Technical University
Sofia

Department of German Engineering and Industrial Management

Integration of Semantic Technologies in the Processing of News

Tsvetan Dimitrov

Supervisor: Assoc. Prof. Adelina Aleksieva

A thesis presented for the degree of

Bachelor of Informatics

Sofia, Bulgaria

July 15, 2016

Abstract

The purpose of this thesis is to develop analytics methods for and visualisations of the news data from a public Ontotext service called News On the Web (NOW) using semantic technologies, standards of the Semantic Web and Linked Open Data. More specific tasks are generating a "News Map of Today's news" and recommending linked news and objects based on analyzing "hidden champions". Hidden champions are concepts which are not directly mentioned in the news, but are linked in the graph with a few concepts which are. For this purpose a cognitive technique called "priming" is used in which the popularity of the mentioned concepts is being spread out across the knowledge graph, following the principle of spreading activation in neural networks. The News Map will be visualised in the form of a word cloud in various options: direct popularity of the concepts based on mentions count in the news and degree of importance of these news, relative popularity - where the popularity within the current day is normalized against the popularity within the last year, concepts that are "hidden champions" and a combination of "hidden champions" and relative popularity. Analogous maps will be created for a separate news article as a way to visualise the important subjects in it.

Acknowledgments

I would like to thank my supervisor Assoc. Prof. Adelina Aleksieva for the guidance throughout the process of writing this thesis and the valuable feedback I received from her. I would also like to thank my other supervisor from Ontotext Milena Yankova, Ph.D and the CEO of Ontotext Atanas Kiryakov for the creative ideas and also their help in the data set work of cleaning and mapping to other Linked Open Data data sets. And last but not least I am very grateful to all my colleagues from Ontotext for their year long work and expertise in the areas of semantic technologies and text mining with whom without this thesis would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Semantic Web is a network of structured data represented in a format that enables automatic interpretation of its meaning. Data is represented in the form of graphs and its meaning is defined in ontologies - semantic schemes enabling automatic interpretation by deductive logical conclusion. After 2006 the Semantic Web is realized in the form of Linked Open Data - a network of data, published on different servers, but interlinked with hyperlinks (same as the hyperlinks in Web 1.0). Today there are thousands of databases (knowledge bases) published and interlinked in this way, that contain more that a trillion facts in all areas of life. Linked Data is accepted as a standard by the governments of the United States, United Kingdom and other countries for publishing public data - for example statistical information, company registers etc. Amongst the most popular databases are DBPedia (a structured version of Wikipedia) and GeoNames (containing most of Earth's geographical information).

Open Linked Data has many purposes. One of them is presenting knowledge as graphs (e.g. Google Knowledge Graph) when analyzing and indexing natural language in a process called "semantic annotation". Text is being analyzed with the purpose of finding mentions of people, companies, places and other concepts - known or unknown. Ontotext offers a solution of this kind called "Dynamic Semantic Publishing (DSP)", which persists the text, enriched with semantic tags, alongside with the knowledge graphs in a semantic graph database. Using these technologies a demo platform called "News On the Web (NOW)" has been developed, which offers

browsing news, data linked with them or other news. NOW operates on a stream of about 10 000 news a month which it analyzes and links with a knowledge graph that includes DBPedia and GeoNames.

1.1 Limitations of current search technologies

Searching information on the World Wide Web is an idea dating from its very beginning. Although based on the same technologies as general information retrieval, the web brings additional challenges such as how to scale efficiently and how to implement correct ranking in its ever changing environment. Different search engines return different search results due to the variation in indexing and search processing. Google, Yahoo and Bing handle queries after processing keywords. They only search recent information given on a web page. They are constantly improving and delivering more and more accurate results, but at the end they cannot answer you a question with an aggregated result and cannot suggest you search queries which are semantically interlinked based on concepts and not only on stems and other similar keywords.

The current web is the biggest global database that lacks the existence of a semantic structure and hence makes it difficult for a machine to understand the information provided by the user. This leads to formulating the two biggest challenges in web search:

- How can a search engine map a query to a document and retrieve meaningful information from it?
- The query results produced by search engines are distributed across different documents that may be connected with hyperlinks. How does the search engine efficiently recognize such distributed results?

The Semantic Web could solve the first problem by using semantic annotations to produce structured by meaning information. The second problem could be addressed by having graph-based query models. Such a goal poses challenges of its own in the areas of knowledge representation, natural language processing and graph databases.

1.2 Why do we need more insight?

In today's world everything is information. The average connected consumer has access to an insane amount of it, thanks to the ubiquity of smartphone access to the web. From checking restaurant reviews and stock prices, to taking pictures of a new pair of jeans and asking the opinion of friends on Facebook, today's consumer is no longer restricted to choosing a brand through a push marketing approach. This change of direction in the purchase cycle has resulted in brands trying to better know the consumer in order to present him with the appropriate content for the purpose of advertising or just offering better customer experience. Although today buzzwords such as "Big Data" seem to be very trendy and loved by marketing people, the reality is that we need better insights, not more data. The data is out there and there is plenty of it, easily accessible through the web. The biggest problem is that it is hard to reason about in order to gain significant benefits. The grand vision of the Semantic Web wants to accomplish exactly that:

- Enrich open data with its own semantics.
- Interlink those semantics between different entities and facts.
- Set a standardized way to access data, its semantics and other links leading to other related pieces of data.

When such knowledge is freely and easily available, a lot of cool applications become possible, e.g. content and user based recommendation systems become way more accurate if they can take advantage of user activity data and content metadata being in a unified data model and interlinked one another to find interesting patterns of user preferences towards particular content. Search becomes smarter because it indexes contextual information, rather than only keywords, and can infer related concepts. This could be extremely helpful when dealing with, e.g. news data (relating people, organisations, locations, etc.) or life sciences data (relating drugs, diseases, treatments, symptoms, etc.). To even further boost the benefits of such enriched data, machine learning techniques could be applied to train models on top of core data

semantics and relations in order to predict behaviour even more accurate than when only using raw data and manually choosing and structuring its features to train upon.

Getting more insight about the world around us and the information we produce every single day plays a huge role in our modern lives and will continue to be even more important in the future.

1.3 Thesis Overview

The aim of this thesis is to present a practical example of an intelligent application for understanding news without reading the articles themselves, based upon the Semantic Web technology stack, and to demonstrate how unstructured content could be processed into something that enables automatic reasoning of its meaning. News are a great example of a raw data source which is rich on facts and knowledge, concerning most areas of life, scattered in free text. The solution to this problem will be a pipeline which will extract the important information from free text, convert it to an RDF (Resource Description Framework) format, interlink it with other Linked Open Data data sets, import the whole data into a semantic database, called a triplestore, run interesting SPARQL queries against it to pick up trends in popularity of certain entities and finally visualise those trends and entities into a word cloud form, based on different popularity criteria and a geographic heat map of news mentioning countries all around the world.

Chapter 2

Background

2.1 The World Wide Web

The World Wide Web is the greatest repository of information ever assembled by man. It contains knowledge in the form of documents and multimedia resources concerning almost every area or subject regarding our lives. All this data is freely accessible to anyone with an Internet connection. Its success is largely due to its decentralized design and nature. Web pages are hosted by different computers and each document can point to other documents independent of where the other document is physically located. As a result everybody all over the world can publish content on the web, allowing it to grow exponentially as more and more people learn how to use it.

However the huge scale of the web leads to some consequences. Due to the sheer volume of available information, it is becoming increasingly difficult to locate useful information. Although search engines, e.g Google, Yahoo, Bing, can provide some assistance, they are far from perfect. For many users, locating the "right" document is still like trying to find a needle in a haystack.

2.1.1 Development of the Web

In 1990 Tim Berners-Lee developed the first version of his World Wide Web program at CERN. The concept behind his invention was to use hypertext as a means of

organizing a distributed document system. The term *hypertext* refers to a collection of documents with references to each other (links) that enable readers to navigate from one document to the other without following a sequential pattern. In order to make it work on the Internet, Berners-Lee had to develop three crucial parts of the whole system:

1. Mechanism for addressing documents on different machines.

2. Protocol that allowed computers to request documents.

3. Simple language to describe the documents.

Before inventing the World Wide Web there were more powerful hypertext systems available, but he built his according to simple specifications that he later published as standards. That enabled other people to use his system in their own web servers, web browsers and especially integrate them in their own websites. In time his system proved to be the most popular and most used in the world and that is why he founded W3C (World Wide Web Consortium) in order to oversee these standards and evolve them in the future.

2.1.2 Enabling Technologies

HTTP

Hypertext Transfer Protocol (HTTP) is a method for encoding and transporting information between a client (e.g. a web browser) and a web server. HTTP is the primary protocol for transmission of information across the Internet. Information is exchanged between clients and servers in the form of hypertext documents, from which HTTP gets its name. Hypertext is structured text that uses logical links (hyperlinks), between nodes containing text. Hypertext documents can be manipulated using the Hypertext Markup Language (HTML). Using HTTP and HTML clients can request different kinds of content (such as text, images, video, and application data) from web and application servers that host the content.

HTTP follows a request-response paradigm in which the client makes a request and the server issues a response that includes not only the requested content, but also relevant status information about the request in the form of *headers*. This self-contained design allows for the distributed nature of the Internet, where a request or response might pass through many intermediate routers and proxy servers. It also allows intermediary servers to perform value-added functions such as load balancing, caching, encryption, and compression. HTTP resources such as web servers are identified across the Internet using unique identifiers known as Uniform Resource Locators (URLs). HTTP is an application layer protocol and relies on an underlying network-level protocol such as Transmission Control Protocol (TCP) to function.

HTML

HTML is a markup language designed for the web as a way to structure documents semantically. Web browsers can read HTML files and render them into visible or audible web pages. Internally HTML describes a hierarchical structure depicted from the ability to nest tags within each other. This structure is called the Document Object Model (DOM) and plays an important role when manipulating HTML with Javascript for achieving web page interactivity. HTML5 is the newest standard which adds many new features to the language such as native support for including and handling multimedia and graphical content - the new `<video>`, `<audio>` and `<canvas>` elements, support for Scalable Vector Graphics (SVG) content and MathML for mathematical formulas. Another neat feature are semantic tags used to enrich the semantic content of documents, new page structure elements such as `<main>`, `<section>`, `<article>`, `<header>`, `<footer>`, `<aside>`, `<nav>` and `<figure>` are added.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. CSS is designed primarily to enable the separation of document content from document presentation, including

aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Javascript

Javascript is a dynamic programming language originally designed for the web and standardized in the *ECMAScript* language specification. It is a multi-paradigm language, supporting imperative, object-oriented and functional programming. Functions are treated as first class citizens. A lot of language constructs are inspired by languages such as Java and Scheme. Although meant to live in the browser, Javascript has proven over time to be suitable as a general-purpose programming language. NodeJS is an example of a runtime environment allowing to use Javascript for server side development, MongoDB accepts queries written in Javascript, Adobe's Acrobat and Adobe Reader support JavaScript in PDF files etc..

