia, Chengqing Zong, and Chu-Ren Huang, "A framework of feature selection methods for text categorization", *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, Suntec, Singapore: Association for Computational Linguistics, 2009 pp. 692–700, ISBN: 978-1-932432-46-6, URL: http://dl.acm.org/citation.cfm?id=1690219.1690243.

[92] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, "The WEKA data mining software: an update", *SIGKDD Explor. Newsl.* 11.1 (2009) pp. 10–18, ISSN: 1931-0145, DOI: 10.1145/1656274.1656278, URL: http://doi.acm.org/10.1145/1656274.1656278.

[93] Matthias Wendt, Martin Gerlach, and Holger Duwiger, "Linguistic Modeling of Linked Open Data for Question Answering", *Proceedings of Interacting with Linked Data (ILD 2012), workshop co-located with the 9th Extended Semantic Web Conference, May 28, 2012, Heraklion, Greece*, ed. by Christina Unger, Philipp Cimiano, Vanessa Lopez, Enrico Motta, Paul Buitelaar, and Richard Cyganiak, 2012 pp. 75–87, URL: http://ceur-ws.org/Vol-913.

[94] Daniel Gerber and Axel-Cyrille Ngonga Ngomo, "Bootstrapping the Linked Data Web", *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*, 2011.

[95] Omer Gunes, Christian Schallhart, Tim Furche, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo, "EAGER: extending automatically gazetteers for entity recognition", *Proceedings of the 3rd Workshop on the People's Web Meets NLP: Collaboratively Constructed Semantic Resources and their Applications to NLP*, 2012.

[96] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge", *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ACM, 2008 pp. 1247–1250.

[97] Nitish Aggarwal and Paul Buitelaar, "A System Description of Natural Language Query over DBpedia", *Proceedings of Interacting with Linked Data (ILD 2012), workshop co-located with the 9th Extended Semantic Web Conference, May 28, 2012, Heraklion, Greece*, ed. by Christina Unger,

Philipp Cimiano, Vanessa Lopez, Enrico Motta, Paul Buitelaar, and Richard Cyganiak, 2012 pp. 97–100, URL: `http://ceur-ws.org/Vol-913`.

[98] Elena Cabrio, Alessio Palmero Aprosio, Julien Cojan, Bernardo Magnini, Fabien Gandon, and Alberto Lavelli, "QAKiS @ QALD-2", *Proceedings of Interacting with Linked Data (ILD 2012), workshop co-located with the 9th Extended Semantic Web Conference, May 28, 2012, Heraklion, Greece*, ed. by Christina Unger, Philipp Cimiano, Vanessa Lopez, Enrico Motta, Paul Buitelaar, and Richard Cyganiak, 2012 pp. 88–96, URL: `http://ceur-ws.org/Vol-913`.

[99] ziqi zhang, Anna Lisa Gentile, Isabelle Augenstein, Eva Blomqvist, and Fabio Ciravegna, "Mining Equivalent Relations from Linked Data", *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria, 2013.

[100] Isabelle Augenstein, Anna Lisa Gentile, Barry Norton, Ziqi Zhang, and Fabio Ciravegna, "Mapping Keywords to Linked Data Resources for Automatic Query Expansion", *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, Lecture Notes in Computer Science, Springer, 2013.

[101] Christiane Fellbaum, ed., *WordNet: an electronic lexical database*, MIT Press, 1998.

[102] Michael E. Lesk, "Word-word associations in document retrieval systems", *American Documentation* 20.1 (1969) pp. 27–38.

[103] C. J. van Rijsbergen, *Information Retrieval*, Buttersworth, London, 1989.

[104] Cornelis Joost van Rijsbergen, *The geometry of information retrieval.* Cambridge University Press, 2004 pp. I–XII, 1–150.

[105] Saeedeh Shekarpour, Konrad Höffner, Jens Lehmann, and Sören Auer, "Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features", *7th IEEE International Conference on Semantic Computing, September 16-18, 2013, Irvine, California, USA*, 2013.

