FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Implementation of a Knowledge Discovery and Enhancement Module from structured information gained from unstructured sources of information

Celso Ricardo Costa

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Ferreira da Silva, PhD

26th July 2010

Implementation of a Knowledge Discovery and Enhancement Module from structured information gained from unstructured sources of information

Celso Ricardo Martins Maia Costa

Mestrado Integrado em Engenharia Informática e Computação

26 de Julho de 2010

Resumo

O conhecimento é um dos maiores diferenciadores entre empresas, é essencial para criar uma vantagem competitiva sobre a concorrência. O acesso ao conhecimento hoje em dia não é restringido por limitações físicas. Toda a informação está a um clique de distância, mas sistemas modernos proporcionam fraco acesso a informação, especialmente se não existirem ferramentas correctas para a localizar e processar. Como parte de um desafio global para desenhar e criar uma arquitectura foi implementada uma prova de conceito de um Sistema de Gestão de Conhecimento, com o apoio de Processamento de Linguagem Natural e informação semântica, seguindo uma filosofia de "Open Innovation". Esta dissertação descreve toda a pesquisa e todo o sistema desenvolvido. O sistema é uma prova de conceito de um método para gerar uma representação semântica de um documento, melhorar o volume de informação actual e respectiva qualidade recorrendo a dados externos e metadados para aumentar a base de conhecimento existente.

Com a expansão da web semântica, uma rede de dados com um significado definido, transversal a vários domínios, permite aos computadores e utilizadores trabalhar em cooperação para gerar novas possibilidades de representar e integrar o conhecimento emergente.

A análise sintáctica e semântica dos documentos, origina uma representação do conhecimento, numa estrutura taxonómica Quando combinadas com ferramentas de anotação de metadados, que podem expor informações semânticas novas, escondidas noo conteúdo e conectar com outros dados relacionados, o volume de dados e a qualidade é muito superior.

Ao explorar a estrutura da taxonomia, mecanismos de inferência e consulta, a informação contida na rede é muito superior à existente nos documentos; relações novas são reveladas.

Abstract

Knowledge is the largest differentiator between companies and essential to create a competitive advantage over competitors. Nowadays access to knowledge is not hindered by physical limitations; all information is a click away, still modern systems provide poor information distribution, especially if there are no proper tools to locate and process.

As part of a global challenge to conceptualize, design and implement a proof of concept Knowledge Management System, with the support of Natural Language Processing and semantic information, following a philosophy of "Open Innovation". This dissertation focuses on all the research and the system it originates. The system is a proof of concept of a method for generating a semantic representation of a document; enhance the volume of current information and their quality, using external data and metadata to improve the existing knowledge base.

With the raise of the semantic web, a web of data with defined meaning, transverse to multiple domains, enable computers and users to work in cooperation to generate new possibilities to represent and integrate knowledge.

The syntactical and semantic analyses of the documents lead to a knowledge representation, a taxonomical structure. When combined with metadata annotation tools, which can expose new semantic information, hidden in any content and connect to related data, the data volume and quality is greatly increased.

By exploring the taxonomy structure, inference and query engines, the network information is vastly superior to the original document knowledge; new relations are revealed.

Acknowledgments

Those who provided know their importance.

Index

List of Figures

Symbols and Abreviations

AI	Artificial Inteligence
API	Application Programming Interface
DC	Document Clustering
DOM	Document Object Module
FOL	First Order Logic
IT	Information Techologies
IR	Information Retrieval
LOD	Linked Open Data
KD	Knowledge Discovery
KM	Knowledge Management
KMS	Knowledge Management System
KR	Knowledge Retrieval
NLD	Natural Language Desambiguation
NLI	Natural Language Interface
NLP	Natural Language Processing
NER	Named Entity Recognition
NET	Named Entity Type
OWL	Ontology Web Language
PDF	Portable document Format
POS	Part of Speach
Q&A	Question and Answering
RDF	Resource Description Format
RDFS	Resource Description Format Schema

SDK	Software Development Kit
SW	Semantic Web
SPARQL	SPARQL Protocol And RDF Query Language
TM	Text Mining
TREC	Text Retrieval Conference
W3C	World Wide Web Consortium
WD	Working Draft
WSD	Word Sense Desambiguation
WWW	Wold Wide Web
XML	Extensible Markup Language
XPath	XML Path Language

1 Introduction

This chapter provides a context for the tackled problem and proposes a solution. The problem understanding helps with the problem interpretation, and shines a quick look over the proposed solution.

The Section *Context* presents the background of the project. *Retail Business* offers a peek into retail and the retailer's world. *Wipro Retail* section gives a historic portrayal of the company and proficiency area. The dissertation incentive is described in *Motivation*. The coarse problem definition and objective enumeration is presented in *Problem Description and Objectives*.

Finally, *Dissertation Structure* describes the organizational structure of this document.

1.1 The Context

Developed in a business environment, over a restrictive 16 weeks length period, with a clear goal, the project combined individual and team effort to give birth to the thesis.

The thesis proponent, Wipro Portugal SA, presented the challenge of conceptualizing and implementing a Knowledge Management System (KMS) with support for Natural Language Processing (NLP), giving special focus to the study of the current state-of-the-art, both in technological and scientific approaches, while following a spirit of *Open Innovation*.

> “Open innovation is a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology” [CHW03]

Aiming to improve the research process the foment Open Innovation, incitement to create a blog[1] and use of some agile software development processes, who would reduce and improve research and experimentation times was given.

The blog was intended to disclose more than experiences, research and work. Feedback from external sources and interaction with specialists was extremely valuable. The blog is supported by the project team.

The project is developed by a research team composed of three elements; also as part of their dissertations. Despite having different dissertations, each member did a combined effort

1 http://whatisprymas.wordpress.com

into research, on the behalf of the overall project. After the whole concept and global architecture design were defined, individual research focus started. The individual research focused in differnt sets of modules.

1.2 Retail Business

The retail business can be defined as "*the sale of products or marketing services directly to final consumers*" [BDS91]. However, nowadays the focus is set in the management involved in the acquisition and provision of resources to address the customer needs [GDA93].

Business requirements are dynamic. To sustain current needs and provide a stable and continuous growth, retailers discuss different evolution processes. Most times the approaches are dissimilar, different approaches are used to satisfy customer needs, nowadays more observant and demanding, a reflection of the modern competition between retailers. To satisfy customers, the "shopping experience" takes particular important [PHM08].

The businesses and market evolution increased the processes complexity. Today, the amount of business processes, concepts and their relations is intricate to experts, but daunting to new system users.

Information technologies emerge to lend a hand. The use of information systems dedicated to the business enable Retailers on the front line to manage the market evolution and manage customer's satisfaction, creating further value, essential to the company sustainability.

1.3 Wipro Retail

Enabler was formed in 1997 from the "Direcção de Sistemas de Informação da Sonae distribuição". In few years Enabler become a multinational company, relying on their list of large retailer clients, spread across United Kingdom, Italy, Spain, Germany and Brazil.

In 2006, the Enabler was acquired by Wipro, a multinational company based in Bangalore, India. The Enabler acquisition by Wipro emerged as a way to solve the needs of retail customers. Enabler is now Wipro Retail SA, a division of Wipro Technologies for retail business solutions. The company's growth has become even more rapid after acquisition.

With the knowledge base, attained over the years in retail price management, customer analytics, global data synchronization, store management and supply chain management, Wipro Retail provides customers with methods to link business processes and information systems. Business solutions support all supply chain operations, in domains as different as food administration and fashion retail.

1.4 Motivation

*"*Knowledge is power.*"* Sir Francis Bacon [BAC97].

"All men by nature desire knowledge." Aristotle [WDR53].

The combination of both *Sir Francis Bacon* and *Aristotle* wisdom makes clear that knowledge is the power everyone covets.

Companies are not different. Among the ones who produce, collect and store more information are organizations.

Information Technology (IT) plays a vital role in modern society. Technology has found its place, reaching to all the tools and activities of modern civilization. Access to knowledge nowadays is not hindered by physical limitations; all information is a click away.

With large datasets, forecasts can be made and they are crucial to most organizations. Banks anticipate which loan applicants are likely to default, treasuries forecast tax revenues and medical researchers use bioinformatics to predict the likelihood of illness from gene sequences. Also the growing access to information has drastically changed the behaviour of markets, the course of business and private life of people, leading to new ways to make use of it.

However as Salton and McGill [SMJ86] foreseen, finding useful information would be rather difficult as information increases. Just like text information on the Web, text information in retail is growing at an astounding pace. When the information was small, it could be found based on naming conventions or organized file systems [MKO06], but the rapid growth of information in retail and the distribution across multiple platforms made it increasingly difficult to locate knowledge, which could improve in the decision-making process and improve further the costumer relation. There is a modern problem of information management.

Despite the ease to download and store most of the information, the abundance is a modern problem and the fact is that unstructured text overwhelms the user without the correct tools to locate and process the information. Furthermore knowledge is also distributed across multiple platforms and specialities.

It is unlikely we could find or think of any material that is not already documented and accessible. This has led to a great deal of interest in developing useful and efficient tools to assist users in the search [FAA03, YQG07].

Since data increases at a huge pace, to achieve a comprehensive and dynamic Information Retrieval Systems, the use of automatic methods and semi-automatic methods for knowledge enhancement, process and extraction, from multiple sources, grows in demand.

Moreover, there is a classic need for methods to consult and visualize data. Nowadays users want simple systems, with simple interfaces and meaningful results, beyond classic information query. The necessity for new means to solicit knowledge requires both direct use of information, but also use of new inference tools and natural language processing.

The above mentioned needs lead to the research and creation of this document.

1.5 Problem Description and Objectives

To Wipro Retail, information maintenance is essential for the company strategy. According to [AMS05], knowledge is the biggest differentiator between companies, and essential to create a competitive advantage over competitors.

According to results of a recent Accenture survey[2], middle managers spend up to 2 hours a day searching for information to do their jobs, and more than 50% of the information they obtain has no value to them, still more problems exist. Below presents the results of the percentage proportion of respondents:

- 59% - every day as a consequence of poor information distribution, information is missed, which might be valuable to the job, cause it exists somewhere else in the company and they just can’t find it.

2 http://newsroom.accenture.com/article_display.cfm?article_id=4483

- 42% - accidentally use the wrong information at least once a week.
- 53% - less than half of the information received is valuable .
- 45% - gathering information about what other parts of their company are doing is a big challenge, whereas only 31% said that competitor information is hard to get.
- 57% - having to go to numerous sources, just to compile information is a difficult to manage. In order to get information about competitors, customers, project responsibility or another department, respondents said they have to go to three different information sources, on average.
- 36% - there is so much information available that it takes a long time to actually find the right piece of data.

Wipro teams are generally disseminated across multiple geographic places and time zones, use multiple knowledge directories and the specialist are numerous, with diverse skills and schedules; the effort to obtain and effective share the knowledge becomes too high.

Whenever a collaborator needs to gather new knowledge, he is limited to three options:

- Keyword search in the documentation.
- Online search engine, the classic Google.
- Ask a specialist in the module.

All options have significant drawbacks.