2.2 Knowledge Representation

Knowledge Representation is a field of artificial intelligence that deals with the symbolic representation of knowledge of a subject area. This is accomplished in an automatic way by using reasoning programs. More informally, it is a part of AI, which is concerned with thinking and how thinking leads to intelligent behavior. As a field of study, it proposes an approach to understanding intelligent behavior that is radically different from the others. Instead of studying people or animals (their biology, their nervous system, their psychology, their sociology, their development, etc.), it is the people's knowledge that is to be studied. It is accepted that people possess the right knowledge and this knowledge can apply in different situations in order to achieve their goals. Therefore, in the range of KR you deal with knowledge and not with the owner of knowledge.

The goal is the development of a formalism by which knowledge about the world can be described in an abstract way and can be effectively used to implement intelligent applications. The nature of knowledge is a difficult (philosophical) question. KR is only limited to conceptual knowledge. Other types of knowledge are temporal knowledge, spatial knowledge, procedural knowledge, knowledge of knowledge, etc. KR makes a description of the concept of a lecture, a computer, an illness, a workpiece, etc. - a "what is a XYZ" description.

2.2.1 Logic in Knowledge Representation

The main goal of logic (there is no uniform study of logic but many) is to express knowledge about a certain phenomena or a specific part of the world by means of a formal language. The core that defines reasoning about real world things is their encoding by using a precise set of deterministic rules called *inference rules*, that everyone agrees upon. Correct reasoning enables particular knowledge to be represented by a logical sequence from a set of facts. In logic correct reasoning chains are constructed by chaining applications of simple inference rules that transform the original knowledge in a derived conclusion.

First Order Logic

First Order Logic is a formalism by which one can describe statements very expressively. Just like prepositional logic, first order logic has:

- a syntax that determines which strings are its valid formulas.
- semantics that determines the meaning of each of these formulas.

FOL deals with objects (for example, the residents of Frankfurt and their relationships or the natural numbers and their addition and multiplication) and statements about their properties. In contrast, the propositional logic deals not with objects but only with "true" and "false" statements and their combination. The world is modeled on terms of:

- *Objects* - things with individial identities.
- *Features* - characteristics of objects that distinguish an object from other objects.
- *Relations* - connections between sets of objects.
- *Functions* - a subset of relations, which only take one value as input.

FOL overtakes, modifies and extends the syntax of prepositional logic:

- Similarities:
  - Operators $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.
- Differences:
  - Variables do not correspond to true oder false expressions, but to elements of the universe of a σ-structure.
  - Variables are not atomic formulas any more.
- What is new:
  - Quantifiers: $\exists$ ("does exist") and $\forall$ ("for all").
  - There are symbols for elements from the σ signature.

Description Logic

Description logic is a family of knowledge representation formalisms, which allow the key terms of a field to be described in a formal, logic-based language. Such types of logic are used in various applications, but especially for the semantic annotation of data in data integration and the World Wide Web. The familiar Web Ontology Language (OWL) is substantially based on a description logic. Syntax and semantics are always well-defined logic-based:

- *Syntax:* recursive definition in which the constructors that can be used to form concept terms are stated. Some constructors are related to logical constructors in first-order logic (FOL) such as *intersection* or *conjunction* of concepts, *union* or *disjunction* of concepts, *negation* or *complement* of concepts, *universal restriction* and *existential restriction*. Other constructors have no corresponding construction in FOL including restrictions on roles for example, *inverse*, *transitivity* and *functionality*.

- *Semantics:* strengthens the meaning of representing knowledge in a precise, unambiguous way.

  - *Declarative semantics:* independent from the processing, e.g. you want to represent (describe), not to program in something like Prolog. It enables widespread application independence and is based on logical structures.

Central elements of descriptive logic are:

- *Concepts* - describe classes from objects.

  - person, course, university, table, student, etc.

  Can be described by using logical expressions (formulas).

- *Roles* - binary relations between objects.

  - listen, teach, partOf, etc.

  In most cases can be described by simple expressions.

- TBoxes (assertion of concepts) - define and connect concepts.

  - *Definition of concept*
    student $\equiv$ person $\sqcap\exists$ listens.lecture

  - *General background knowledge / Constraint*
    student $\sqcap$ lecture hall $\sqsubseteq \bot$

- *ABoxes (assertions of individuals)* - describe individuals (objects) and their features.

  - student(hans)
    lecture $\sqcap\exists$ subject.informaticsSubject(blv)
    listens(hans, blv)

2.2.2 Semantic Networks and Frame Systems

One of the oldest knowledge representation formalisms are semantic networks. Each concept in a semantic net is represented by a node in a graph. Concepts that are semantically related, are connected by arcs, which could be optionally labeled. Implication of meaning is represented by the way a concept is connected to other concepts. Special arcs are used to represent abstraction, although their semantics are often unclear. Now it is rather common to use two arcs for this purpose. An *is-a* arc indicates that one concept is a subclass of another, while an *instance-of* arc indicates that a concept is an example of another concept (see *Figure 2-1*). These types of arcs have correlations in basic set theory: *is-a* is like the subset relation and *instance-of* is like the element of that relation. A collection of *is-a* arcs specifies a partial order on classes often referred to as a *taxonomy* or *categorization hierarchy*. The taxonomy can be used to adjust the abstraction level - either generalize a concept to a more abstract class or specialize a class to its more specific concepts.

In the 1970's Marvin Minsky introduces frame systems. In the terminology of such systems, a frame is a named data object that has a set of slots, where each slot represents a property or attribute of the object. Slots can have one or more values (called fillers), some of which may be pointers to other frames. Since each frame has a set of slots that represent its properties, frame systems are usually considered to be more structured than semantic networks. However, it has been shown that frame systems are similar to semantic networks. Famous KR languages based on frame systems are KRL and KL-ONE (based on description logic).

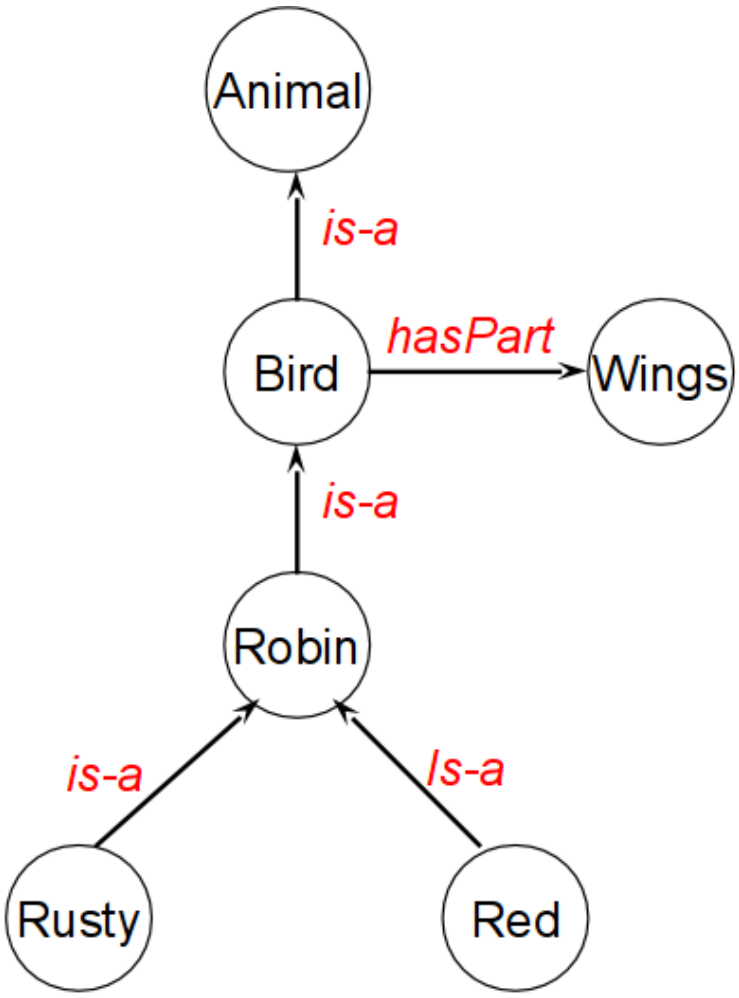

Figure 2-1: Semantic net inheritance

2.2.3 Ontologies

In order for information from different sources to be integrated, there needs to be a shared understanding of the relevant domain. Knowledge representation formalisms provide structures for organizing this knowledge, but provide no mechanisms for sharing it. Ontologies provide a common vocabulary to support the sharing and reuse of knowledge. Although there is no universally accepted definition, the main thread of ontology in the philosophical sense is the study of entities and their relationships. The question an ontology asks is: *What kinds of things exist or can exist in the world, and what manner of relations can those things have to each other?*. On the Semantic Web ontologies define the concepts and relationships (also referred to as terms) used

to describe and represent an area of concern. They are used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms. An example is to use ontologies to organize knowledge. Libraries, museums, newspapers, government portals, enterprises can now use ontologies for modelling their knowledge domain by using standard formalisms which aid in sharing (data integration) and leveraging the power of other interlinked open data.

2.3 Semantic Web

The Semantic Web reveals the next generation in the development of the Internet. It enables interconnectedness of data from one source to another and the representation of this data in a machine readable form. This way computers can understand the information and can process ever more demanding tasks. The Semantic Web consists of a stack of standards and best practices for sharing data and its semantics across the Internet in order to be used by applications. Heavily backed up by the W3C the Semantic Web is also built using the same type of standards such as, e.g. HTML and CSS. These standards are: the RDF data model, the SPARQL query language, the RDFS and OWL standards for vocabularies and ontologies. A product could be created with built-in semantics but if it does not adhere to these standards it cannot be part of the Semantic Web.

2.3.1 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is the result of the work of the W3 Consortium for a web infrastructure for metadata. It is standardized on 22.February 1999. It represents a data model with well-defined formal semantics, based on directed graphs and modelling data as a set of triples *Triple (subject, predicate, object).* A set of triples builds a graph.