Saeedeh Shekarpour

Curriculum Vitae

ACADEMIC EDUCATION

2010-2014	**PhD Candidate in Computer Science**, *Institute for Applied Computer Science at Bonn Universität*, AKSW research group, Institute of Computer Science(IfI), Leipzig Universität, Germany.
2005-2008	**M.Sc. in Computer Science**, *Department of Computer Science and Engineering, School of Engineering, Shiraz University*, Shiraz, Iran.
2000-2004	**B.Sc. in Computer Engineering(Software)**, *Department of Electronic and Computer Engineering, Shahid Beheshti University*, Tehran, Iran.

Honors

1998	**Semifinalist of National Computer Olympiad**, *Iran.*
2000	**B.Sc. Admission**, *Ranked within top 0.3 of the nationwide university entrance examination participants*, Iran.
2010	**PhD Admission at Tehran University**, *Tehran, Iran.*
2011	**PhD Grant from DAAD**, *(German Academic Exchange Service).*

Presentations and Talks

WIIA 2011	**Web Intelligence Conference**, *Lyon, France*, August 2011.
ISWC 2011	**International Semantic Web Conference**, *Bonn, Germany*, October 2011.

+49 17684419554 • ✉ *shekarpour@uni-bonn.de*
http://eis.iai.uni-bonn.de/SaeedehShekarpour.html

JIST 2012	**Joint International Semantic Technology Conference**, *Nara, Japan*, December 2012.
WWW 2013	**World Wide Web Conference**, *Rio de Janeiro, Brazil*, May 2013.
IBM Talk 2013	**Presenting my work in IBM Research Center (Watson project-DeepQA)**, *New York, USA*, December 2013.

Summer Schools

ESSIR 2011	**Information Retrieval Summer School**, *Koblenz, Germany*, August 2011.
ISSLOD 2011	**IndianSummer School on Linking Open Data**, *Leipzig, Germany*, September 2011.
Web Science 2012	**Web Science Summer School**, *Leiden, Netherland*, July 2012.

Language

Persian	**Native**.
English	**Fluent**.
German	**Falimiar**.
Arabic	**Familiar**.

Research Interest

1. Semantic Web.
2. Semantic Search.
3. Linked Data.
4. Information Retrieval.
5. Web Mining.
6. Natural Language Processing.

+49 17684419554 • *shekarpour@uni-bonn.de*
http://eis.iai.uni-bonn.de/SaeedehShekarpour.html

Community Service (Selection)

2010-2014 **Reviewing**.
Semantic Web Journal, Journal of Web Engineering, International Journal on Semantic Web and Information Systems, Journal of Computational Intelligence, Language Resource and Evaluation Conference (LREC), The International Conference on Building and Exploring Web Based Environments,

2014 **Program Committee**.
Clef Labs

Skills

Programming Languages	**JAVA, C#, C,C++, MATLAB**.
Database Management	**Virtuoso, SQL-Server**.
Experiences	**Latex, Linux, Office**.

References

1. Name **Prof. Dr. Sören Auer**.

   Affiliation *, The department Enterprise Information Systems (EIS) at the Institute for Applied Computer Science at University of Bonn and Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Bonn, Germany.*

   E-mail **auer@cs.unibonn.de**.

   Homepage **http://eis.iai.unibonn.de**.

   Mobile **+4915784988949**.

2. Name **Dr. Axel C. Ngonga Ngomo**.

   Affiliation *, AKSW Research Group, Business Information Systems(BIS) of the Institute of Computer Science(IfI) , University of Leipzig and Institute for Applied Informatics (InfAI).*

   E-mail **ngonga@informatik.unileipzig.de**.