Documents are no more than vector words, the keyword search provides good results, for a single and even for a combination of keywords (to represent a concept), but usually only the search for a single word or concept provides results, even when combining regular expressions. The number of results can range from zero to the total number of appearances in the documents, but the information is not sorted by relevance. The search system pays no attention to context and semantic relations.

On a classic search engine, when searching the document database for the keyword and concept combinations, depending on the engine aptitude, can gather a huge list of result or very small (possibly no result). If we want superior efficiency we need, besides indexing documents and ranking algorithms, methods to use the text semantics to boost results.

Of course asking a specialist, will surely generate a good answer, on the other hand, besides taking our time (like the other options) and the specialist time, we need to wait for the answer, especially if we use email for questioning or set a time to call (direct call, skype, instant messaging) and ask about the problem. It can be problematic since team can be spread across the world, and matching communications can be complicated. Furthermore it can overburden the specialist with support work, blocking an important company asset allocated to other task.

It’s especially nefarious if the information exists in physical or electronic format but the specialist is still required in support. Moreover it’s difficult to know all the system specialists and the directory were the knowledge resides.

Users want to go beyond classic search, they want to provide a question and retrieve a simple concise answer. A challenge to study of the current state of art, both in technological and scientific approaches to conceptualize and implement a KMS with support for NLP, who could do something similar rises.

The majority of Wipro information is in documents and they have no metadata, it’s going to be an uphill struggle to get better results out of the capable search technologies, without adding some metadata. When creating a document representation it’s important to include the semantic information in the text.

The NLP has techniques to analyse text, extract information and gather relations between words, allowing fragments of information to represent the text semantics, but it still only solves part of the problem.

The Semantic Web (SW) is a very recent trend and seemingly all can be represented in the semantic web, from molecules description, to business processes; options present themselves as unlimited.

According to the original vision, machine-readable metadata is kept in a format helpful to machines and automatic information gathering agents. Some systems already provide information in this format, making tempting the creation of mashups[3] to integrate the interesting parts in the system and make the most of it.

The research presented in this dissertations document aims to create a semantic representation of documents. To improve the representation, enhancement and deduction of additional information is essential. It aims to improve question and answering (QA) systems results.

The time constrains put a strain in the objectives, but results are essential, a proof of concept is required.

To achieve the objectives and make the most of the time, it is necessary to resort to a rich set of knowledge areas and use community open standards and make use of useful agile software development techniques. Furthermore following the Open Innovation idea and the need to collaborate in work, and the business environment require extra effort.

1.6 Dissertation Structure

The objective of this work, as a part of a larger KM system developed by Wipro, is grounded in the research and creation of a proof of concept mechanism to generate a document representation, with semantic knowledge, to improve the QA system results and solve Wipro's knowledge access problems.

To improve the representation, enhancement and deduction of additional information is essential. Our proposal is to resort to a rich and diverse knowledge areas and use community open standards.

The dissertation innovative format and relations is also a novelty factor to analyse and experiment.

The following chapter makes a synthesis of the essential concepts required to automating the process of gathering and structuring information in order to create a knowledge representation. A complementary focus is also given to methods that enhance knowledge and improve inference in the proof of concept.

A description of related work, previously done and essential to the current implementation is given in Chapter 3. A broad description of the tools and methods is given, together with reasons who led to the preference. Chapter 4 makes a deeper description of the proposed system architecture and result expectations, followed by the methodology taken and a complete rundown of the development environment. The proposed system implementation is given in Chapter 5. A practical example, some tests and result examination is provided in Chapter 6. A discussion of the dissertation, implementation results and expectations is given on Chapter 7.

3 Applications that take data, usually Web service data, usually from more than one source.

2 Relevant Technologies

To better understand the work, an introduction to the research field and related concepts is essential.

Section 2.1 gives an overview of the Semantic Web. Section 2.2 gives the objectives and expectations, with a deeper view of the SW architecture and layers (Section 2.2.2). In Subsection 2.2.3, the concept of ontologies is discussed, followed by a list of SW languages (Section 2.2.4).

OWL language is the knowledge representation language used in our project and is discussed more deeply in section 2.3, with information about the building blocks classes (Section 2.3.1), properties (Section 2.3.2) and individuals (Section 2.3.3). A view into OWL query language is also given (Section 2.3.4).

The world of named entity recognition (Section 2.4) is an intricate field in NLP. It faces many problems like word sense disambiguation (Section 2.4.1), co-reference and alias resolution (Section 2.4.2).

Built side by side with the semantic web, linked open data (Section 2.5) presents a lot of potentialities when the addition of knowledge is important.

A short resume and conclusion about the technologies and relevance to the work is provided in Section 2.6.

2.1 Semantic Web

The *Semantic Web is a Web of data*. Nine years ago in Scientific American magazine, Tim Berners-Lee, James Hendler and Ora Lassila unveiled a nascent vision of the Semantic Web; a highly interconnected network of data that could be easily accessed and understood by automatic processing tools [TBL01].

> "*The Semantic Web is a vision: the idea of having data on the Web defined and linked in such a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications."* in Semantic Web Activity Statement[4]

4 http://www.w3.org/2001/sw/Activity

They illustrated a future with intelligent software agents that would head out on the World Wide Web and automatically book flights, hotels or trips, update medical records and give a customized answer to a particular question, without having to search for personal information or explore through results. The Semantic Web is a vision of information that is understandable by computers.

They also proposed the technologies required to make the vision come true, the 4 building blocks of the Semantic Web:

- Resource Description Format data model
- Ontology languages
- Inference engines
- Support technologies

Sceptics have said the Semantic Web would be too difficult for people to understand and exploit. Not so; technologies have come a long way. A large and active community of early adopters provided a steady output of tools for exploiting Semantic Web. Large companies and research facilities have ongoing major projects that have greatly improve the existing tools and improve common operation and scientific research [DLS07].

Companies are also using the Semantic Web to enhance business-to-business and business-to-costumer interactions [DHY08, RDO05], enhancing data-processing structures, and backend, with new services. And like an iceberg, the tip of this large body of work is emerging in direct consumer applications, too. Examples of application already using it are:

- Dbpedia - Published structured data from wikipedia.
- Good relations - Standardized language e-comerce application, adopted by BestBuy, Yahoo, OpenLink Software, O'Reilly Media and others.
- Linked Open Data - W3C sponsored effort to create openly accessible, and interlinked, data on the Web.
- OpenPsi - Linked data service that supports research.
- Umbel - Concept reference structure for the Web.

There is already hundreds of components that can be used in creation or extention of semantic web applications.

2.2 Semantic Web Objectives

> "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [MKB09]

The Semantic Web is not a separate entity from the World Wide Web, it is simply a component of it. It associates meaning with data, extending documents into further data, it describes methods and technologies to allow machines to understand the meaning also referred as "semantics".

New standards and technologies provide automated environments for machine and databases processing, taking away some of the burdens currently faced and access the Web more intelligently way.

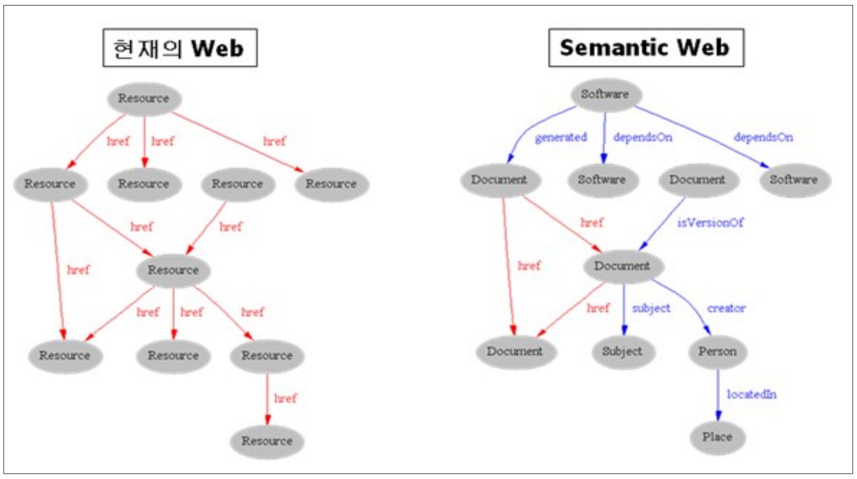


Figure 2.1: Existing web and Semantic Web[5].

Even though the SW is a Web of data (Figure 2.1), it is above all intended for creation of automatic content for humans. The agents would be able to perform tasks automatically and locate related information on behalf of the user. The work focus is shifted into delivering better services and experiences.

Semantic publishing refers to produce new information, to enhance the meaning of a document. It aims to add metadata to document, allowing computers to understand the structure and even the meaning of the published information

Semantic publishing on the Web or semantic web publishing refers to publishing information on the web as documents accompanied by semantic markup. Currently there are two main approaches to semantic publishing:

- Formal markup, like microformats[6], who use custom HTML/XHTML tags to convey metadata.
- Ontologies and vocabularies, to create domain specific information and publish as data objects

Providing support to current KD is one of the main goals and ontologies and vocabularies, they present good list of data sources and support tools. The support to knowledge discovery and integration, with support for automated tools can lead to the creation of outstanding products, services and experiences.

The Semantic Web has the potential to improve human information access to unstructured and semi-structured information. One of most sought fields is automatic processing of textual document. With the integration of semantic web tools new ways to deal information are expected, like:

- Enhancement – Data integration from multiple external sources.
- Search – Go beyond keyword search, into semantic search.
- Publication – New formats to encoding descriptive, administrative, and structural metadata
- Presentation – Different publication format require new visualization formats.
- Share - New ways to distribute information.

5 http://semanticwiki-en.saltlux.com/index.php/Semantic%20Web

6 http://microformats.org/wiki/existing-classes

Searches should be more precise, data more complete, within context and less prone to human error. The quality of information should go beyond what is available.

Nowadays a large number of data is already presented in Semantic Web format, including documents. Still the majority of textual data (produced nowadays, older and in physical format) is unstructured.

If semantic web is to surpass the current size of the web, all available current data need to be annotated, metadata needs to be included, leading to the need of semantic publishing. The transformation of unstructured data presents itself as an excellent way to create new and unique data to integrate with the semantic web format. In short term only domain specific documents are relevant, but long term, any kind of document need to be supported.

The Semantic Web will enable machines to comprehend documents.

2.3 Open Semantic Enterprise

Suitable business applications include data federation, data warehousing, search engine, enterprise information integration, business intelligence, competitive intelligence, knowledge representation, and so forth. They aim to reduce risks and costs for business, while improving deployment speed and responsiveness.