A *resource* is something that is uniquely identified and something could be expressed about it. *Subjects* und *predicates* are always resources and an *object* could

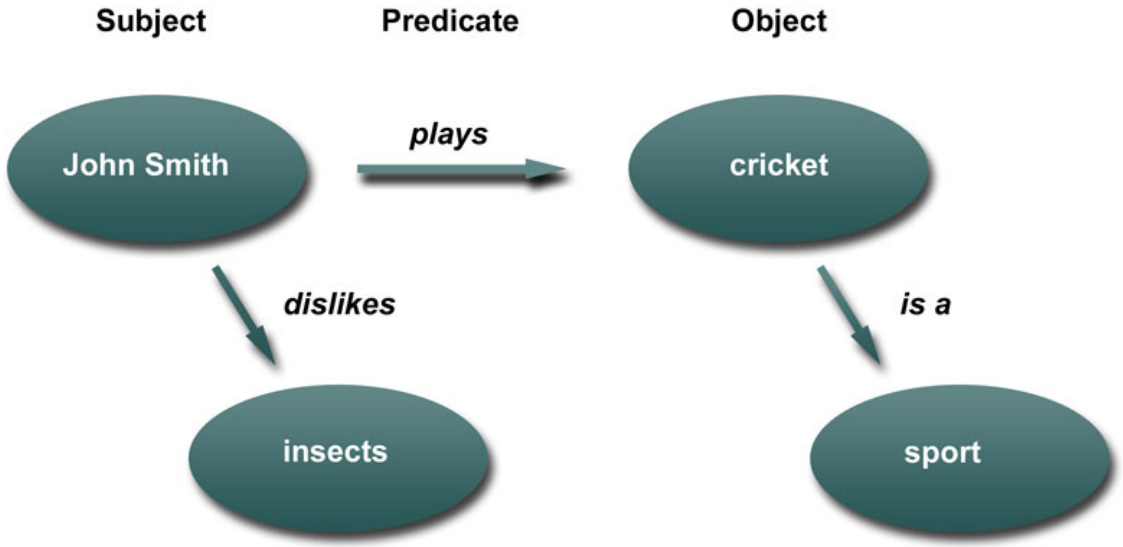

Figure 2-2: RDF Triple Scheme

either be a resource or a *literal*. Literals could be strings or other datatypes (numbers, dates, etc.).

A crucial difference in comparison to other data models such as the relational data model is that RDF represents information as a graph and all objects are connected by edges. For example the sentence "Dresden is a city in Sachsen." partitions itself as "Dresden" and "Sachsen" being objects that are connected with the edge "is a city in". In RDF the following triple is constructed: *subject (Dresden)*, *predicate (is a city in)* and the *object (Sachsen)*. In its essence, this is the whole philosophy. So why bother having this model? HTML describes the structure of a web page in a standardized fashion. This leads to strong decoupling from the used browsers when consuming web page content. The answer is that the same has to be achieved by RDF, but for data.

Data from the Web today is being provided through various custom formats and interfaces. If you want to aggregate it to find it new applications, these interfaces must be individually connected and brought up in a new individual format. In addition today lots of data is listed directly in web pages, e.g. in HTML. In order to utilize this data, it must be extracted individually from the websites. This means for a very high

expenditure and also restricts the actual availability of data and sometimes associates with poorer quality. All of this changes when RDF is used because it eliminates the transformation of data since it is provided and used in the same exact model.

In addition individual graphs could be connected with new links, something understood as the *"Linked Data"* concept which is all about connecting information on a topic so that it could be structurally navigated.

RDF is a data model which could be represented in different formats like e.g. XML (RDF/XML), HTML (RDFa), JSON (JSON-LD) or Turtle (a lighter more human readable format). RDF itself is schema-less, but there do exist different schemes like OWL (Web Ontology Lanquage) and RDFS (RDF Schema). There is also SPARQL (SPARQL Query Language) which is a query language like SQL but for RDF.

2.3.2 Web Ontology Language(OWL)

The OWL (Web Ontology Language) is designed to enable processing of the information content of applications instead of only presenting it to the user. OWL facilitates additional vocabulary in conjunction with formal semantics and allows stronger interpretations of web content than XML, RDF and RDFS. It consists of three languages with increasing expressiveness : OWL Lite, OWL DL and OWL Full. It differs *classes, properties and instances* and works according to the open world assumption which allows inferring new facts related to a class, even though previously you had none for this class. OWL also consists of a set of axioms which describe the classes, their properties and the relationships between them.

2.3.3 SPARQL Query Language (SPARQL)

SPARQL is a query language, i.e. a semantic query language for databases, storing and manipulating RDF. It is a standard which has been created by the RDF Data Access Working Group (DAWG) of the W3C and is recognized as one of the key technologies of the Semantic Web. Although SPARQL queries RDF, it is not only limited to data in the RDF formats. Commercial and open source utilities are avail-

able for handling relational data, XML, spreadsheets and other formats such as RDF, so that you can write SPARQL queries against each of these sources or a combination of them. The "protocol" part of the SPARQL standard sets rules for clients and SPARQL processing servers on how to exchange results between each other. Those rules are specified in a document separate from the query language specification and it usually concerns only developers of SPARQL-processors.

The query in the next example finds the names of all African capitals and the countries in which they are located.

```
PREFIX abc: <http://example.com/exampleOntology#>

SELECT ?capital ?country
WHERE {
    ?x abc:cityname ?capital ;
       abc:isCapitalOf ?y .
    ?y abc:countryname ?country ;
       abc:isInContinent abc:Africa .
}
```

Variables are prefixed with a "?" (a possible alternative is also "$"). As a result the query above returns all variable assignments for "?capital" and "?country", that match the four RDF-triple patterns. Writing out the whole URIs each and every time ruins the readability of the query so that is why prefixes are used. In this case "abc:" stands for `"http://example.com/exampleOntology"`.

2.3.4 Triplestores (semantic databases)

When you want to save lots of triples, persisting them in a huge file as Turtle or RDF/XML is probably not the brightest idea, because at this scale you will need a system that indexes your data for fast retrieval when searching and also a system that decides which parts of your data to keep in memory for optimal performance without swapping to disk. This is exactly the job of a DBMS (Database Management

System), e.g. MySQL or Oracle, but one that is optimized for RDF data. Such types of databases are called triplestores. One of the best on the market today are *GraphDB from Ontotext* und *Virtuoso from OpenLink*.

Figure 2-3: Famous triplestore logos

(a) Ontotext GraphDB logo

(b) OpenLink Virtuoso logo

2.4 Linked Data

The term *Linked Data* refers to a set of best practices for publishing and connecting structured data on the Web. These best practices have been adopted by an increasing number of data providers over the last years, leading to the creation of a global data space containing billions of facts - the *Web of Data*.

The Web today relies on HTML documents connected by untyped hyperlinks. Linked Data on the other hand needs documents containing data in RDF format. However rather than simply connecting these documents, Linked Data uses RDF to make typed statements that link arbitrary things in the world. The result is what we refer to as the Web of Data - a network of connected data pieces (facts) that are strongly interconnected enabling easy machine processing and new types of applications based on this technology.

There is already a lot of structured data accessible on the Web through Web 2.0 APIs, e.g. the APIs from Google, Yahoo or Amazon. Compared to them, Linked Data has the advantage of providing a single, standardized access mechanism instead of relying on diverse interfaces and result formats. This gives the following capabilities regarding data sources:

- Easy crawling by search engines.
- Access using generic data browsers.
- Adds links between data from different data sources.

2.4.1 Linked Data Principles

Berners-Lee (2006) outlined a set of "rules" for publishing data on the Web in a way that all published data becomes part of a single global data space:

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs, so that they can discover more things.

These rules have become known as the *"Linked Data principles"*, and provide a basic recipe for publishing and connecting data using the infrastructure of the Web while adhering to its architecture and standards.

2.4.2 Linked Open Data (LOD) project

The most popular example of adoption and application of the Linked Data principles is the *Linked Open Data* project, a big community effort that started in January 2007 and is supported by the W3C Semantic Web Education and Outreach Group. The ongoing aim of the project is to start building the Web of Data by identifying existing data sets that are available under open licenses, convert them to RDF, according to the Linked Data principles, and publish them on the Web.

The early stages of the project involved researchers and developers from universities and small companies. Since then the project has grown considerably and large organisations such as the BBC, Thomson Reuters and the Library of Congress started

contributing to the project. This growth is enabled by the open nature of the project, where anyone can participate simply by publishing a data set according to the Linked Data principles and interlinking it with existing data sets. The figures below depict the range and scale of the whole initiative:

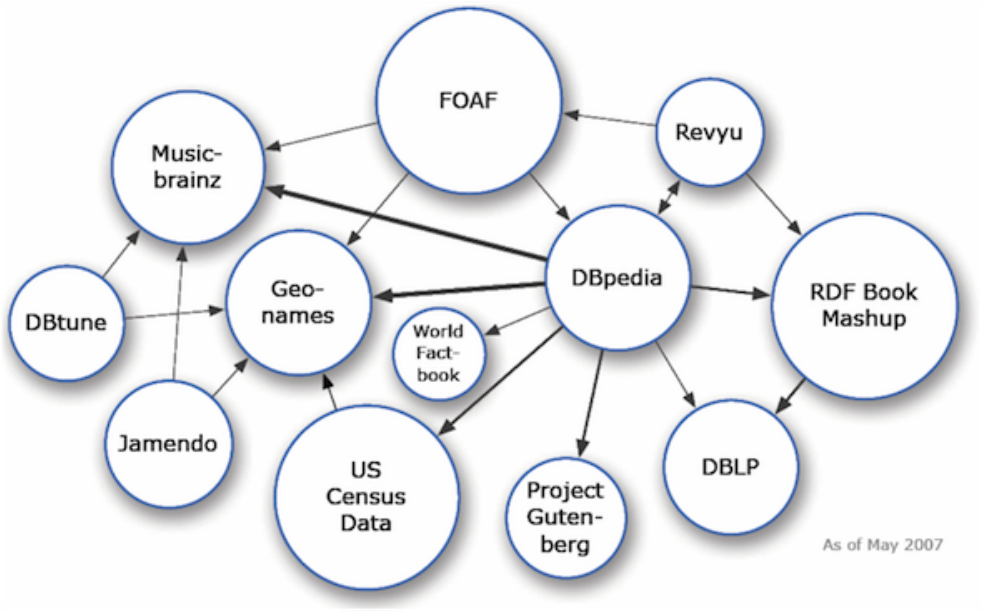

Figure 2-4: Linked Open Data state - May 2007

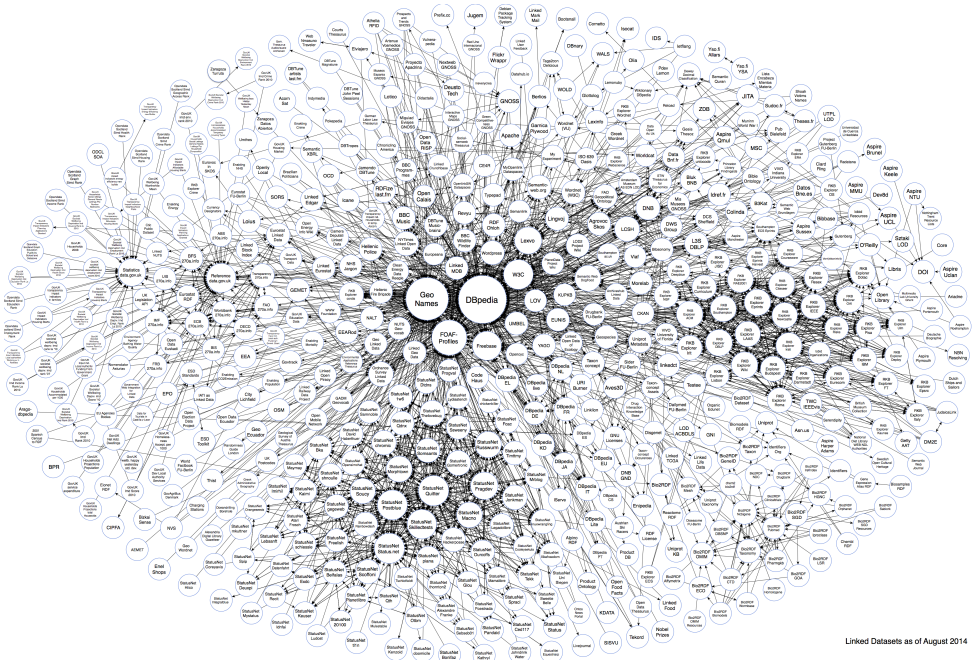

Figure 2-5: Linked Open Data state - August 2014

2.4.3 Publishing Linked Data on the Web

Publishing a data set as Linked Data on the Web involves the following three basic steps:

1. Assign URIs to the entities described by the data set. These entities must be retrievable (dereferenceable) from that URI over the HTTP protocol as RDF representations.