+49 17684419554 • *shekarpour@uni-bonn.de*
http://eis.iai.uni-bonn.de/SaeedehShekarpour.html

	Homepage	**http://aksw.org/AxelNgonga.html**.
	Mobile	**+4917623517631**.
3.	Name	**Dr. Jens Lehmann**.
	Affiliation	, *AKSW Research Group, Business Information Systems(BIS) of the Institute of Computer Science(IfI) , University of Leipzig and Institute for Applied Informatics (InfAI).*
	E-mail	**Lehmann@informatik.unileipzig.de**.
	Homepage	**http://aksw.org/JensLehmann.html**.
	Mobile	**+4917670350423**.
4.	Name	**Prof. Dr. S. D. Katebi**.
	Affiliation	, *Department of Computer Science and Engineering, School of Engineering, Shiraz University.*
	E-mail	**katebi@shirazu.ac.ir**.
	Homepage	**http://www.cse.shirazu.ac.ir/ katebi**.

PhD Thesis

Title	**Semantic Interpretation of User Queries for Question Answering on Interlinked Data**.
Supervisor	**Prof. Dr. Sören Auer**.
Demo	**http://sina.aksw.org**, *runs on DBpedia.*
Demo	**http://sinalinkeddata.aksw.org**, *runs on three interlinked lifescience datasets, i.e. Drugbank, Diseasome and Sider.*
Demo	**http://lod-query.aksw.org**, *runs on DBpedia.*

Abstract , *Developing a question answering over these interlinked data sources is still challenging. In this respect, several challenges such as resource disambiguation, vocabulary mismatch, inference, link traversal are raised. The contributions of this work are as follows: (1) A novel approach for determining the most suitable resources for a user-supplied query from different datasets (disambiguation approach). We employed a Hidden Markov Model, whose parameters were bootstrapped with different distribution functions. (2) A novel method for constructing federated formal queries using the disambiguated resources and leveraging the linking structure of the underlying datasets. This approach essentially relies on a combination of domain and range inference as well as a link traversal method for constructing a connected graph, which ultimately renders a corresponding SPARQL query. (3) Regarding the problem of vocabulary mismatch, our contribution is divided into two parts, First, we introduce a number of new query expansion features based on semantic and linguistic inferencing over Linked Data. We evaluate the effectiveness of each feature individually as well as their combinations, employing Support Vector Machines and Decision Trees. Second, we propose a novel method for automatic query expansion, which employs a Hidden Markov Model to obtain the optimal tuples of derived words. (4) We provide two benchmarks for two different tasks to the community of question answering systems. The first one is used for the task of question answering on interlinked datasets (i.e. federated queries over Linked Data). The second one is used for the vocabulary mismatch task. We evaluate the accuracy of our approach using measures like MRR, precision, recall, and F-measure on three interlinked life-science datasets as well as DBpedia. The results of our accuracy evaluation demonstrate the effectiveness of our approach. Moreover, we study the runtime of our approach in its sequential as well as parallel implementations.*

+49 17684419554 • ✉ *shekarpour@uni-bonn.de*
http://eis.iai.uni-bonn.de/SaeedehShekarpour.html

Master Thesis

Title	**Trust Modeling for Semantic Web**.
Supervisor	**Prof. Dr. S. D. Katebi**.
Grade	**Thesis was evaluated as Excellent with the grade 19.95 out of 20**.
Support	**Thesis was supported financially by Iran Telecommunication Research Center (IRTC)**.
Abstract	, *The aim of this work is two folds. Firstly, some of the well known methods of trust modeling and trust evaluation that relates mainly to the semantic web structure are reviewed and analyzed. A categorization for calculation of trust and an analytical view of possible models of trust rating through a chain of acquaintances are presented. Based on experimental results the well known methods are compared and contrasted. Secondly a new method for evaluating trust is also proposed. This new model has the advantages of simplicity in calculation and enhanced accuracy. The method is associated with two algorithms, an algorithm for propagation and another for aggregation. The propagation algorithm utilizes statistical techniques and the aggregation algorithm is based on a weighting mechanism. The technique is named Maxweight method and is also implemented and the results are compared based on a designed accuracy metric. The proposed method may be employed as a subsystem for trust management in semantic web and trust evaluation in human interaction in a social networks as well as machines (artificial agents). Experimental results illustrate the efficiency and effectiveness of the proposed method.*

Journal Publications, peer-reviewed

- **In Press:** Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data", *Journal of Web Semantics Science, Services and Agents on the World Wide Web*, 2014.
- **Published:** Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler, "Generating SPARQL

queries Using Templates", *Web Intelligence and Agent Systems Journal* 11.3 (2013) pp. 283–295.