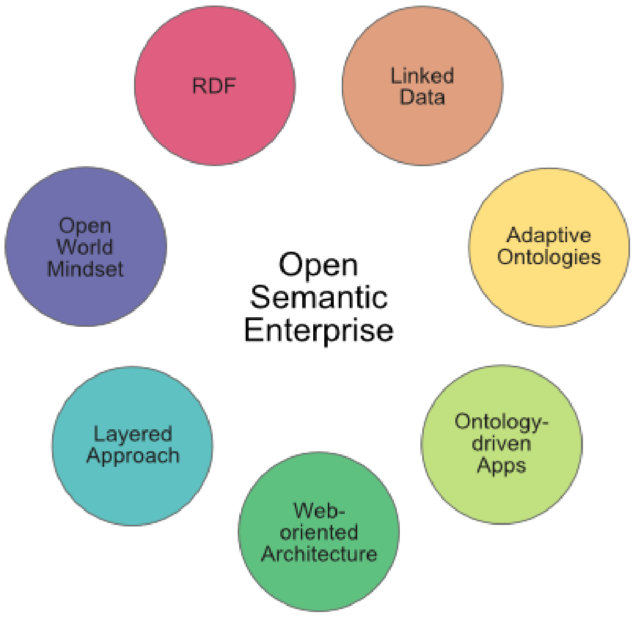


Figure 2.2: Foundations for the Open Semantic Enterprise.

The integration of domain knowledge and semantic technologies don’t require discarding good practices, replacing current systems or assets. It can be applied equally to public or proprietary information. It can be deployed incrementally at low risk and cost; RDF, RDFS, OWL, SPARQL and others can be integrate into existing information assets, using the best practices of linked data and the open world assumption, and targeting knowledge management applications. Results will appear even early on.

- The integration of domain knowledge and semantic technologies can assure:
- Incremental analysis and refinement of domain representations.

- Flexible and robust frameworks, with constant infusion of new information.
- Data with partial characterizations can be combined with other data having partial or complete characterizations
- The data and the structures can be used and expressed in multiple building blocks.
- Public and private information can be combined.

By providing successful web architectures, users can create loosely coupled, distributed systems that can grow and interoperate in a decentralized manner, the perfect architectures for flexible collaboration systems and networks.

As stated by Mike Bergman (Figure 2.2), the benefits rise from a combination of technologies and interactions between them. Let's take a look at them:

- **RDF Data Model** – Used to structure content, it is possible to create more expressive vocabularies within RDF, making it a powerful data model and language for data federation and interoperability across disparate datasets.
- **Linked Data Techniques** – Set of best practices[7] for publishing and deploying instance and class data using the RDF data model, useful to both humans and machine agents. Linked data is applicable to public or enterprise data, open or proprietary.
- **Ontologies** - The guiding structures for how information is interrelated and made coherent using RDF and its related schema and ontology vocabularies, RDFS and OWL. A large number already exists, ranging from generic, to domain oriented. The reuse of existing ontologies is encouraged.
- **Ontology-driven Applications** - modular, generic software applications designed to operate in accordance with the specifications contained in an adaptive ontology. The design limits software frailty and maximizes software re-use. It also reduces creation and maintenance effort.
- **Web-oriented Architecture** – Provide industry backed methods for linking documents. Web-oriented architecture is a subset of the service-oriented architectural and integrates diverse disparate applications, which can be used within multiple business domains and platforms.
- **Layered Approach** - Adaptive layer is an interoperable stack, following industry standards. Semantics help to bridge and communicate across multiple existing systems and schema. The layered approach allows incremental improvement and separation between ontologies and assets.
- **The Open World Mindset** – Companies are hostage of closed world assumption, very fit to transaction and operational systems, but not supportable by real circumstances. While transactions require completeness and performance; insight requires drawing connections in the face of incompleteness or unknowns. Open World Mindset doesn't assure it, but thresholds on information and understanding aren't affordable or achievable with traditional, closed-world approaches.

These practices do not require replacing current systems and provide a significant initiative which can lead to potentially huge benefits, with manageable risks and costs.

7 http://www.mkbergman.com/index.php#ose8

2.3.1 Semantic Web layer

The *Semantic Web Stack* is an illustration of the hierarchy of languages, where each layer exploits and uses capabilities of the layers below. It shows how standard Semantic Web technologies are organized to make the Semantic Web possible.

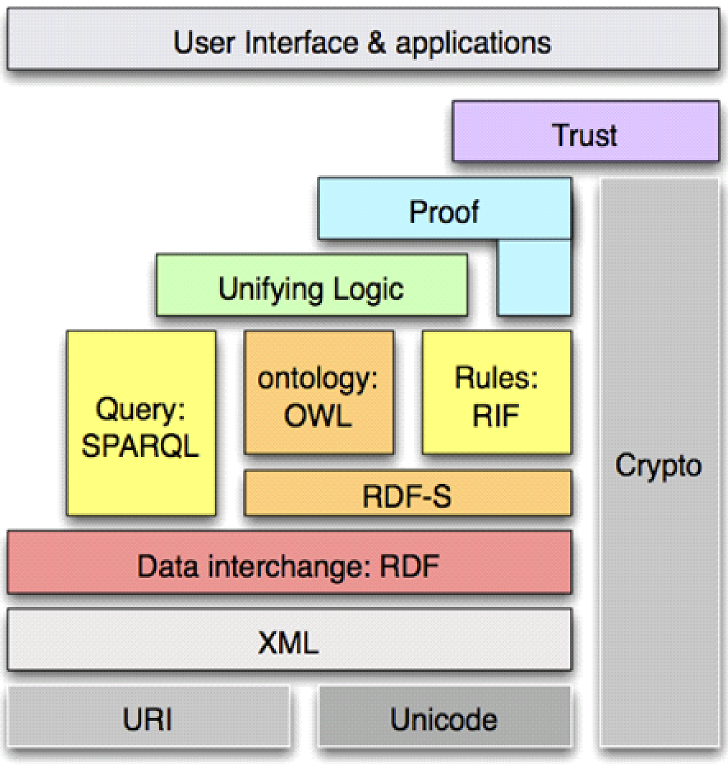


Figure 2.3: Semantic Web Stack.

As shown in Figure 2.3, the lower layer, URI and Unicode, follows the important features of the existing WWW. Unicode is a standard of encoding international character sets and it allows representation and manipulation of text in many languages. Semantic Web should also help to bridge documents in different human languages.

Uniform Resource Identifier (URI) provides means for uniquely identify resources[8] (e.g., documents). An international variant, the Internationalized Resource Identifier (IRI) allows usage of Unicode characters in identifier and for which a mapping to URI is defined. URI is a subset of IRI. Semantic Web needs unique identification to allow provable manipulation with resources in the top layers.

Extensible Markup Language (XML) layer with XML namespace and XML schema definitions makes sure that there is a common syntax used in the semantic web. XML enables creation of documents composed of structured data. A XML document contains elements that can be nested and that may have attributes and content. XML namespaces allow specifying different markup vocabularies in one XML document. XML schema serves for expressing schema of a particular set of XML documents.

A core data representation format for semantic web is Resource Description Framework (RDF). RDF is a framework for representing information about resources in a graph form, so-called triples. It was primarily intended for representing metadata about WWW resources, such

8 A Resource is anything that can have a URI; this includes dividual elements of an XML document

as the title, author, and modification date of a Web page, but it can be used for storing any other data. All data in the semantic web use RDF as the primary representation language. The normative syntax for serializing RDF is XML in the RDF/XML form. Formal semantics of RDF is defined as well.

RDFS[9] has a collection of controlled vocabulary terms organized into a hierarchical structure and can be can be used to describe taxonomies of classes and properties to create lightweight ontologies. RDF Schema (RDFS) was created together with its formal semantics within RDF.

More detailed ontologies can be created with Web Ontology Language (OWL). The OWL is a language derived from description logics, and offers more constructs over RDFS.

RDFS and OWL don’t provide enough rules to fully support description logic. RIF and SWRL are rule languages, being standardized for the semantic web to bring rule support.

For querying RDF, RDFS and OWL, a Simple Protocol and RDF Query Language (SPARQL) is available. SPARQL is SQL-like language, but uses RDF triples and resources for both matching part of the query and for returning results of the query. It is also a protocol for accessing RDF data.

Unifying Logic and Proof layers are undergoing active research. It is expected that all the semantics and rules will be executed at the layers below Proof and the result will be used to prove deductions.

On top of these layers, application with user interface can be built.

2.3.2 Ontologies

Ontology has been attracting a lot of attention recently, while research started in the 90’s, it accelerated due to recent efforts to extend the capabilities of the World Wide Web through the addition of formal semantics.

One of the key concepts is *conceptualisation*, the AI term defined as:

> “*a set of objects which the observer thinks exists in the world of interest and relations between them”* [GNN87].

So ontology can be seen as a "*formal, explicit specification of a shared conceptualisation*" [TGR93]. Ontologies are engineering artefacts that can formally represent the concepts and their relationships within the given knowledge domain, while supporting automatic processing.

Individuals or groups may want to define terms and data they frequently use, as well as the relations among those items [FAF02]. This bag of definitions is called ontology. To specify the conceptualisation, common components of ontologies include:

- Classes: represent general concepts in the domain.
- Properties or attributes of the objects described.
- Relation between objects.

Ontologies are used by people to exchange information about a given domain or knowledge area (medicine, biology, pharmacy). Some of the practical benefits:

- Make explicit the scope, definition, and language and semantics of a given domain or world view.

9 http://www.w3.org/TR/rdf-schema

- Generalize about the domains .
- When hierarchically structured in part, can provide the power of inheritance.
- Mechanisms to reason or infer over its domain.
- Structured and controlled vocabularies, helpful for disambiguating context.

Ontologies facilitate information search and integration of data from different communities, because they provide a common basis that ensures consistency of data.

Ontologies can be classified into 2 groups. Upper ontologies (also known as foundation ontology or adaptive ontology), assures semantic interoperability between a large numbers of ontologies, without detailing concepts [WMA04]. The domain ontology models a specific domain, its complement to upper ontologies and structures the information driving the applications [RPV04].

Recently, ontologies have found their way into higher-level information fusion where they provide means to combine heterogeneous data sources, to maximize knowledge sharing. Thoughts on how to improve and provide better services are already available. Let's see some examples:

- Web Portals – Information is optimized for human readability, with special focus on aesthetic concerns. If the portal provides methods for automatic representation with ontologies, it opens the door to automatic external contributions and enrichment of information.
- Multimedia – Nowadays multimedia content is tagged, but the tagged information provides little semantics, relations between content are feeble. If ontologies are used, besides the original information, additional cues can be provided, and automatic agents can add overtime more related data.
- Services – Creation of mashups have provided good results. Mashups are applications that take data, usually Web service data, usually from more than one source, and uses it to create something new. From collecting friend's messages, to business forecast systems, the potentialities are enormous.

  With the power to create taxonomies, the expressive power

2.3.3 Languages

Ontolingua[10] is the original language for ontology representation and sharing. Create by Knowledge System Lab (KSL) at Stanford University, it is designed by adding a frame like representation to the Knowledge Interchange Format[11] (KIF), a computer-oriented language for the interchange of knowledge among disparate programs.