2. Set RDF links to other data sources on the Web, so that clients can navigate the Web of Data as a whole by following RDF links.

3. Provide metadata about published data, so that clients can assess the quality of published data and choose between different means of access (SPARQL endpoint, RDF dumps) besides dereferenceable URIs.

As already stated above URIs should be valid, i.e. a request to that URI should provide the searched entity's RDF representation. Data providers can choose between two HTTP URI usage patterns to identify entities:

- **303 URIs** - use a special HTTP status code, "303 See Other", to distinguish non-document resources from regular web documents. The server is configured to answer requests to such URIs with a 303 HTTP status code and a *Location* header providing a new URL to a document describing the resource, e.g.:

    1. Make request to `http://example.com/id/alice`.
    2. Server responds with 303 status code and employs content negotiation (`Accept: application/rdf+xml`, other type or none) to send either the URL of an HTML document or RDF.
    3. Requests for HTML would be redirected to `http://example.com/people/alice` and requests for RDF data would be redirected to `http://example.com/data/alice`.

- **Hash URIs** - URIs that contain a special part called a *fragment* which is usually not interpreted as part of the URI, e.g. `http://example.com/data#alice`.

Requesting clients will always strip off the fragment part, resulting in a request to this URI: `http://example.com/data`, which will serve an RDF document describing all entities under the `"data"` namespace. Again by employing content negotiation one can redirect to an HTML document (`http://example.com/data.html`) or other RDF descriptions (`http://example.com/data.rdf`).

Hash URIs have the advantage of usually making less requests in order to retrieve a whole set of RDF descriptions, but if the data set is huge and you only want to retrieve a particular description of an entity then 303 URIs are better suited due to having more fine-grained control on the server side.

In an open environment like the Web, different information providers publish data about the same real world entity like a geographic location or a person and thus introduce different URIs to identify the same entity. DBpedia uses the URI `http://dbpedia.org/resource/Berlin` to identify Berlin, while Geonames uses the URI `http://sws.geonames.org/2950159/` to identify Berlin. As both URIs refer to the same thing, they are called URI aliases. URI aliases are common because it cannot be expected that all information providers agree on the same URIs to identify an entity. URI aliases also provide an important social function to the Web of Data as they are dereferenced to different descriptions of the same entity and allow different views and opinions to be expressed. In order to still be able to track these diverse opinions, it is a common practice that data providers set `owl:sameAs` links to all URI aliases they know about.

Chapter 3

Smart processing of news

3.1 Semantic News Pipeline

This chapter will present the whole news processing pipeline from raw text data to interlinked RDF with other Linked Open Data data sets. The steps will include extraction of relevant entities and relationships between them, conversion to RDF, a data preparation step that includes data cleanup and interlinking with other public RDF data sets. Finally everything will be imported in an Ontotext GraphDB™ triplestore repository where interesting aggregating SPARQL queries will be executed against the imported metadata to find trends of popularity.

3.1.1 Concept Extraction

The first processing step of our whole pipeline is the extraction of relevant entities from free text, in this case news articles, and finding relationships between them. Entities are called those phrases in text that usually carry most of the meaning in a piece of information. They are divided into categories/classes of real world things, e.g. people, organisations etc.. When examining the context of the article being processed, different entities could be interlinked based on it. Those are the relationships between them which add significant information value to the discovered metadata.

The *Ontotext Concept Extraction Pipeline* is a tool for automated analysis of

large volumes of textual content, through which mentions of specific concepts and relationships between them can be discovered and represented in a machine-processable format. It enriches the content aggregated from various news sources based on data from DBpedia, WikiData, GeoNames and others. The pipeline utilizes a wide variety of linguistic and algorithmic resources such as semantic gazetteers, rules triggered by particular linguistic patterns, various statistical models for classification and sequence tagging trained against human-annotated corpora, etc.. These resources are chained together in a sequence (hence the name pipeline) that provides content enrichment of gradually increasing complexity, where each phase builds upon the results produced by the one that precedes it. The concept extraction pipeline recognises mentions of entities such as Person, Organisation, and Location, and links them if possible to a particular Knowledge Base (DBpedia, WikiData, GeoNames etc.). It also tags the particularly relevant noun phrases, called key phrases thus allowing a fast grasp into the topic of the document. Furthermore it detects relationships between the extracted entities, as well as their relevance and confidence to the text. Due to the fact that text mining and natural language processing techniques are outside the scope of this thesis, we will not examine the workflow of this processing step in any more details.

3.1.2 RDF Data Preparation

After processing some raw news data to discover relevant mentions and converting them to RDF, the next step is to interlink NOW RDF data with other data sets that will give more information about those mentioned entities. For this purpose the following mappings and data sets are used:

- DBPedia
- GeoNames
- `sameAs` statements between DBPedia and Geonames
- NOW RDF news data

FactForge News

Ontotext has an old free service called *FactForge*, found at `http://factforge.net`, which is something like a search engine for RDF data. It allows you to write SPARQL queries against these huge amounts of data, e.g. you could ask *"Which are all the airports that in 50km of range of my location?"*. It is basically a triplestore repository with imported and interlinked data sets like GeoNames, DBPedia, etc. and some custom mappings. The setup will resemble the setup that we will describe in this and later processing steps with our news data in this chapter. Eventually it will become the new FactForge service. It is now called *FactForge News* due to the added news data that was missing in the old one.

Data Cleanup

In order to make everything work correctly, some data cleanup is required and some additional mappings have to be introduced, which will be prefixed like this:

```
PREFIX ff-map: <http://factforge.net/ff2016-mapping/>
```

1. **Delete dbo:subsidiary misuse linking company to country or town**

   dbo:subsidiary is erroneously used to indicate that a company has a subsidiary in a specific country or town. This implies that the company owns the country or the town so it needs to be removed. The following query fixes the most obvious problem:

   ```
   PREFIX dbo: <http://dbpedia.org/ontology/>

   DELETE { ?parent dbo:subsidiary ?child }
   WHERE {
     ?parent dbo:subsidiary ?child.
     ?child a ?class .
     FILTER (?class IN (dbo:Town, dbo:Country))
   ```

```
}
```

2. **Additional mappings fixing messy industry classifiers**

   DBPedia's industry classifiers are very noisy. There are multiple identifiers used for one and the same syntactic variation. They are also unstructured, e.g. *RailTransport* is by no means connected to *Transport* and *Brewery* is not related to *Food_and_beverages*.

   In order to fix this mappings between the most often used ones, their "synonyms" (via **ff-map:industryDuplicate**) and their more specific variants (via **ff-map:industrySubsector**, transitive) are introduced. Both of those are subproperties of **ff-map:industryVariant**, which can be used when one needs to get all the duplicate and more specific industry identifiers.

```
ff-map:industryDuplicate rdfs:subPropertyOf ff-map:industryVariant .
ff-map:industrySubsector rdfs:subPropertyOf ff-map:industryVariant .
ff-map:industrySubsector a owl:TransitiveProperty .
ff-map:industryVariant a owl:TransitiveProperty .
ff-map:industryDuplicate a owl:SymmetricProperty .
```

   Finally, **ff-map:industryCenter** is used to denote the representative id for this industry (this simplifies some queries).

   A sample of such mappings for the retail industry:

```
dbr:Retail ff-map:industryCenter dbr:Retail .
dbr:Retail ff-map:industryDuplicate
    dbr:Retail, "Retail"@en,
    dbr:Retailing, dbr:Retailer, "Retailing"@en, "retail"@en,
    dbr:Retail_industry, dbr:Retail_trade, "Retailer"@en .
dbr:Retail ff-map:industrySubsector
    dbr:Online_retailing, dbr:Online_Retail,
```

```
    "Clothing retail"@en, "Retail coffee and tea"@en,
    dbr:Online_retailer, "Retail Coffee"@en .
```

After successfully applying all those mappings one could now query, e.g. all software companies like this:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX ff-map: <http://factforge.net/ff2016-mapping/>

SELECT DISTINCT ?org  {
  dbr:Software ff-map:industryVariant ?industry .
  ?org dbo:industry ?industry .
}
```

3. **Different properties indicating industry**

   Unfortunately DBPedia uses lots of properties to denote an industry (not only **dbo:industry**). This could be easily checked for a particular industry, e.g. **dbr:Software**:

```
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?p (COUNT(*) as ?c) {
  ?x ?p dbr:Software
}
GROUP BY ?p ORDER BY DESC(?c)
```

   The result of this query is the following:

   dbp:industry has to be mapped to **dbo:industry** but it is not. The following query shows that companies like Fog Creek Software, Microsoft India PL, etc., are not being mapped with **dbo:industry**:

dbp:industry	1158
dbo:industry	1100
dbp:products	74
dbo:product	120
dbp:genre	25
dbo:genre	13
dbp:services	8
dbo:service	11
dbp:type	8
dbo:type	5
dbp:data	4
dbo:division	3

Table 3.1: DBPedia diverse industry mapping properties for dbr:Software industry

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT COUNT(*) {
  ?x dbp:industry dbr:Software
  FILTER NOT EXISTS {?x dbo:industry dbr:Software}
}
```