- **Published:** Sebastian Tramp, Philipp Frischmuth, Timofey Ermilov, Saeedeh Shekarpour, and Sören Auer, "An architecture of a distributed semantic social network", *Semantic Web* 5.1 (2014) pp. 77–95.
- **Published:** Edgard Marx, Tommaso Soru, Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, and Karin Breitman, "Towards an Efficient RDF Dataset Slicing", *Int. J. Semantic Computing* 7.4 (2013) p. 455.
- **Published:** Saeedeh Shekarpour and S. D. Katebi, "Modeling and evaluation of trust with an extension in semantic web", *Journal of Web Semantics Science, Services and Agents on the World Wide Web* 8.1 (2010) pp. 26–36.
- **Published:** Saeedeh Shekarpour and S. D. Katebi, "A Trust Model For Semantic Web", *International Journal of Simulation Systems, Science and Technology* 10.2 (2010).

Conference Publications, peer-reviewed

- **Published:** Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "Question Answering on Interlinked Data", *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, 2013 pp. 1145–1156.
- **Published:** Saeedeh Shekarpour, Konrad Höffner, Jens Lehmann, and Sören Auer, "Keyword Query Expansion on Linked Data Using Linguistic and Semantic Features", *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, 2013 pp. 191–197.
- **Published:** Edgard Marx, Saeedeh Shekarpour, Sören Auer, and Axel-Cyrille Ngonga Ngomo, "Large-scale RDF Dataset Slicing", *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, 2013.
- **Published:** Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, "Keyword-Driven Resource Disambiguation over RDF Knowledge Bases", *Semantic Technology, Second Joint International Conference, JIST 2012, Nara, Japan, December 2-4, 2012. Proceedings*, Springer, 2012 pp. 159–174.
- **Published:** Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler, "Keyword-Driven SPARQL Query Generation Leveraging Background Knowledge", *Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2011, Campus Scientifique de la Doua, Lyon, France, August 22-27, 2011*, 2011 pp. 203–210.
- **Published:** Saeedeh Shekarpour and S. D. Katebi, "A Trust Model Based on Statistical Propagation and Fuzzy Aggregation for Semantic Web", *EMS 2008, Second UKSIM European Symposium on Computer Modeling and Simulation,*

Liverpool, England, UK, 8-10 September 2008, 2008 pp. 459–464.
- **Published:** Saeedeh Shekarpour, Hassan Abolhasani, and S. D. Katebi, “Web Structure Mining Techniques”, *IDMC 2007, First Data Mining Conference, Amirkabir University, Tehran , Iran*, 2007.
- **Submitted:** Saeedeh Shekarpour and Sören Auer, “Query Reformulation on RDF Knowledge Bases using Hidden Markov Models”, *Submitted to the Eighth International Conference on Web Search and Web Data Mining, WSDM 2015*, 2015.

Workshop and Doctoral Consutiom Publications, peer-reviewed

- **Published:** Saeedeh Shekarpour, “DC Proposal: Automatically Transforming Keyword Queries to SPARQL on Large-Scale Knowledge Bases”, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part II*, Springer, 2011 pp. 357–364.
- **Published:** Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer, “Query Segmentation and Resource Disambiguation Leveraging Background Knowledge”, *Proceedings of WoLE Workshop*, 2012.

+49 17684419554 • ✉ *shekarpour@uni-bonn.de*
http://eis.iai.uni-bonn.de/SaeedehShekarpour.html