10 http://www.ksl.stanford.edu/software/ontolingua/

11 http://www-ksl.stanford.edu/knowledge-sharing/kif/

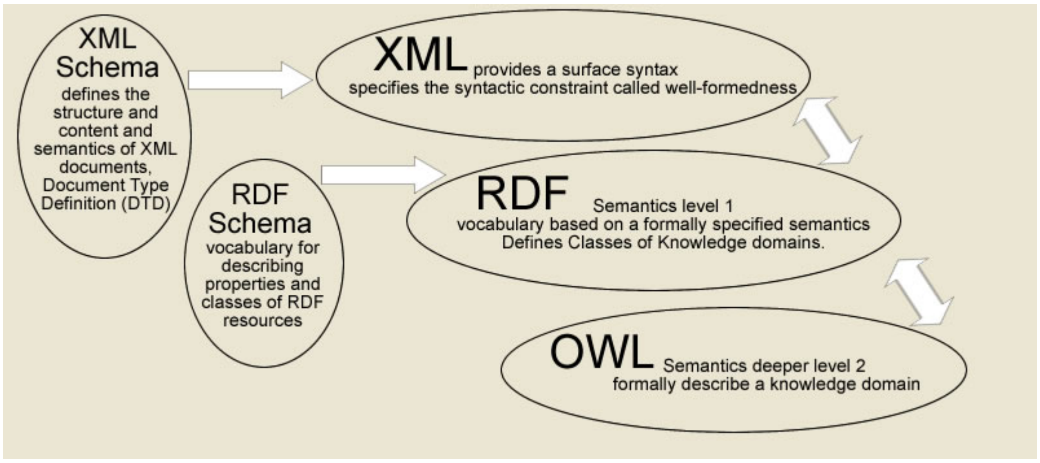

XML Schema
defines the structure and content and semantics of XML documents, Document Type Definition (DTD)
XML provides a surface syntax
specifies the syntactic constraint called well-formedness
RDF Schema
vocabulary for describing properties and classes of RDF resources
RDF Semantics level 1
vocabulary based on a formally specified semantics
Defines Classes of Knowledge domains.
OWL Semantics deeper level 2
formally describe a knowledge domain

Figure 2.4:Semantic Languages Layers[12].

With the emergence of the Semantic Web as a broad, commercial platform, new encoding tools for semantic information in documents were fashioned.

Following is a description of the major languages (Figure 2.4) that have marked the evolution of the Semantic Web.

RDF

"A framework for describhttp://www.w3.org/TR/REC-rdf-syntax/ing and interchanging metadata " [BRT98]

Developed by the WWW Consortium (W3C), the language Resource Description Framework[13] (RDF), the first draft was published in October 1997 and achieved W3C recommendation status in February 2009.

Resource Description Framework (RDF) is a framework for representing information about resources in a graph form. RDF has XML based syntax, and resembles XML markup[14], but it's more than a language; it's a data representation model [15].

RDF is carefully designed to have the following characteristics:

- Scalability
- PropertyTypes are Resources
- Independence
- Interchange
- Values Can Be Resources
- Properties Can Be Resources

RDF is a standardised *top-level* XML, that has metadata and elements follow domain-specific schemata. [16].

12 http://instructionaldesign.com.au/Academic/TechnicalTheme1.htm

13 http://www.w3.org/TR/rdf-mt/

14 http://www.w3.org/TR/REC-xml/

15 http://www.w3.org/TR/rdf-concepts/

16 http://www.w3.org/TR/REC-rdf-syntax/

By generalizing the concept of resource (web resource), metadata can represent document information like author or title or any other information available. The mechanism for doing this with RDF is to construct an **RDF triple** which consists of a subject (the resource), property and object (its value).

Since it was primarily intended for representing metadata about WWW resources, it is built around resources with URI.

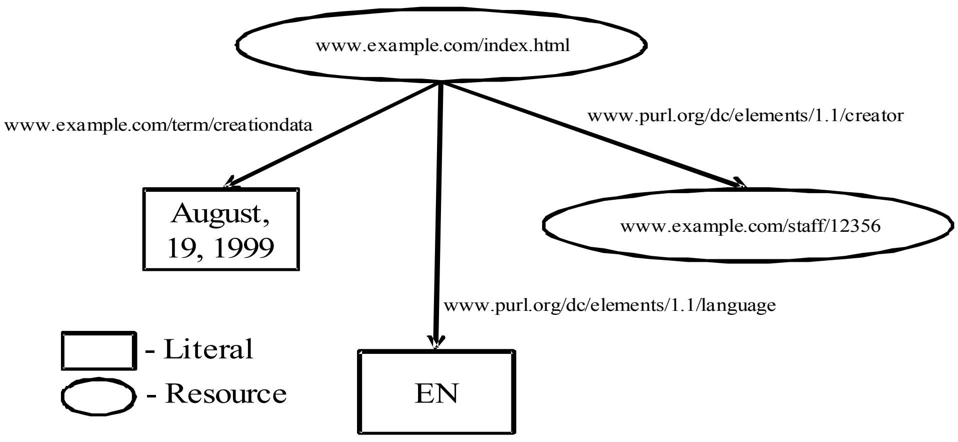

www.example.com/index.html
www.example.com/term/creationdata
www.purl.org/dc/elements/1.1/creator
August, 19, 1999
www.example.com/staff/12356
www.purl.org/dc/elements/1.1/language
- Literal
- Resource
EN

Figure 2.5: RDF triple model.

Information is represented by the AI well known triplet model (Figure 2.5), a <resource, attribute, value> tuple. All elements of this triple are resources by default, defined by a unique URI. The last element, value; it can be also a literal.

Resources and Value are nodes; attribute is a link between nodes. They are the base of the semantic network.

Literal in the RDF sense is a constant string value such as string or number. Literals can be either plain literals (without type) or typed literals typed using XML Datatypes [17].

Lets check the string "Celso Costa", the document author, the markup is:

```
<author>Celso Costa</author>
```

Its markup is clear enough to informe that "Celso Costa" is the author. But lets say the document is available at "*http://www.example.org/crc.pdf*", the RDF description would be:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#>
   ...
<rdf:description rdf:about="http://www.example.org/crc.pdf">
            <author>Celso Costa</author>
   </rdf:description>
   ...
</rdf:RDF>
```

The graph representation would be:

17 http://www.w3.org/TR/xmlschema-2

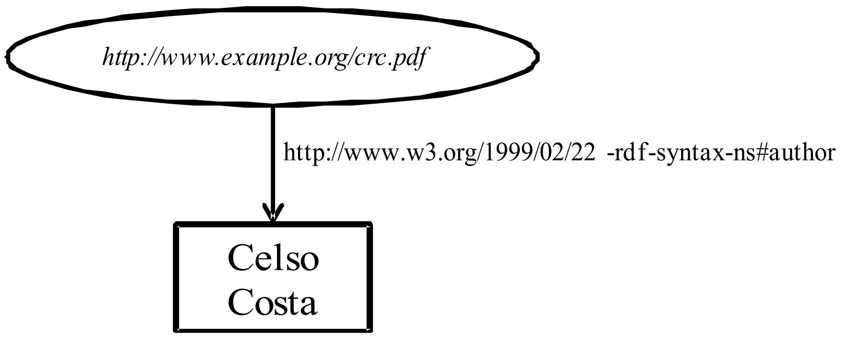

http://www.example.org/crc.pdf
http://www.w3.org/1999/02/22 -rdf-syntax-ns#author
Celso Costa

Figure 2.6: RDF Graph representation.

The first line of the xml code identifies the xml version. The default namespace is in the second line and is used to identify the author, about, rdf and description. The resource “crc.pdf“, with URI http://www.example.org/ has author[18] “Celso Costa”. A visual representation is provided in Figure 2.6.

RDFS

RDF schema (RDFS) extends RDF vocabulary to allow describing taxonomies of classes and properties.

RDFS has built-in classes and meta-classes (classes with other classes as members), by which users can define new classes and relations to define resources and objects. The resources and objects can be further characterized, with properties like domain and range of properties, relations of the RDF classes and taxonomic properties using the XML vocabulary.

The RDFS vocabulary is uses a XML namespace with code RDFS:.

Taking previous example, the property “author” could be further characterizing in the following way:

```
<rdfs:Property ID="author" >
<rdfs:label>Author</rdfs:label>

<rdfs:subPropertyOf
rdf:resource="http://purl.org/dc/elements/1.1/dcmes.rdf#Creator"/>

<rdfs:domain rdf:resource= "#Person"/>

<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>

</rdfs:Property>
```

The RDF declaration *label* element provides a human-readable version of a resource's name. The type hierarchy is defined using *subPropertyOf* element. The *domain* constraint specifies the attachment of properties to classes and the *range* constraint can be used to indicate the classes that the values of a property must be members of.

Topic Maps

Topic Maps is a standard for knowledge integration developed by the ISO's SGML working Group in 1996. The specification was developed had different purposes than RDF,

18 As defined in Dublin Core

however, the results turned out to have a lot in common and this has led to calls for their unification[19].

Besides semantic networks [SBA91] and conceptual graphs [SJF84], topic maps are one candidate for the task of information encoding. The methods attempt similar ways to connect pieces of data into a graph which represents the relationships between them. These methods do not attempt to enforce a rigid structure on the information they describe but rather they provide a lightweight way of navigating the information which exists in separately maintained data sources.

One potential application of these technologies would be to facilitate translating the knowledge provided by one source into a form that can be used elsewhere, much like semantic web.

OWL

Web ontology language (OWL) is a language developed by w3c (reference). It's a common language for ontology representation based on DAML+OIL (reference), based on the design and experience achieved from its use. OWL is an extension of RDF schema (and RDF of course) and also employs the triple model.

It was designed to enable Semantic Web and extensibility, modifiability and interoperability were the highest priorities, while trying to keep scalability and expressive power.

A more complete description can be found in Section 2.3.

2.4 Ontology Web Language

In 2000, DARPA [20]started development of DAML[LLW05]. In March 2001[21], the *"Joint US/EU ad hoc Agent Markup Language Committee"* decided that DAML should be merged with OIL [LLW05]. DAML+OIL was intended to be a thin layer above RDFS, with formal semantics(study of the semantics, or interpretations, of formal and also natural languages) based on a Description Logic (DL) [IPP03] . OWL started as a research-based[22] revision of DAML+OIL aimed at the semantic web.

The Web Ontology Language OWL extends RDF and RDFS. The primary aim is to bring the expressive and reasoning power of description logic to the semantic web. Unfortunately, not everything from RDF can be expressed in DL. For example, the classes of classes are not permitted in the DL, and some of the triple terms would have no sense in DL. That is why OWL can be only syntactic extension of RDF/RDFS. To partially overcome this problem, and also to allow layering within OWL, three species of OWL are defined.

- *OWL Lite* can be used to express basic taxonomy and constraints, such as 0 and 1 cardinality. It is the simplest OWL language which corresponds to description logic *SHIF*.
- *OWL DL* supports maximum expressiveness while retaining computational completeness and decidability. The DL in the name shows that it is intended to

19 http://www.w3.org/TR/2005/WD-rdftm-survey-20050329

20 Defense Advanced Research Projects Agency.

21 http://www.daml.org/2001/03/daml+oil-index

22 http://www.w3.org/TR/2002/WD-owl-features-20020729/

support description logic capabilities. OWL DL corresponds to description logic *SHOIN*.

- *OWL Full* is the complete OWL language and has no expressiveness constraints, but also does not guarantee any computational properties. It is formed by the full OWL vocabulary, but does not no impose any syntactic constrains, so that the full syntactic freedom of RDF can be used.

Every legal OWL Lite ontology is a legal OWL DL ontology, every legal OWL DL ontology is a legal OWL Full ontology. Furthermore, validity OWL Lite conclusions are the same for OWL DL conclusions, all featured in OWL Full conclusions. The inverses of these relations generally do not hold. The chosen language takes into account the problem requirerments.

Every OWL ontology is a valid RDF document (DL expressions are mapped to triples), but not all RDF documents are valid OWL Lite or OWL DL documents.

2.4.1 OWL Classes

Classes are the basic building blocks of an OWL ontology. A class is a concept in a domain. Classes usually constitute a taxonomic hierarchy.

OWL classes have associated a sets contain individuals, called the *class extension*. The individuals in the class extension are called the *instances* of the class.

A class has an intentional meaning, a concept manifestation, which is related but not equal to its class extension. Thus, two classes may have the same class extension, but still be different classes.

All ontologies classes start with the namespace definition of the syntax used. For OWL, the prefix is "*owl:*". Classes are defined with the prefix "*owl:Class*".

The owl:Thing is the root class. Every individual in the ontology is a member of the class owl:Thing, thus each class is implicitly a subclass of it.

The property subClassOf (*rdfs:subClassOf*), declares that the class derives from another and in OWL declares class hierarchies. Multiple inheritance is possible.