A possible reason could be that in the infobox templates used for these companies, nobody mapped to **dbo:industry**. Probably the same defect applies to other pairs (**dbp:products vs dbo:product, dbp:genre vs dbo:genre,** etc). So in order to find more companies, we compare counts from companies mapped only with **dbo:industry** with companies mapped with additional mappings (**dbp:products, dbp:genre, dbp:services, dbp:type**, etc.):

```
SELECT COUNT (DISTINCT ?org)  {
  ?org dbo:industry dbr:Software
}
```

```
SELECT COUNT (DISTINCT ?org)  {
  ?org dbp:industry|dbo:industry|dbp:products|
       dbo:product|dbp:genre|dbo:genre|dbp:services|
       dbo:service|dbp:type|dbo:type dbr:Software
}
```

First query yields 1100 companies and the second - 1418 which is almost a third more. To battle these inconsistencies a collective property **ff-map:industry** is introduced which should only be applied to real "industry" values. While it is correct to map **dbp:genre dbr:Software** to **ff-map:industry dbr:Software**, mapping **dbp:genre dbr:Drama** to **ff-map:industry dbr:Drama** will not be. We define an "industry" to be those values that have some **ff-map:industryVariant** property:

```
PREFIX ff-map: <http://factforge.net/ff2016-mapping/>
PREFIX onto: <http://www.ontotext.com/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>

INSERT {GRAPH ff-map: {?x ff-map:industry ?industry}}
WHERE {
  ?x dbp:industry|dbo:industry|dbp:products|
       dbo:product|dbp:genre|dbo:genre|dbp:services|
       dbo:service|dbp:type|dbo:type ?industry
  FILTER EXISTS {[] ff-map:industryVariant ?industry}
}
```

4. **Add GeoNames parents**

Unfortunately there are countries that do not know their continent, e.g. **dbr:Western_Europe** is not child of **dbr:Europe**. To fix this issue we have

to add correct parent relatioships where missing. This is a small snippet of the most obvious relationships missing in Turtle format:

```
@prefix gn: <http://www.geonames.org/ontology#> .
@prefix dbr: <http://dbpedia.org/resource/> .

dbr:Western_Europe  gn:parentFeature dbr:Europe.
dbr:Southern_Europe gn:parentFeature dbr:Europe.
dbr:Melanesia       gn:parentFeature dbr:Oceania.
dbr:Switzerland     gn:parentFeature dbr:Europe.
dbr:Norway          gn:parentFeature dbr:Europe.
dbr:Iceland         gn:parentFeature dbr:Europe.
dbr:Liechtenstein   gn:parentFeature dbr:Europe.
dbr:W_National_Park gn:parentFeature dbr:Africa.
```

5. **Different country identifier mappings**

   There are different country URIs and literals that identify a country. There has to be a way to unify them. For the purpose the property **ff-map:hasCountryVariant** is introduced which is inverse to itself, e.g. USA refers to United States and United States are referred by USA:

```
ff-map:countryVariantOf owl:inverseOf ff-map:hasCountryVariant .
```

   Now all country variants have to be mapped with the new property, e.g.:

```
dbr:United_States ff-map:hasCountryVariant "United States"@en .
dbr:United_States ff-map:hasCountryVariant dbr:USA .
dbr:United_States ff-map:hasCountryVariant "USA"@en .
```

6. **Associate countries and organisations**

   For the sake of better analytics, it is beneficial to have good mappings between organizations and countries. These are the most frequently used properties between organisations and locations (based on DBPedia 2014 stats):

```
dbo:location
dbp:location
dbo:locationCity
dbp:locationCity
dbp:locationCountry
dbo:locationCountry
dbo:foundationPlace
dbp-prop:headquarters
```

   To further abstract synonyms and partial inconsistencies we insert new mappings:

```
ff-map:orgHQCity rdfs:subPropertyOf ff-map:orgCity .
ff-map:orgHQCountry rdfs:subPropertyOf ff-map:orgCountry .
ff-map:orgFoundationCountry rdfs:subPropertyOf ff-map:orgCountry .
```

   To fully integrate these new mappings a series of SPARQL Update queries have to be executed in a similar manner as previous mappings.

3.1.3 Triplestore persistance

The setup for persisting the data includes a triplestore. For this project Ontotext's *GraphDB*^{TM} is used - one of the fastest and most scalable triplestores which provides real-time OWL inference.

Ontotext GraphDB™

Some of GraphDB's features include:

- **GraphDB Workbench** - database administration tool.
- **Connectors** - external plugins synchronizing repository data with Lucene, Solr or Elasticsearch which implement full-text search - enable fast retrieval and facet (aggregated) search by building an external index on custom property chains.
- **Built-in Indices** - optional indices on predicates, literals and contexts (named graphs).
- **Built-in plugins** - a collection of standard plugins used on demand and configurable through SPARQL. Plugins used in this project are:
  - **RDF Rank plugin** - RDF Rank is an algorithm that identifies the more important or more popular entities in the repository by examining their interconnectedness. The popularity of entities can then be used to order query results in a similar way to the internet search engines, the way Google orders search results using PageRank.
  - **Geospatial plugin** - support for 2-dimensional geo-spatial data, which uses the *WGS84 Geo Positioning RDF vocabulary (World Geodetic System 1984)*. Special indices can be used for this data, which permit the efficient evaluation of special query forms and extension functions that allow finding locations, which are:
    - within a certain distance of a point, i.e. within the specified circle on the surface of the sphere (Earth);
    - within rectangles and polygons, where the vertices are defined using spherical polar coordinates;

Tuning GraphDB™

In order to optimize the speed and efficiency of our queries we have to pick which of GraphDB™'s features to use and how to tune them appropriately.

Geospatial index will be enabled because GeoNames is imported which contains 2-dimensional data from *wgs84* ontology.

Due to all of the news map queries targeting popularity, a metric will be needed that will allow sorting the end results based on their interconnectedness. So **RDF Rank** will be generated that will set weights on each node of the graph. See *Figure 3-1* to understand how the RDF Rank plugin works and how to enable it.

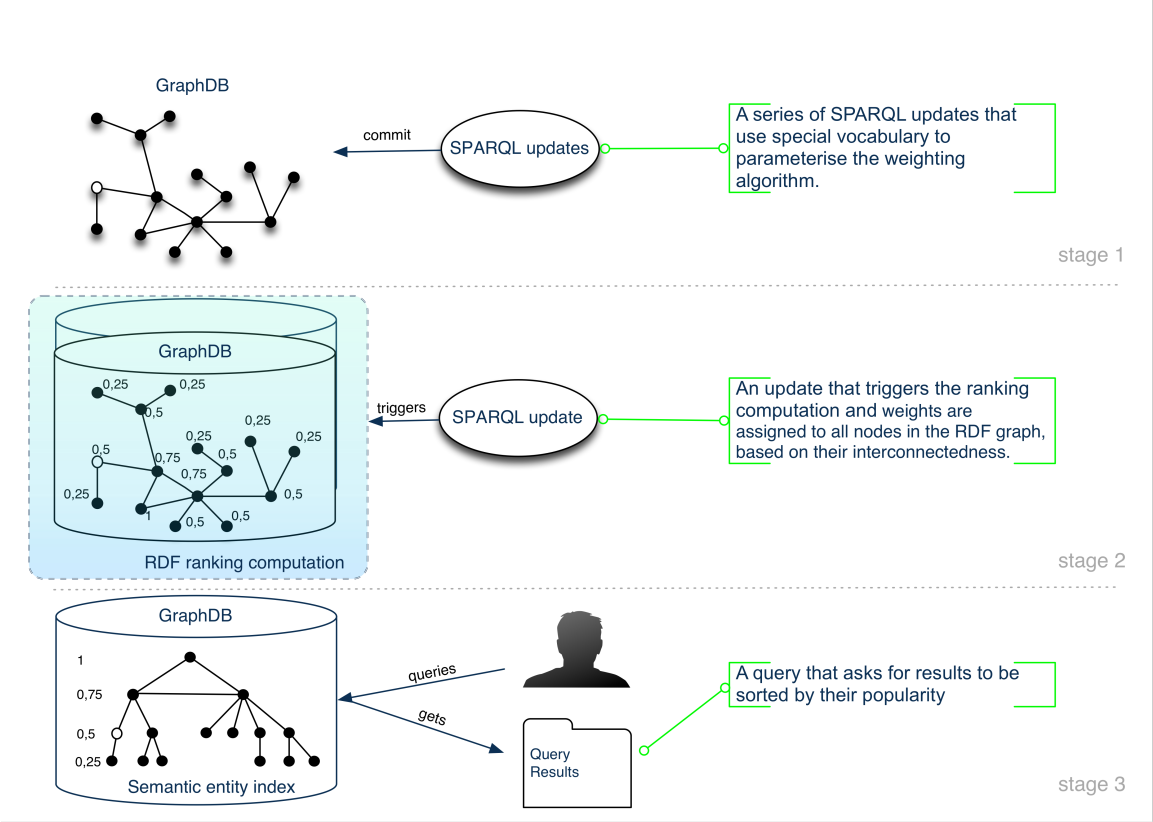

Figure 3-1: RDF Rank plugin workflow

Context indices are another optimization that could be utilised, so they will be enabled. There are two types of context indices used to speed up query evaluation when searching statements via their context identifier. These indices are the PCSO (Predicate-Context-Subject-Object) and the PCOS (Predicate-Context-

Object-Subject) and they are switched on together.

Finally the **Lucene Connector** will be used on some key entities from DBPedia and GeoNames and also on news data as a separate index - title, content, creation dates, etc. The Connectors provide synchronisation at the entity level, where an entity is defined as having a unique identifier (a URI) and a set of properties and property values. In terms of RDF, this corresponds to a set of triples that have the same subject. In addition to simple properties (defined by a single triple), the Connectors support property chains. A property chain is defined as a sequence of triples where each triples object is the subject of the following triple.

3.1.4 RDF Data Visualization

After finally importing and interlinking the data, the next step is to write the news aggregation SPARQL queries. This section aims to present some of the cool data analytics features of GraphDB^{TM} Workbench which will help us to better know our data and to more easily write the SPARQL queries. The next three figures present some of the data visualisation diagrams available in the Workbench to give an in-depth overview of the imported data:

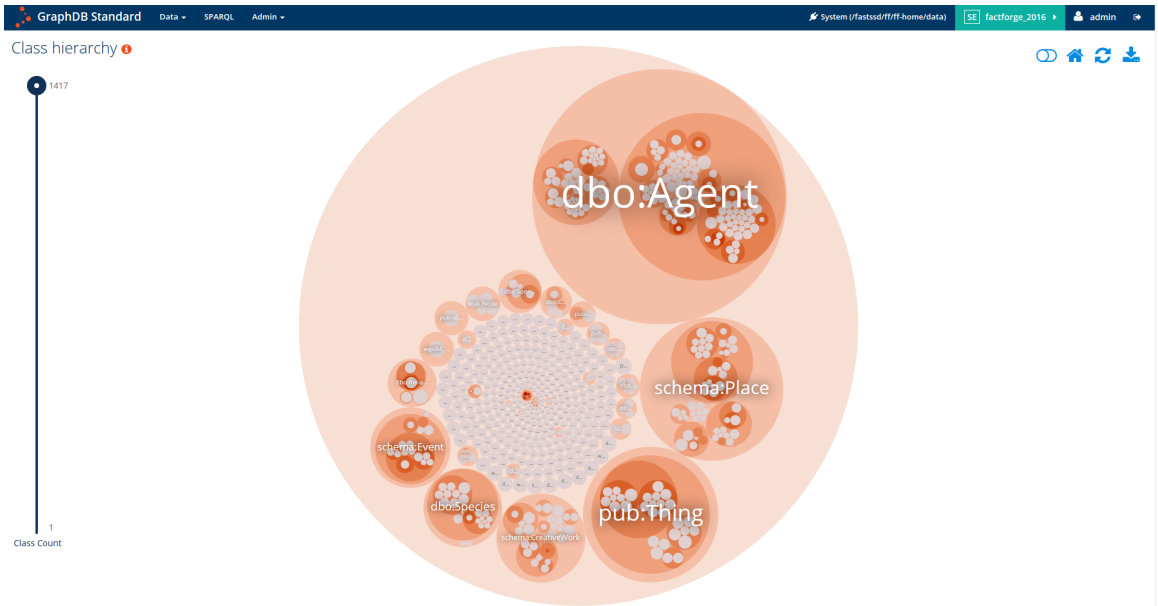

Figure 3-2: RDF Class Hierarchy diagram overview

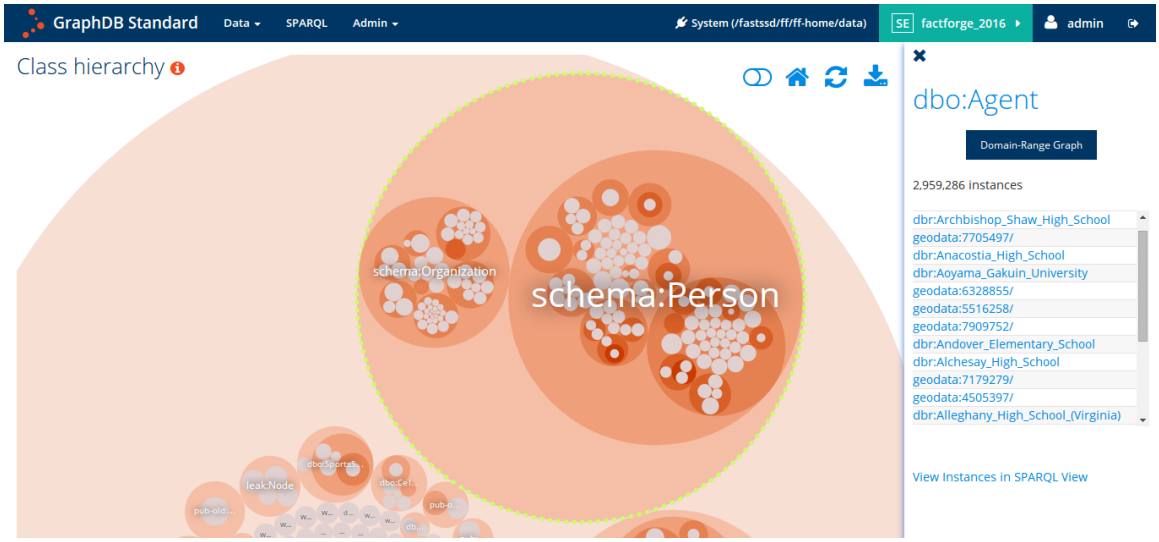

Figure 3-3: RDF Class Hierarchy diagram zoomed on dbo:Agent class

Figure 3-2 shows the RDF class hierarchy of our complete data set in the form of a circle packing diagram. There are three types of circles depicting RDF classes:

- *Top level parents* - circles with children and without parents.
- *Parents* - circles with children and with parents.
- *Children* - circles without other children but with/without parents.

Our data set contains over 1400 RDF classes, which is quite a lot. There are classes from DBPedia, GeoNames, NOW News data, etc., all interlinked with each other where it makes sense. This is why visualisation tools could be very helpful to get a better idea of the current schema of your data because it changes very dynamically in time when new data or ontologies are being added.

As an obvious example we see on the diagram that `dbo:Agent` is a top level parent but its child labels are not visible. The next picture (*Figure3-3*) shows exactly them. They become visible after clicking on the circle with `dbo:Agent` label which zooms to it and shows their labels which are `schema:Person` and `schema:Organization` and hides the parent's label. Zooming can be continued further until a child class is reached.

Figure 3-4 presents another type of data visualisation called the *Domain-Range*

graph. In RDF terminology *domain* and *range* properties are intended to give you insight about the way the predicate links a subject to an object. In the case of the domain property, when you link a subject to an object using a property with this associated attribute, then the subject qualifies as a type of thing specified in the domain. Range property works exactly like the domain, but it applies to the object of the statement and not the subject, e.g.:

```
"Mother"(domain) hasChild "Child"(range)
```

A more complex example is the child-parent relationship:

```
"Child" (domain) hasParent "Mother","Father" (range)
```

Here there are two parents (objects), which are pointed by the child (subject). On the current picture the class `dbo:Agent` is selected and all of its ingoing and outgoing predicates could be examined for domain and range based on the classes to which they are connected either as subjects or objects. This helps to trace relationships easier and also do some form of graph navigation because a double click on some of the green nodes selects that node and rerenders the diagram with its ingoing and outgoing properties.

Figure 3-4: RDF class dbo:Agent domain-range graph diagram

The *Semantic News Map* application presented in *Chapter 4* will use `FactForge News` as a remote data repository.

3.1.5 News Aggregation SPARQL queries

This subsection will present some of the techniques for normalization used in the news aggregation SPARQL queries. There are two questions that have to be answered when examining entity popularity:

- *Which entities are relevant to the topic?*
- *How relevant to the topic they are?*

The NOW news data has the following URIs:

```
PREFIX news-pub: <http://ontology.ontotext.com/publishing#>
PREFIX news: <http://ontology.ontotext.com/taxonomy/>
```

There is a property called `news-pub:relevanceScore` whose value is generated from the concept extraction pipeline described in the first processing step. It measures the count of mentions of the entity in the news article plus its positioning in it, i.e the more the entity is placed at the beginning of the article the better the chance it could be part of a summary which automatically boosts its importance. `news:exactMatch` is a property which links mentioned entities in an article against external resources such as from DBPedia, e.g. *"Volkswagen"* classified as *Organisation* is matched against `dbo:Organisation` from DBPedia. There are also two very important additional mappings regarding relevance.The first one is `ff-map:agentRelation` which gives all related entities to the observed one by utilizing GraphDB^{TM}'s **RDF Rank** plugin which generates node weights based on interconnectedness to other nodes on the graph. This mechanism is called *spreading activation*, i.e. a cognitive technique that starts with initiating weights or "activation" of source nodes and then iteratively propagating or "spreading" that activation out to other nodes linked to the them. This powerful mechanism will allow for tracing hidden relationships in our *hidden*

champions scenario. The second one is `ff-map:longtermPopularity` which takes into account mentions in the last year when evaluating current popularity.

By combining the above mechanisms we could now achieve the following scenarios:

1. **Direct popularity** will be calculated through simply counting the number of mentions in the examined news articles and summing up with `news-pub:relevanceScore`.

2. **Relative popularity** will take into account the formula:

   ```
   (daily popularity * daily popularity / longterm popularity) * 100
   ```

   Daily popularity is the maximum `news-pub:relevanceScore` found and longterm popularity comes from matching against `ff-map:longtermPopularity`.

3. **Hidden champions** are matched against `ff-map:agentRelation` to trace their indirectly related connections, e.g. if "Sergey Brin" is mentioned in the article, he will be automatically related to "Google" and the result from the query will return "Google".

Finally `news:exactMatch` is used everywhere because matched entities against, e.g. DBPedia or GeoNames usually have better descriptions and more information related to them.

Chapter 4

Semantic News Map Application

4.1 Overview

This chapter will present the end user web application which will visualise the already processed news in various interesting ways. Its goal is to allow you to "read" news in an entirely different way. Usually in articles and almost any other forms of text, the relevant and important information consists in probably less than half the whole text. Harnessing the power of semantics as described in *Chapter 3* with presenting a whole processing pipeline for extracting meaningful information from news data, we now possess an extremely powerful asset extracted in an automated way. The next step is how to present this powerful asset to the user in a unique, effective and interesting way.

4.1.1 Main Functionalities

The *Semantic News Map* application consists of a set of visualisations that present the relevant content of a news article or a group of news articles by doing aggregations on the already extracted metadata through SPARQL queries against a GraphDB™ repository with all needed data imported. There are two main types of visualisations:

- Word Cloud
- Geographic Heat Map

There are a few different versions of word clouds that present data based on different criteria:

1. **Direct popularity** of the concepts based on mentions count in the news and degree of importance of these news.
2. **Relative popularity** - where the popularity within the current day is normalized against the popularity within the last year.
3. Concepts that are **"hidden champions"** - entities that have an indirect relationship to the currently most popular ones.
4. Combination of **"hidden champions"** and **relative popularity**.

The geographic heat map has two main functionalities:

1. Colors all world countries with different degrees of color intensity according to a specified color scale, based on the number of news in which they are mentioned and displays the news count of each when hovering with the mouse over them.
2. Presents the list of news mentioning a specific county in a side panel when clicking on it.

The following figures will present some screenshots of the whole web application.