```
<owl:Class rdf:ID="CarMidSize">
        <rdfs:subClassOf rdf:resource="CarShape"/>
<owl:Class/>
```

The example creates the class "*CarMidSize*" as subclass of "*CarShape*".

2.4.2 OWL Properties

Properties are binary relations on individuals[23]. OWL distinguishes between two main categories of properties that an ontology builder may want to define:

- *Object properties* relates individuals to individuals.
- *Datatype properties* relates individuals to datatype values. OWL uses XML schema to define datatypes

The presented code gives information about the car manufacturer and its horsepower.

23 instances of properties linking individuals.

```
<owl:ObjectProperty rdf:ID="manufacturer">
<rdfs:domain rdf:resource="# CarMidSize " />
<rdfs:range rdf:resource="# CarBrand " />
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="horsepower">
<rdfs:domain rdf:resource="# CarMidSize " />
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

The 2 properties are different, shown by the fields *rdfs:domain* and *rdfs:range*. The first sample is a object property and relates relate one instance "*CarMidSize*" with an instance of "*CarBrand*", while the second is a datatype property and relates a class instance with a string.

OWL allows the meaning of properties, enhancing reasoning[24] through use of property characteristics. The characteristics of OWL Lite[25] can be *transitiveProperty, SymmetricProperty, FunctionalProperty, inverseOf* and *InverseFunctionalProperty*[26].

2.4.3 OWL Individual

Individuals, are instance of the classes, represent objects in the domain of discourse. Properties can be used to relate one individual to another.

```
<owl:thing rdf:ID="Charger">
   <rdf:type rdf:resource=" CarMidSize"/>
   < manufacturer rdf:resource="Dodge"/>
   < horsepower rdf:datatype="&xsd;string">325HP</ horsepower >
</owl: thing >
```

This represent the individual "Charger", who is "CarMidSize" instance, has manufacturer "Dodge" and horsepower "325HP".

2.5 Ontology Inference

The OWL language is rooted in description logic, a family of knowledge representation languages designed for encoding knowledge about concepts and concept hierarchies. An OWL inference engine's core responsibilities are to follow the formal semantics while processing information encoded in OWL, discover inconsistencies and derive new information from known information [YTH04].

A simple example demonstrates the power of inference: Joe is visiting San Francisco and wants to find an Italian restaurant in his vicinity. His wireless PDA tries to satisfy his desire by searching for a thing of type *restaurant* with a *cuisineType* property with the value *Italian*. The goodPizza restaurant advertises its cuisine type as *Pizza*. These cannot be matched as keywords or even using a thesaurus, since *Italian* and *Pizza* are not equivalent in all contexts. The restaurant ontology makes things clearer: *Pizza rdfs:SubClassOf ItalianCuisine*. By using an

24 More restriction and constrains lead to better reasoning.

25 http://www.w3.org/TR/2004/REC-owl-features-20040210/#s2.1

26 http://www.w3.org/TR/2004/REC-owl-guide-20040210/#PropertyCharacteristics)

inference engine, Joe's PDA can successfully determine that the restaurant goodPizza is what he is looking for. An inference engine for OWL language is designed to accomplish this task.

An inference engine is required for the processing of the knowledge encoded in the semantic web language OWL. An OWL inference engine should have following features:

- **Checking ontology consistency -** The ontology imposes a set of restrictions on the model graph. The OWL inference Engine should check the syntax and usage of the OWL terms and ensure that the OWL instances meet all of the restrictions.
- **Computing entailments -** Entailment, also known as logic implications are used to control the inference tasks for an OWL inference engine. They should provide a convenient interface to process rules that involve OWL classes, properties and instance data.
- **Processing queries -** OWL inference engines need powerful, yet easy-to-use, language to support queries, both from human users and software components.
- **Handling XML data types -** An OWL inference Engine should be able to test the satisfiability of XML data types, such as integers, floating point numbers, strings and complex types.

Description Logics are generally given a semantics that make them subsets of first-order logic. Therefore, several different approaches based on those logics have been used to design OWL inference engines:

- **Using a specialized description logic reasoned - Since** OWL is rooted in description logic, it is not surprising that DL reasoners are the most widely used tools for OWL reasoning. DL reasoners are used to specify the terminological hierarchy and support subsumption. It has the advantage of being decidable.
- **Using full first order logic (FOL) theorem tester.** OWL statements can be easily translated into FOL, enabling one to use existing FOL automated theorem test to do the inference.
- **Using a reasoner designed for a FOL subset.** A fragment of FOL and general logic based inference engine can also be used to design the OWL inference engine.

2.6 Sparql

SPARQL Protocol And RDF Query Language (SPARQL) for data stored natively as RDF or viewed as RDF via middleware.

Built upon rdfDB, RDQL, and SeRQL, SPARQL has several valuable new features of its own. It is designed to meet the use cases and requirements identified by the RDF Data Access Working Group[27]. On 15 January 2008, SPARQL became an official W3C Recommendation[28].

Even the simplest dialect of OWL, OWL Lite, is a DL without algorithms allowing for efficient entailment[29] and query answering over knowledge bases (KB) scaled to millions of facts (triples). SPARQL allows globally unambiguous queries using triple patterns, conjunctions, disjunctions, and patterns. Like SQL for conventional databases, SPARQL is now emerging as a leading query framework for RDF based "triplestores[30]", allowing globally unambiguous queries using triple patterns, conjunctions, disjunctions, and patterns.

27 www.w3.org/2009/sparql/wiki

28 http://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation

29 Logic implication.

30 Form of database.

Moreover, in keeping with the distributed nature of the Internet, distributed SPARQL "*endpoints*" are emerging, which represent specific data query points at IP nodes, the results of which can then be federated and combined. Every day new endpoints emerge to support integration models that can either follow the data federation[31] approach or the consolidated data warehouse[32] approach or combination of both.

The answer to a SPARQL query depends on the ontology entailment regime and the vocabulary from which the resulting answers are taken.

The first version of SPARQL was defined only for simple entailment[33], defining a set of conditions the result need to meet. Different entailment regimes can lead to different sets and more complex inferences. The current entailment regimes are[34]:

- RDF Entailment
- RDFS Entailment
- OWL RL Entailment
- OWL Full Entailment
- OWL 2 DL, EL, and QL Entailment
- RIF Entailment

With SPARQL and correct exploitation of entailment regimes, besides the existing, new knowledge can be derived. The new knowledge is dependent on the ontology taxonomy and instances relations.

2.7 Linked Open Data

Integrating large dissimilar data sources in a large corporation is expensive, but the use of Semantic Web can significantly reduce costs[35].

In 2006 Tim Berners-Lee wrote the following seminal linked data design note [TBL06]:

"*The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data. With linked data, when you have some of it, you can find other, related, data.*"

He proposes the philosophy of web of document in data. The Web enables us to link related documents. Similarly it enables us to link related data. Linked Data refers to a set of best practices for publishing and connecting structured data on the Web, four linked data principles are stated in Tim Berners-Lee note:

- All items should be identified using URI.
- All URI's should be dereferenceable, using HTTP URI[36]

31 Search of multiple online databases or web resources.

32 Dimensional approach or the normalized storage of data.

33 http://www.w3.org/TR/rdf-mt/#entail

34 http://www.w3.org/TR/2009/WD-sparql11-entailment-20091022/#d4e619

35 http://www.w3.org/DesignIssues/Business.html

36 http://www.w3.org/Protocols/rfc2616/rfc2616.html

- When someone looks up a URI (rdf property is interpreted as hyperlink), return useful information.
- Links to other URIs should be included, to enable discovery of more knowledge.

In 2009 Tim Berners-Lee modernized the concept, he stated that data lookup should be done “using the standards (RDF, SPARQL)”. The discussion about the inclusion of RDF in core principles is ongoing[37].

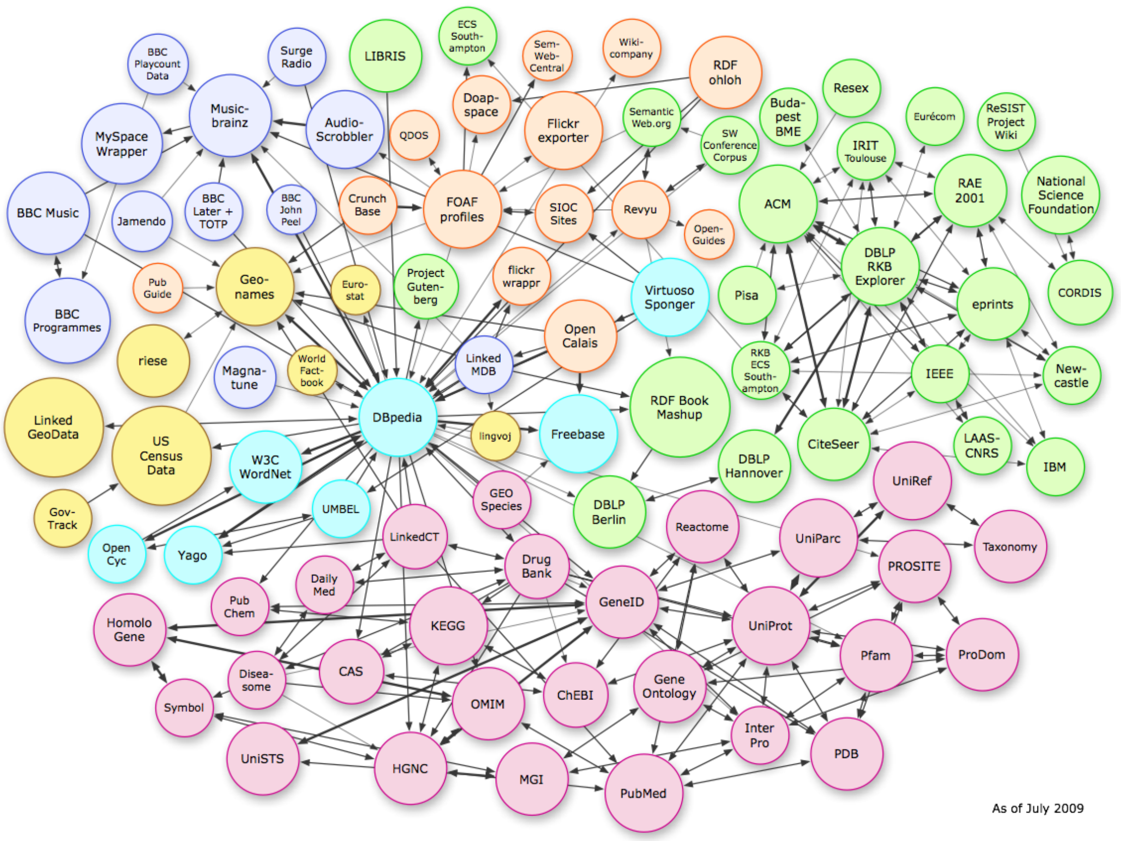


Figure 2.7: Linked Open Data as in July 2009.