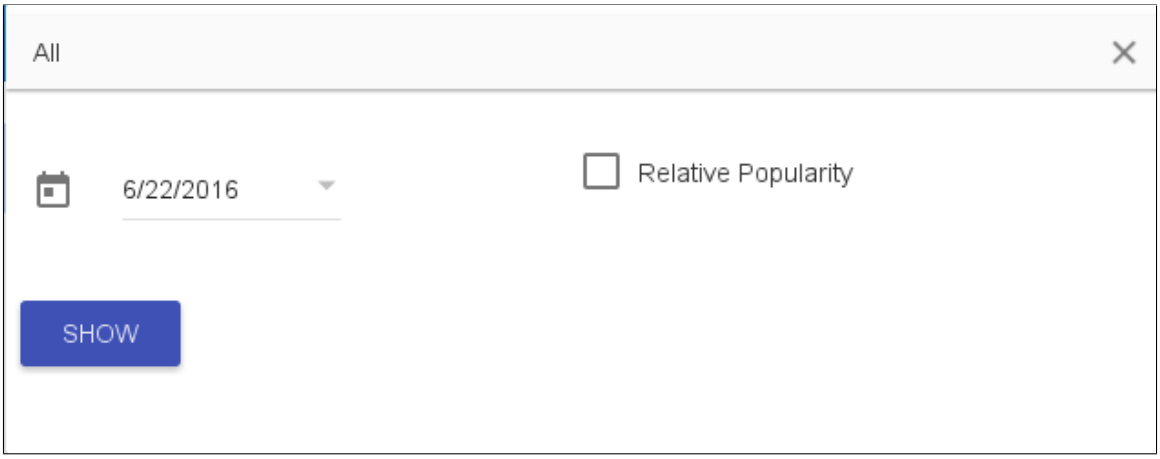

Figure 4-1: News search dialog

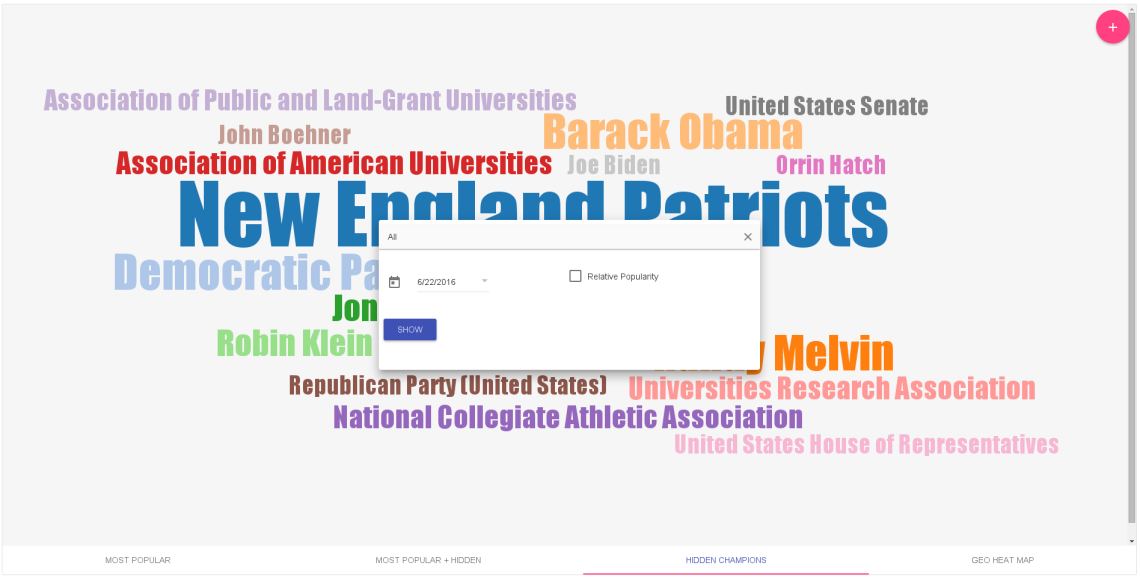

Figure 4-2: News search dialog popping in the middle of the screen

The figures above show the news search dialog in closeup and as a position on the screen when it appears. To see the dialog you have to click the round plus button on the top-right corner of the screen. This dialog serves the purpose of gathering parameters for news aggregation. *Figure 4-1* exposes the elements in close detail. On the top there is a text field with auto-completion to select a news category like Sports, Lifestyle, Business, etc.. Beneath it is a date picker field to choose to aggregate news only published on the selected date. The *"Relative popularity"* checkbox calculates relative popularity of the entities according to their popularity in the last year as was explained earlier. Finally there is a *"Show"* button which sends a request to the backend with the selected parameters and all types of visualisations start loading.

Figure 4-3: Hidden Champions criterion word cloud visualisation

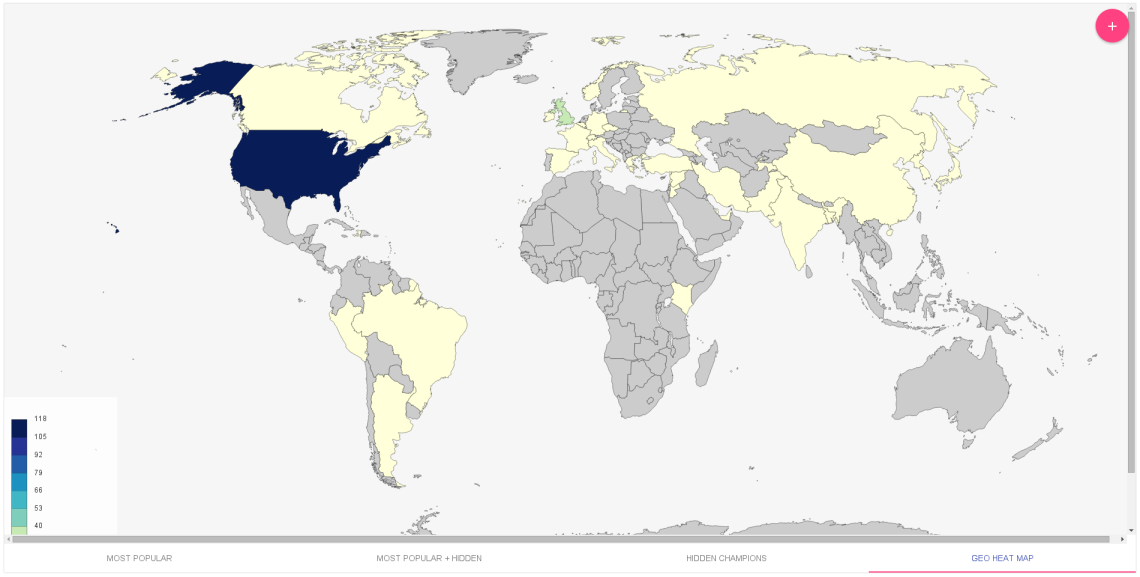

Figure 4-4: News frequency geographic world heat map

Figures *4-3 and 4-4* show how a word cloud and a geographic world heat map look like. In order to switch between visualisations you just have to click the tabs below which have labels showing what to expect. There are 4 tabs available: three word clouds and one geographic heat map. They are labelled as:

1. **MOST POPULAR** - word cloud diagram, showing direct popularity.

2. **MOST POPULAR + HIDDEN** - word cloud diagram as the previous, but showing the related entities of the popular ones.

3. **HIDDEN CHAMPIONS** - word cloud diagram showing only related entities to the popular ones.

4. **GEO HEAT MAP** - geographic heat map diagram, showing frequency of news mentioning a specific country.

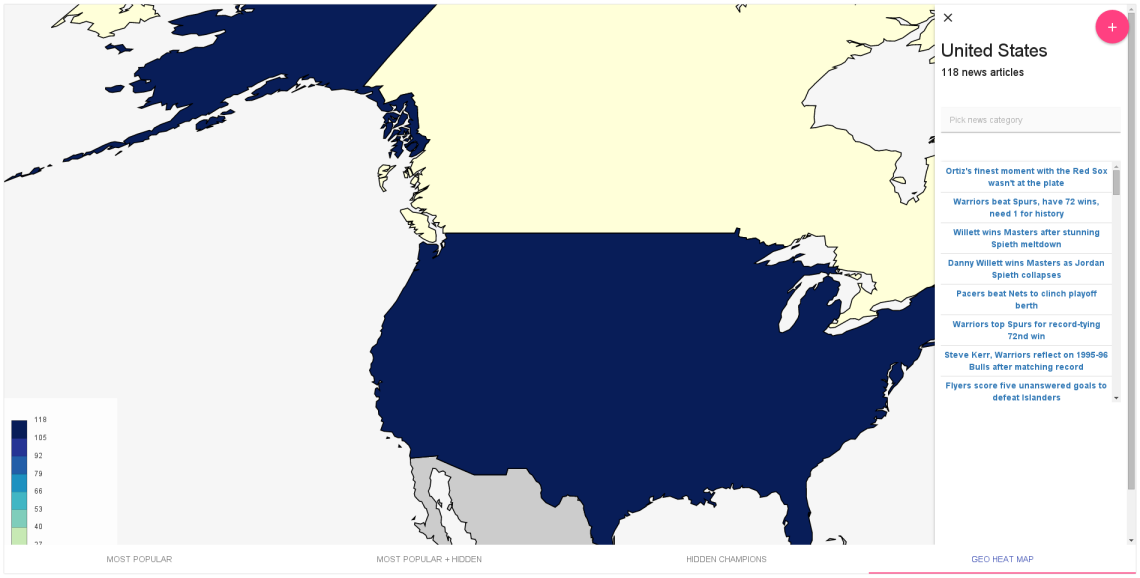

Figure 4-5: News mentioning USA and heat map zoomed on USA

Figure 4-5 shows what happens when you click on a country. The screen zooms to that country and a side panel on the right side pops up showing the name of the country, the news count of news mentioning it and a list of the news articles themselves with links leading to the Ontotext NOW web application (*Figure 4-6*) for reading them. The news article displayed there has highlighted words which when you hover on, a tooltip pops up showing the type of entity discovered, its relevance and confidence scores. In this case when you hover over `"Volkswagen"`, you see it is recognized as an `Organisation` with a relevance score of 72.9% and a confidence score of 92.8%.

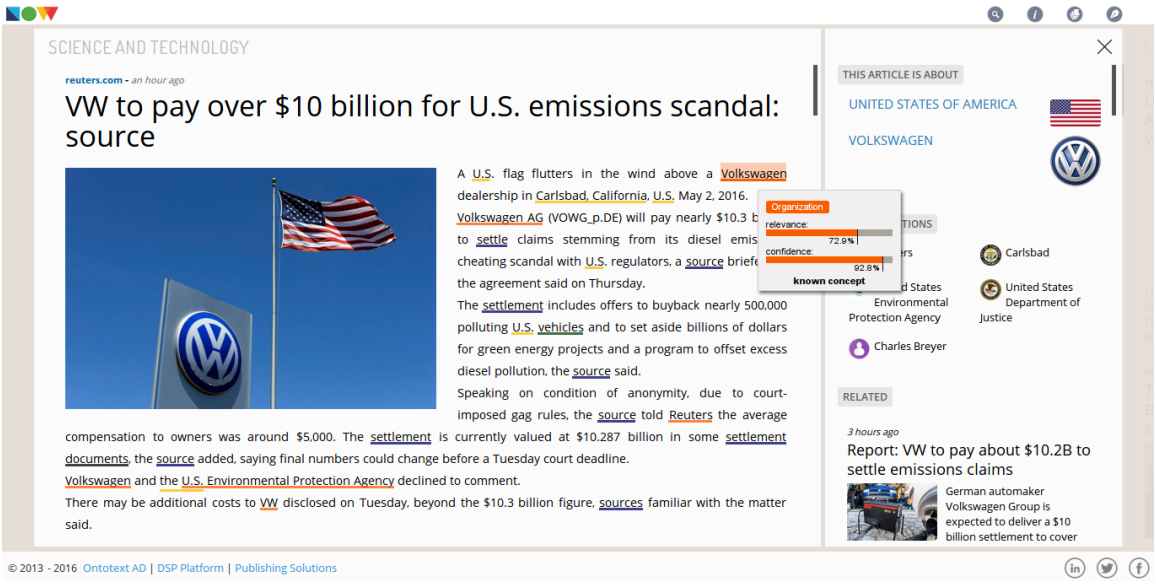

Figure 4-6: NOW news article from Science and Technology category

The last two screenshots (*Figures 4-7 and 4-8*) show the details views that show up when you click on an entity from a word cloud. This view is presented again in a tabbed user interface with two tabs:

1. **NEWS MENTIONING ENTITY**
2. **NEWS MENTIONING RELATED ENTITIES**

Both views display a table of SPARQL results for things like: news title, news date, news link, relevance score and related entities details for the *NEWS MENTIONING RELATED ENTITIES* results. These results also could be searched in the upper right search field.