The data can be accessed using Linked Data browsers, just as the traditional Web of documents is accessed using HTML browsers. However, instead of following links between HTML pages, Linked Data browsers enable users to navigate between different data sources by following RDF links. This allows the user to start with one data source and then move through a potentially endless Web of data sources connected by RDF links. Data sources can be:

- More easily crawled by search engines.
- Accessed using generic data browsers.
- Enables links between data from different data sources.

From 2007 on, things advance at a sound pace in the Linked data initiative. *Chris Bizer* and *Richard Cyganiak* launched the *Linked Open Data Community Project*[38] and W3C standards to support LOD were released.

37 http://cloudofdata.com/2009/07/does-linked-data-need-rdf/

38 http://linkeddata.org

The results, as of July 2009 (Figure 2.7), show the data sets that have been published and interlinked by the project so far. Collectively, the data sets consist of over 13.1 billion RDF triples[39], which are interlinked by around 142 million RDF links[40].

Mashup are Web application that pulls data from different sources. Mashups are fairly common these days, Mashupfeed.com lists 4674 existing mashups, with 2.88 being added per day. But each of these mashups is fairly static as far as the types of information they can present. LOD gives control over a multitude of dynamic data.

2.8 Named Entity Recognition

The Named Recognition Field (NER) field was coined at the sixth Message Understanding Conference (MUC-6) [MUC95]. It has emerged as an important step for many natural language applications. When defining IE tasks, its common the extraction of entities mentions (Figure 2.8), such as person names, locations, dates, numeric expressions, specialized terms and product terminology to integrate in the model and empower organizations [CSS92]. Noteworthy research was conducted by extracting proper names from texts.

[Named Entity Type, Named Entity]
"[Country Germany]'s representative to the [Organization European Union]'s veterinary committee [Person Werner Zwingman] validated on [Location France] report …"

Figure 2.8:Named Entity Recognition.

Saul Kripke ´s work Naming and Necessity [KRP82], in the expression "Named Entity", the word "Named" refers to entities that refer one or more rigid designators. A rigid designator designates the same object in all possible worlds [FEB82]. Rigid designators include proper names as well as certain natural terms, such as biological species and substances. They are used when identifying mentions of entities.

One of the first works was done by Lisa F. Rau [RLF91] at the 7th IEEE Conference on Artificial Intelligence Applications. Rau's paper describes a system to "extract and recognize company names.

The NER identifies NEs in texts using methods ranging from *spelling* and *contextual* rules to vast and heterogeneous pool of strategies, methods, and representations, like Maximum Entropy Taggers, SVMs and CRFs. Most early studies were based on hand-crafted rules [MUC07], the most recent use supervised machine learning techniques [SSN04].

Common exploits in the NER approaches:

- Word features of the token and the words in its neighborhood.
- The parts of speech of the word in question and its neighbors.
- Features corresponding to certain prefixes and sufixes of the word and its neighbors.

39 http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics

40 http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/LinkStatistics

- Features corresponding to the labels of its neighbors.

Built upon the concept of rigid designator, some tasks related to NER:

- Personal name disambiguation - Algorithms for distinguishing personal names with multiple real referents in text, using little or no supervision [MGY03].
- Named entity translation – Using the hidden Markov model (HMM), task of translating NEs from one language to another [HFE05].
- Entity anaphora resolution - A lightweight practical approach to pronoun resolution in the case when the antecedent is named entity, using Gate [DMB02].
- Acronym identification - A supervised learning approach to the acronym identification task [NDT05].
- Case restoration - Truecasing technique, a technique for restoring the normal case form to an all lowercased or partially cased text [AAK06]. Useful in machine translation.

Some applications already make use of NER for information retrieval, to improve and facilitate information mining or to linking across documents. Some examples are:

- Question answering - A study on low-level information extraction like NER for the Q&A TREC-8 tests [SRI99].
- Semantic information retrieval - recognition of names and their associated categories within unstructured text traditionally relies on semantic lexicons and gazetteers [PMA04].
- Local search - Geographical location associated with a query in collective human knowledge and propose a solution to correctly detect it [WLW05].
- Text/Web mining - Agent applications to address healthcare problems, including that highlight the close fit between intelligent agent properties and healthcare problems [SDM05].

More than improve keyword search it can open the door to semantic search, faceted search and document repurposing [DPS07], while providing essential resources to domain ontologies [RPV04] .

Human language is ridden with semantic ambiguity. Semantic ambiguity refers to differences in meaning, and is further broken down into homonymy or polysemy, depending on whether or not the meanings are related.

2.8.1 Word Sense Desambiguation

The bark of a dog versus the bark of a tree is an example of homonymy; opening a door versus opening a book is is one of polysemy.

Syntactic and semantic ambiguity are orthogonal, since a word can have related meanings in different categories (“He will review the review when he gets back from vacation”), or unrelated meanings in different categories (“Can you see the can?”).

[Named Entity Type, Named Entity]
"[Person John Williams] conducted a summer Star Wars concert..."

"[Person John Williams] lost a Taipei death match ..."

Figure 2.9: Name Entity collision.

Several instances of the same class (e.g., different people, with same name) or different classes (e.g., a type of snake, a programming language, or a movie) may share the same name in. The text referring to the name "*John Williams*" can mean "John Williams the star wars composer" or "John Williams the professional wrestler", depending on the surrounding context (Figure 2.9).

Word sense disambiguation (WSD) is the process that governs the identification of the sense of a named entity, using a predefined inventory of senses. *WordNet*[41] is the most commonly used computational lexicon of English for WSD.

There are four conventional approaches to WSD:

- Dictionary based: These rely primarily on dictionaries, gazetteers and lexical knowledge bases, without using any corpus evidence.
- Supervised methods: These make use of annotated corpora to train from.
- Semi-supervised : These make use of a secondary source of knowledge such as a small annotated corpus as seed data in a bootstrapping process, or a word-aligned bilingual corpus.
- Unsupervised methods: Avoid external information and work directly from raw unannotated corpora.

The distinction between the different senses of words will favour ontologies [GCC05], especially in large document corpus [HCU04].

2.8.2 Co-Reference and Alias Resolution

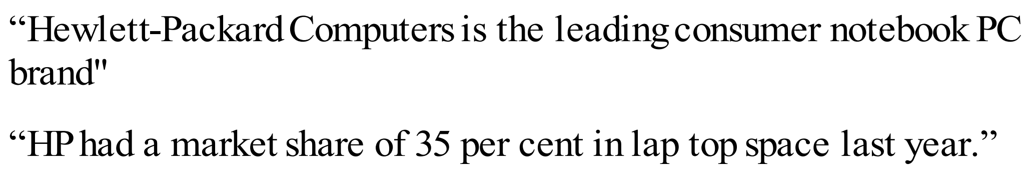
Co-reference resolution can be viewed as the classification task of finding the right antecedent for a referent using grammatical, contextual and morphological features. The use of semantic resources is uncommon, co-references is mainly done by exploiting the context of every occurrence.

"Bill Gates was chairman of Microsoft. He is married to Melinda Gates."

Figure 2.10: Classic co-reference in sentences.

For example, in the Figure 2.10, *Bill Gates* and *he* are most likely co-referent to the same named entity.

41 Lexical concept network



"Hewlett-Packard Computers is the leading consumer notebook PC brand"

"HP had a market share of 35 per cent in lap top space last year."

Figure 2.11:Entity alias sample.

Aliases of an entity are the various ways in which the entity is written in a document. For example in the figure Hewlett-Packard Computers and HP are alias for named entity Hewlett-Packard (Figure 2.11).

Co-references and aliases resolution in a text can be reduced to the same problem of finding all occurrences of a named entity in a document. They pose one of the biggest problems in modern semantic analysis. Methods ranging from supervised to unsupervised exist to tackle the problem [LXM04].

2.9 Conclusion

Ontology design is a crucial research area for semantic technologies. The Semantic Web initiative offers a set of standards (RDF, RDFS and OWL) for the representation and exchange of information.

From the test cases done while exploring the use of RDF/OWL, a range of potential RDF/OWL applications were idealized. It would be foolish to assume any depth knowledge of the RDF and OWL standards, due to scale and complexity, but a summary of their main features in order to make the discussion accessible.

We identify the primary strengths of RDF/OWL as:

- Support for information integration and reuse of shared vocabularies
- Management of semi-structured data
- Web integration
- Flexibility to changes
- Inference mechanisms power
- Classification mechanism, based on a formal semantics, help knowledge extraction
- Representation flexibility, with syntax separated from data modelling
- Ability to represent instance and class information in the same formalism and hence combine them.

Weaknesses noted are:

- Fairs poorly in document validation
- Expressive limitations
- Serious performance issues in larger ontologies
- Learning curve is steep, the concept is abstract
- It is also largely unsuited to domains involving continuous or fuzzy categories
- No support to processes and change representations in vanilla version.

- RDF/OWL is particularly suited to model applications and data, while providing fairly good support to data integration and distribution, providing interoperability and development of resilient systems networks to cope with data models.

The usefulness is diminished in pipeline processing problems, where data model is very stable, not available to clients, and with little or no external interaction. It is also mostly unsuitable to map fuzzy domains.

The combination of a large documentation corpus, with data disambiguation and enhancement, promises a wide variety of approaches to problems.

3 Related Work

The current chapter presents a synthesis of related work, essential to the dissertation, more precisely to the module presented in this document, and the reasons who lead to the choices.

Section 3.1 is dedicated to Jena, the main tool to manipulate ontologies and RDF. The ontology (Section 3.1.1) and query (Section 3.1.2) manipulation are also presented.

The AlchemyAPI is one of many entity recognition and disambiguation tools, the preference over competition is presented in Section 3.2.

The module discussed in this document requires a specific input format. The details are explained in Section 3.3.

Final a short resume and conclusion is provided in Section 3.4

3.1 Jena

Some aspects of W3C's RDF Model and Syntax Specification require careful reading and interpretation to produce a conformant implementation[MBR00]. Jena2 is an API in the Java programming language, for the creation and manipulation of RDF graphs. It implements the interpretation of the RDF specifications described in sections 2.2.4 and 2.3 above and specified in [RDS04]. Jena was developed to satisfy two goals:

- Provide an API that was easier for the programmer to use than alternative implementations
- Conformant to the RDF specifications.

Jena2 is the second generation of the Jena toolkit, an Open source Java framework ontology API [JNOWL] to construct Semantic Web Applications [BMC02]. Jena 2 supports the Jena 1 API [JIC03] and will from now on be known as Jena.

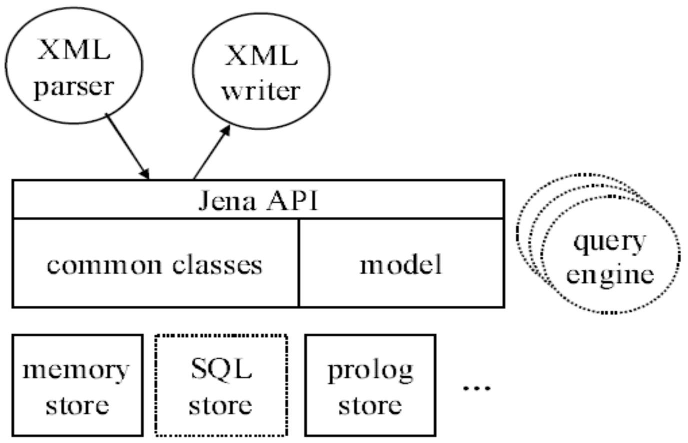


Figure 3.1: Coarse Jena Architecture.

The API (Figure3.1) is particularly adjusted to programmers who wish to learn RDF by rapid prototyping.

It has a collection of Java interfaces representing resources, properties, literals, containers, statements and models. A common set of classes implement these interfaces, though these may be sub-classed or replaced to optimize particular implementations. The model class is a generic implementation of an RDF graph. A standard interface connects model to classes that implement storage and basic querying of RDF statements. The implementation allows the integration of specialized modules for handing parsing, serialize, entailment regimes, store and query of RDF [MBR00].

3.1.1 Jena OWL

The API Jena enables the creation ontologies from scratch with diferent entailment regimes[42]. In Jena, an ontology is treated as a special type of RDF model, *OntModel*. This interface allows the ontology to be manipulated, by coding convenience methods to create classes, property restrictions, and so forth.

Given an ontology and a model, Jena's inference engine support multiple entailment regimes to derive additional statements from the model doesn't express explicitly. Jena provides several Reasoner types to work with different types of ontology. The primary use of this mechanism is to support the use of languages such as RDFS [JAK03] and OWL which allow additional facts to be inferred from instance data and class descriptions.

The framework has various internal reasoners. The default OWL rule reasoner has a rule-based algorithm and gives full support to its own rule format. For complete OWL DL reasoning Jena uses an external DL reasoner such as Pellet, Racer or FaCT.

Jena2 includes an RDFS reasoner[43]. RDFSRuleReasoner can be configured to work at three different compliance levels:

- **Full** - This implements all of the RDFS axioms and closure rules with the exception of bNode entailments and datatypes (rdfD 1). This is an expensive mode because all statements in the data graph need to be checked for possible use of

42 http://jena.sourceforge.net/inference/index.html

43 http://www.w3.org/TR/rdf-mt/#RDFSRules

container membership properties. It also generates type assertions for all resources and properties mentioned in the data (rdf1, rdfs4a, rdfs4b).

- **Default** - This omits the expensive checks for container membership properties and the "everything is a resource" and "everything used as a property is one" rules (rdf1, rdfs4a, rdfs4b). The latter information is available through the Jena API and creating virtual triples to this effect has little practical value. This mode includes all the axiomatic rules.
- **Simple** - This implements just the transitive closure of subPropertyOf and subClassOf relations, the domain and range entailments and the implications of subPropertyOf and subClassOf. It omits all of the axioms. This is probably the most useful mode but is not the default because it is a less complete implementation of the standard.

The OWL reasoners are extensions of the RDFS reasoner and support to all entailments supported by the RDFS reasoner. RDFS is not a subset of the OWL/Lite or OWL/DL languages the Jena implementation is an incomplete implementation of OWL/full.

Currently there are three different implementations of reasoners:

- **OWLFull** – the default implementation, supports all constructs[44].
- **OWLMini** – Nearly the same as OWLFull, but omits the forward entailments from minCardinality/someValuesFrom restrictions.
- **OWLMicro** – Trimmed version of OWLMini, supports RDFS plus the various property axioms, intersectionOf, unionOf (partial) and hasValue.

The three implementations are a set of useful but incomplete implementation of the OWL/Lite subset of the OWL/Full language. The full reasoner passes the normative OWL working group positive and negative entailment tests, and is the one use in the work.

3.1.2 Jena ARQ

SPARQL support in Jena is currently available via a module called *ARQ*. In addition to implementing SPARQL, ARQ's query engine can also parse queries expressed in RDQL or its own internal query language. ARQ is under active development, and is not yet part of the standard Jena distribution. However, it is available from either Jena's CVS repository or as a self-contained download.

Queries to the created model will return not only those statements that were present in the original data but also additional statements than can be derived from the data using the rules or other inference mechanisms implemented by the reasoner. With Jena when the inference Model is queried then the query is translated into a goal and the engine attempts to satisfy that goal by matching to any stored triples and by goal resolution with backward chaining rules.

3.2 AlchemyAPI

Alchemy has built in 'entity recognition and disambiguation' mechanisms, employing tens of millions of contextual hints describing traits of the world’s objects, individuals, and locations. It employs a variety of public and non-public data-sets.

44 Resources that increase the ontology instances expressive power.

Hints vary depending on the specific type of entity being disambiguated. For example, when disambiguating people, the person career information is used, where they're located, who they work for, and so on. For companies: key executives, notable products, industry, location, etc.

Whenever an entity is successfully disambiguated, additional information is returned in API responses.

[Named Entity Type, Named Entity Disambiguation, Named Entity]

“[Company Hewlett-Packard Hewlett-Packard Computers] is the leading consumer notebook PC brand"

“[Company Hewlett-Packard HP] had a market share of 35 per cent in lap top space last year.”

Figure 3.2: Alias Resolution.

The entity extractor knows that "HP" is a common alias for "Hewlett-Packard" and it then can check the employee directory and put the correct full name of the person in the person attribute (Figure 3.2). The date extractor knows about many different date formats that might appear in text and can use document context (such as the year the document was created) to infer the precise date referenced in the document even though it was not specifically provided. The product name extractor can look up the new name of a product and associate it with a product database and insert the correct product ID. Future keyword searches can look up product history and search for all documents with multiple product names.

[Named Entity Type, Named Entity Disambiguation, Named Entity]

“[Person Bill Gates Bill Gates] was chairman of Microsoft. [Person Bill Gates He] is married to Melinda Gates.”

Figure 3.3: Co-reference resolution.

The AlchemyAPI now resolves he/she/his/her/etc co-references into named entities (Figure 3.3), providing a more comprehensive view of processed texts.

The AlchemyAPI provides Named Entity Extraction, with support to semantic ambiguity, alias resolution and disambiguation capabilities for text. It can identify dozens of entity types [AET10] and context-sensitive entity disambiguation [AED10].

Linked Data is a method of exposing, sharing, and connecting data on the Web via dereferenceable URIs. AlchemyAPI integrates with a variety of resources within the Linked Data cloud[45], to provide additional information describing named entities and enriching the overall content. Whenever an entity is successfully disambiguated, Linked Data about the resource is included in API responses. Listed below are the Linked Data resources currently leveraged by AlchemyAPI:

- Freebase
- US Census

45 http://linkeddata.org/

- Geo Names
- UMBEL
- OpenCyc
- Yago
- MusicBrainz
- CIA Factbook
- CrunchBase
- Dbpedia

With AlchemyAPI, semantic content is exposed, allowing the creation of tools to execute analysis and include meta-data annotation in data.

3.3 Tuple Extraction

The process of tuple[46] extraction requires the application of a range of techniques for natural language processing. A related module was developed to handle the natural language processing and organized it in a chain of sequential execution, ie a module takes as input the information from its immediate predecessor.

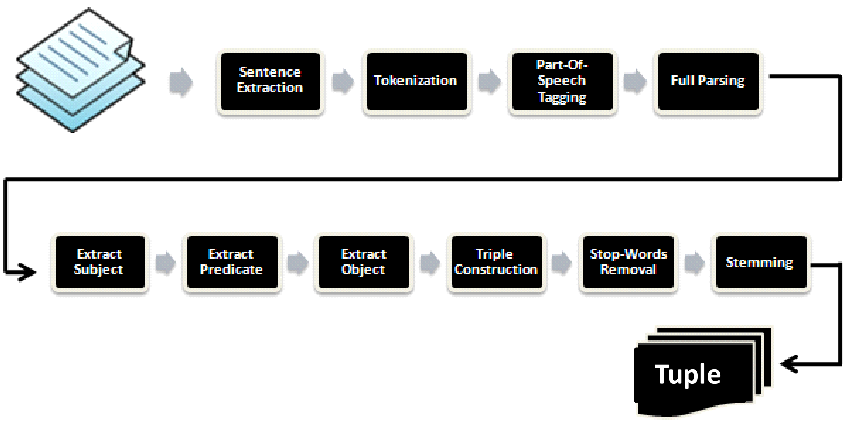


Figure 3.4: Tuple creation pipeline.

The algorithm described here was implemented based on the work presented in [DLF07], it extracts from triple trees generated from parsing the format of annotations style Penn Treebank [PPB04].

The work presented in [JLE04] defines a tuple as a relationship between subject and object, an attribute that relates the predicate. Pretty much the same as RDF, but to achieve a richer knowledge representation, it is necessary to extract the modifiers associated with each of the three basic components of a tuple. Thus an approach was adopted in which the modifiers of a word / phrase is neatly attached to the element of triples that are changing, providing extra semantics.

So for every element that is a tuple is searched their modifiers. For example, the attributes of a name are mostly adjectives, attributes of a verb are mostly adverbs.

46 Ordered list of elements.

A complete research into the subject (Figure 3.4) was done by Fabio Malheiro in "*Extracção, Semi-Estruturação e Interacção com Informação utilizando técnicas de Processamento de Linguagem Natural*".

3.4 Conclusion

AlchemyAPI provides Named Entity Extraction and disambiguation capabilities for analyzing text, providing very fast response time. Compared to existing tools in market (Anexo A), returned entities have good quality and with a few exceptions. With every disambiguation a dereferenceable links to other structured databases such as Freebase and Dbpedia is given.

One tools who provides similar resources is Evri[47], who gathers named entities and include semantic links, but recognizes less entities and only provides a reference to the entity in Evri's own system. The other is OpenCalais[48], fared well in entity recognition, but lacks good disambiguation and Linked Data features. In its current incarnation, OpenCalais results require manual linking between Named Entities to LOD.

Alchemy calls to the API are very fast, and the usage limits are generous, with 30,000 calls per day available, even for commercial uses. The support of 97 different languages is positive, but the best support is in english.

Jena proved to be a very powerful tool for supporting ontologies, with the added bonus of being developed in Java, which is a programming language commonly used commercially, with a large number of resources.

The Jena has a very ample documentation. In large part this is due to the fact that its origin is in the laboratories of HP, which today contribute to the development and documentation generation.

It should also be emphasized that the tutorials and online support facilitate the understanding of the framework.