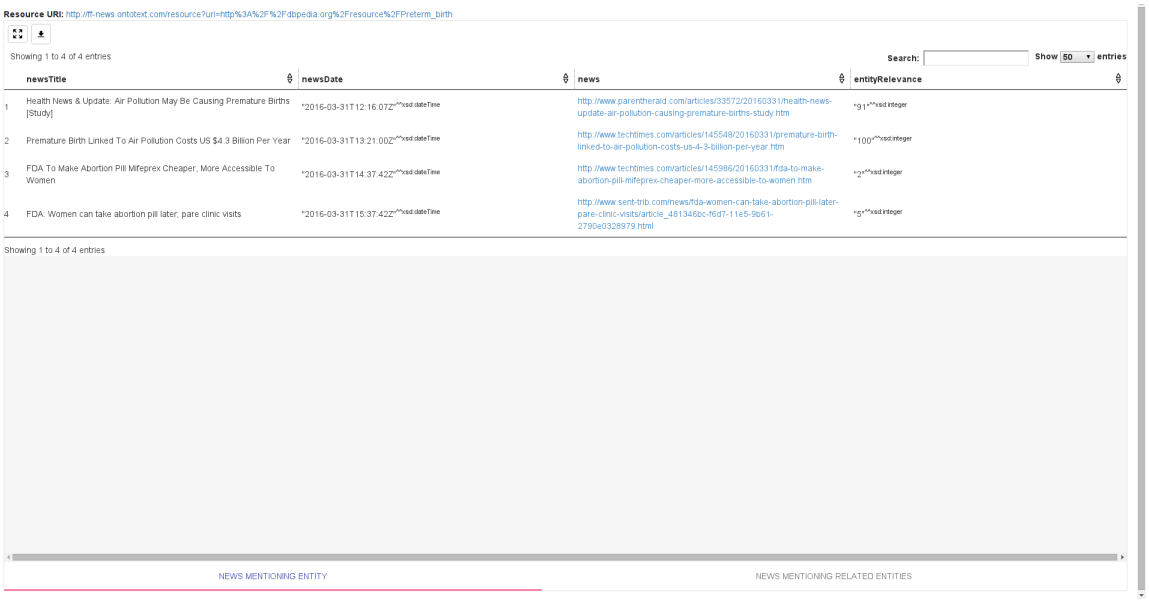

Figure 4-7: News mentioning entity view

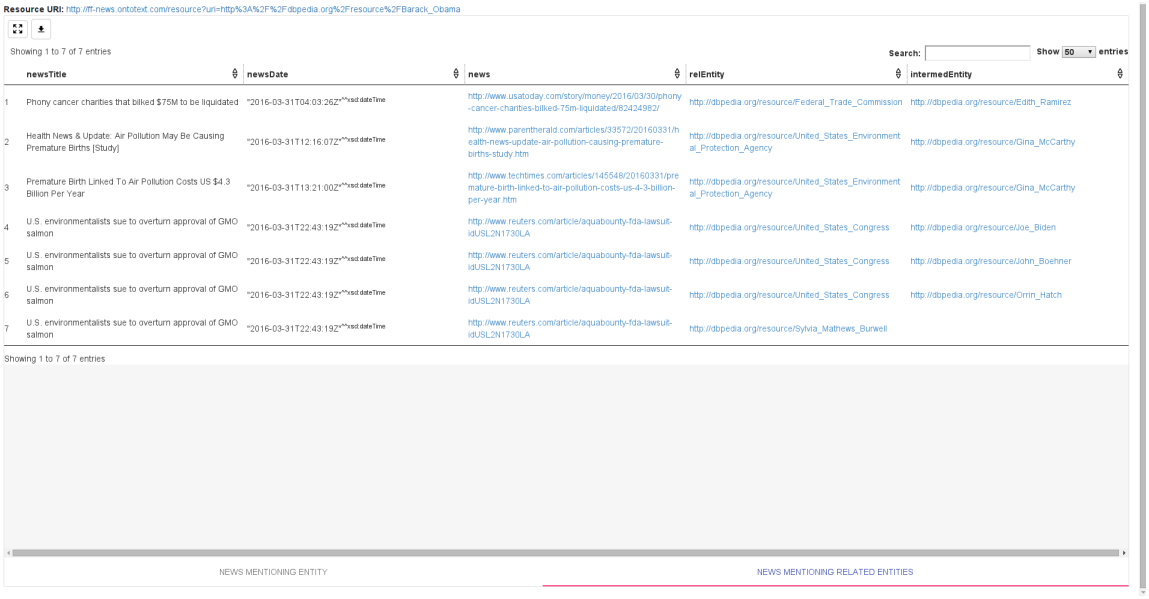

Figure 4-8: News mentioning related entities view

4.1.2 Architecture and used technologies

The whole application is based on a classic `Client-Server` architecture. There is a backend part and a frontend part. The entire project is managed by `Apache Maven` -

a build system for compiling, packaging and dependency management of Java-based projects. It uses the standard Maven project archetype and contains two modules:

```
semantic-news-server
semantic-news-ui
```

The first is the backend module which contains only `Java 8` code, connects to the FactForge News GraphDB™ repository and executes the news aggregation SPARQL queries which will feed the frontend of the application with data. And the second module is the frontend module which only contains `Javascript`, `HTML` and `CSS` code but its build lifecycle is again managed by Maven.

Communication between both modules is accomplished via REST (Representational State Transfer) in order to assure total decoupling of both components so that they can be easily modified without affecting each other. A simple example could be the frontend making a HTTP request to:
`http://<url-host>/rest/semnews/news-mentioning-country?countryCode=USA&from=2016-04-12`
and the backend sending a response containing the results from the execution of the SPARQL query about news titles and URLs of news from 12.April 2016, mentioning the United States in JSON format which the frontend can easily read and handle.

Semantic News Map Server (Backend)

The backend part is written in `Java 8` using the `Spring Framework` as a web framework. More specifically `Spring Boot` is used, which makes it easy to create standalone, production-grade Spring based applications that are extremely easy to deploy. It includes an embedded `Apache Tomcat` application server which eliminates the need for exporting `.war` files, configuring a Tomcat instance separately, copying the `.war` file in the Tomcat distribution and launching the server. Spring Boot generates for you an executable `.jar` file with an already configured embedded application server which only needs to be executed and that is all.

Semantic News Map UI (Frontend)

The frontend part is entirely based on AngularJS Javascript MVC framework. AngularJS is a framework for building Single Page Applications (SPA). This means that the server only serves the `index.html` file on start which then loads all other needed libraries, stylesheets and templates. From there on Angular handles routing, rendering of different views and all other UI specific logic. It will need the backend only as a data provider through AJAX calls to it as was demonstrated above. This style of architecture is called `Client-Side MVC`.

All Javascript code in the project is written using the newest ECMAScript standard ES6 (ES2015) where there are now features like block-scoped variables with `let` instead of `var`, cleaned up syntax for object-oriented programming resembling more Java-style classes and a built-in module system with keywords such as `import`, `export` and `from`. But due to the fact that not all new features are implemented by all browsers, a transpiler called `Babel` is used which translates code written in ES6 to ES5. The module system at the time of writing this thesis is still not available in any browser yet, so a library called `SystemJS` is used which expands `import` statements to SystemJS calls in order to load needed resources.

Javascript dependency management is handled by `NPM (Node Package Manager)` from NodeJS and `JSPM`. JSPM is package manager integrated with SystemJS module loader and works very well with ES6 code.

Most of the UI components are based on Google's design specification *Material Design* used in Android Lollipop and above. AngularJS has an implementation of this spec called `Angular Material` which is used throughout the whole user interface.

All visualizations in the application are SVG-based and implemented with a library called `D3 (Data Driven Documents)`. Word clouds use an additional layout to the library called `d3-cloud` and the geographic heat map also uses an external layout called `d3-geomap`.

Chapter 5

Conclusion

5.1 Analysis

This thesis has demonstrated how to build an intelligent application based on Semantic Web technologies. News were picked as a use case for this project. They posed the challenge to extract their meaning as being free and unstructured text about any kind of subject or area of life, expressing facts, opinions and conclusions from various different topics and categories. The solution that was presented was based partly on text mining partly on semantic technologies. A semantic news pipeline was presented that included processing steps ranging from raw text entity extraction to conversion to RDF data, interlinked with other Linked Open Data data sets, that enabled writing news aggregation SPARQL queries, against a triplestore repository with the data imported, to really get proper insight about trends of popularity of entities, mentioned in the news.

Finally the results of the news aggregation were presented in the form of SVG-based visualisations as word clouds and geographic heat maps with the user's ability to choose a news category of interest, dates for news published in that period and the option to choose between direct popularity or relative popularity normalized within the last year.

5.2 Future directions

Possible improvements to the *Semantic News Map* could be adding a user base which will keep information about news preferences and also could store rendered visualisations as "Favourites" in your account. News details view could be reworked to be more interactive. More experiments could be done with interesting data sets about, e.g. company information (offshore companies, owners, shareholders, etc.).

The power of semantics is limitless. It has the unique property to put the world's knowledge under your disposal and actually give you real insight about it. In a world where information is the most valued asset, this technology is priceless. I truly hope that its popularity will continue to grow so that every person on this world could benefit from it.

Appendix A

Tables

Appendix B

Figures

[2] [4] [3] [1] [5] [6]

Bibliography

[1] Bob DuCharme. *Learning SPARQL.* O'Reilly Media, Inc., 2011.

[2] Paul Warren John Davis, Rudi Struder. *Semantic Web Technologies.* Wiley, 2006.

[3] Dellzell Prestel. *Mathematical Logic and Model Theory.* 2011.

[4] Hector J. Levesque. Ronald J. Brachman. *Knowledge representation and reasoning.* 2003.

[5] Leo Sauermann, Richard Cyganiak, and Max Vlkel. Cool uris for the semantic web. Technical report, Saarlndische Universitts- und Landesbibliothek, Postfach 151141, 66041 Saarbrcken, 2007.

[6] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006.