Without the concept provided in the work " Triplet extraction from sentences" [DLF07], the module developed by Fabio Malheiro wouldnt be possible to develop. The subject of document representation was one of the main subjects of research of this work.

47 www.evri.com/
48 www.opencalais.com/

4 The Prymas System

This chapter makes a deeper description of the proposed system architecture, followed by the methodology required to achieve the planned goals.

This section clarifies the *Prymas* Knowledge Management system concept, design and the modules relations. Secondly, the Knowledge Discovery Layer architecture is presented in particular, showing relations between the Knowledge acquisition layer and knowledge retrieval layer.

Moreover, special relevance to system expectations, methodology and tools are given.

4.1 Knowledge System

To handle the demands of a vibrant growing market, large organizations began to pay special attention to the problems of knowledge management. Users nowadays don't want to use complex technical systems to access information, in the majority of the cases a simple question should be more than enough to find the information in the system.

Recognizing the problem, Wipro Portugal, Department of Innovation, put forward the challenge to conceptualize and implement a Knowledge Management System (KMS) with support for Natural Language Processing (NLP), to supporting the entire lifecycle of acquisition, process and access to knowledge, giving special focus to the study of the current state of art, both in technological and scientific approaches

Prymas is the end result. A distributed Knowledge Management system to acquire, process, and inference of knowledge from fragments of text. Split into 3 different modules.

Prymas is intended be one answer to current knowledge management problem within Wipro Retail and customers, supporting better management practices on large collections of knowledge assets and allowing a more effective and quickly interaction with knowledge.

4.1.1 Architecture Overview

Prymas is designed to support all the knowledge development cycle, allowing identification, representation, construction and distribution of knowledge, by adopting the best practices and solutions, within the organization and offer a more effective and closer engagement of individuals with the business processes. Corporate employees will have the opportunity to continuously contribute with knowledge and access information from a central knowledge repository.

Later, this project aims to lend a hand in training processes and consulting projects in retail environment. As a matter of fact, it will provide untrained users with swift access to knowledge that in many cases would take an expert some time to find.

Prymas will materialize into a virtual knowledge expert agent. The document collection, stored in a knowledge repository is linguistically analysed, in order to create a semantic network of facts and relations. The same document collection is also indexed by an Information Retrieval module, allowing a fast retrieval of its contents.

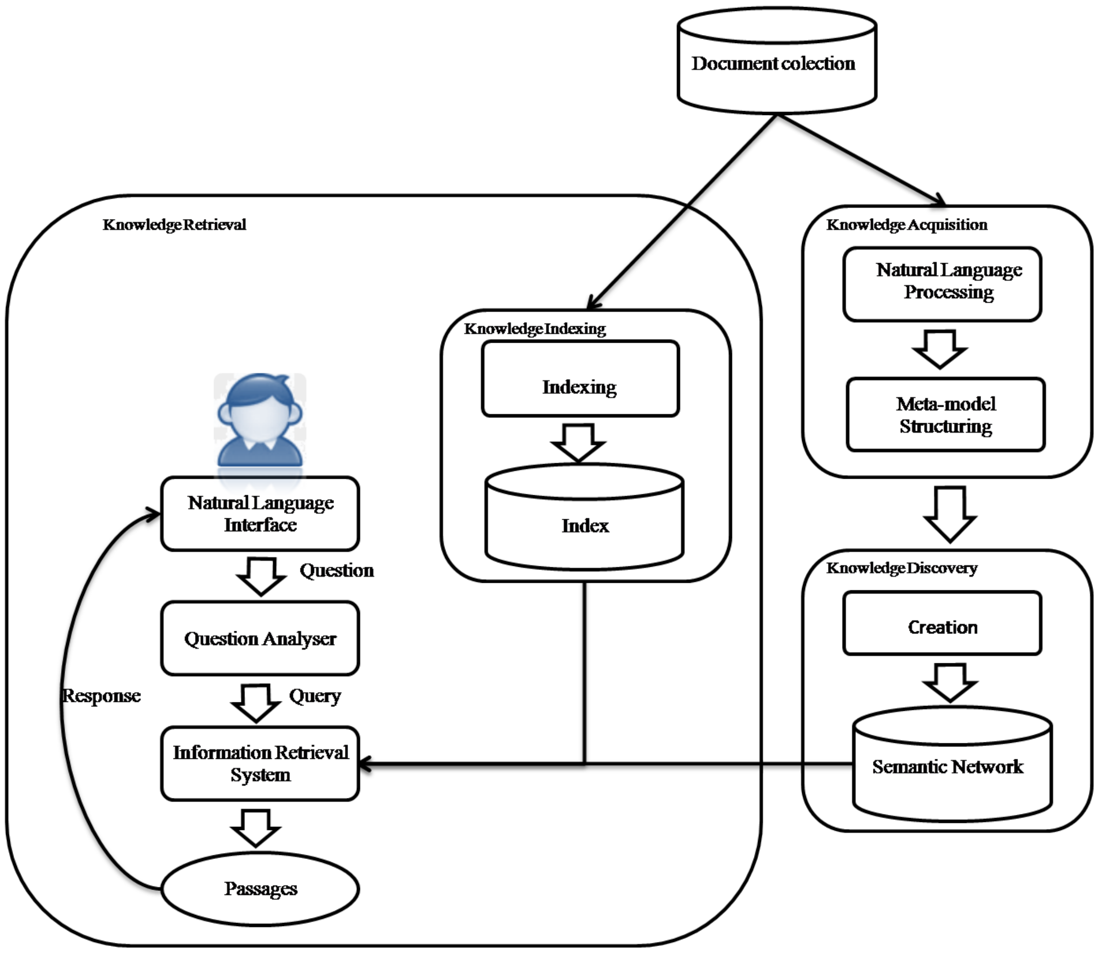


Figure 4.1: System Architecture.

As represented in Figure 4.1, *Prymas* is a Question and Answering (Q&A) system to supports information retrieval, searching relevant passages and references across the document collection. Besides, the document collection it also includes Natural Language processing, in order to build an intermediary representation to facilitate the relationship construction of the semantic network system.

With his contribution, *Prymas* is expected to provide active support to knowledge management, since individuals will make an effort to explicitly encode their knowledge into a shared knowledge repository, plus retrieved knowledge takes into consideration previous feedback from users.

The motivation underlying the development of KM system considers several considerations:

- Increase the use of the organization knowledge base.
- Ease the use of the organization knowledge base.
- Support the products and services lifecycles.
- Ease new product and services learning curve, reduce the overall development cycle.
- Provide access to expert knowledge across the organization in a distributed way.
- Managing intellectual capital and intellectual assets in the workforce (such as the expertise and know-how possessed by key individuals).

Prymas is divided into three important modules; Knowledge Acquisition, Knowledge Discovery, and Knowledge Retrieval.

4.1.2 Knowledge Aquisition Module

The first module from *Prymas*, Knowledge Acquisition, is concerned with acquiring and interpreting information from unstructured sources, such as text documents and technical reports. Then, through text extraction techniques a structured meta-model is produced, containing explicit knowledge in their representations. This is an in-between-representation between data and knowledge, as it already comprises some syntactic relationships between concepts.

Thus, this process includes all the phases from the conversion of the raw text lines from one sentence, to explicit knowledge relationships mapped into a meta-model representation from the original text sentence.

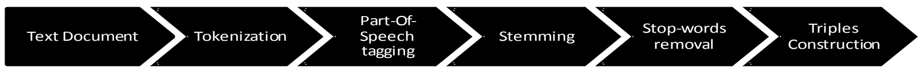


Figure 4.2: Knowledge Acquisition Flow.

Figure 4.2 describes the knowledge acquisition flow, from an unstructured text documen that include the following steps:

- Tokenization
- Part-of-Speech tagging
- Stemming
- Stop-words removal
- Triples Construction (subject-predicate-object)

In fact, the Knowledge Acquisition process described in Figure 4.2, comprises the application of Natural Language mechanisms in order to perform the interpretation and transformation of unstructured knowledge into a structured and explicit relationship. This task uniforms information into a unified representation. Then, as soon as the facts representation is built, it can be manipulated and exploited by automatic computer mechanisms.

4.1.3 Knowledge Discovery Module

The second module from *Prymas*, Knowledge Discovery, is responsible for obtaining implicit knowledge trough explicit knowledge analysis and association. This is carried out through the automatic ontology construction, where relationships within triples are mapped into a network. The input for this layer is the output meta-model representation from the previous Knowledge Acquisition module.

With the raise of the Semantic Web, a web of data with defined meaning, transverse to multiple domains, enabling computers and users to work in cooperation. New possibilities to represent and integrate knowledge emerge, and new languages to deal with information are implemented, for an assortment of purposes.

The ontology is a dependable way to publish information, a declarative description of a concept fundamental understanding, in the world of interest, and is used in this work as structure to represent knowledge.

This form of Knowledge Representation is represented by a graph, where the vertices are the concepts, and the edges are the relationships between the concepts. Each relation in the network is a representation of a fact that is classified by a relation type. Ontology representation supports mechanisms to reason and infer, allowing exploitation of the knowledge. Furthermore, this module uses a generic ontology hierarchy in order to reflect the state of the world.

One of the key challenges of the Semantic Web is how to go from today's unstructured web to a web rich with semantic information. On today's Web, there is a growing number of data sets published according to the Linked Data principals, the majority of them being part of the Linked Open Data (LOD) cloud. As LOD connects data and people across different platforms in a meaningful way, one can assume that harnessing LOD, by means of inclusion and manipulation of data, can greatly improve the KR and the overall KMS.

The ontology's construction method allows the use of diverse domains, increasing the flexibility of the module.

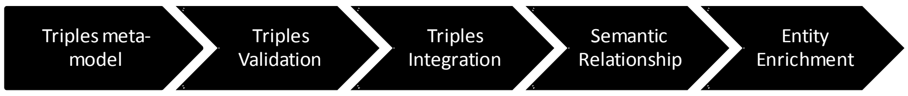


Figure 4.3: Knowledge Discovery flow .

As we can see in the Figure 4.3, this *Prymas* architecture includes an important layer of knowledge detection and enhancement. After the knowledge buried within documents is converted into a triples representation, this Knowledge Discovery layer is responsible for processing the relations and enhanced existing content with recognition of entities and using relations existing in linked open data format, in the web. This increases the existing knowledge with new and useful knowledge .

4.1.4 Knowledge Retrieval Layer

The third module from *Prymas*, Knowledge Retrieval, comprises the interaction and communication between the user and the Question and Answering System. It embraces all the phases from one question from a user, to the final response to the user. This layer includes a natural language interface. More precisely, users interact with the knowledge base my means of a Question and Answering system. Users ask questions in English, and receive answers also in natural language. QA systems grant users the ability of easily inquire knowledge database, requiring only a small learning curve of the system. This is a valuable feature, as we want to have a system that can be used not only by technicians, but also for a wide range of users without technical background.

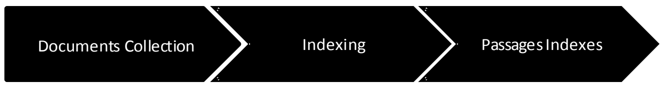


Figure 4.4:Indexing Flow.

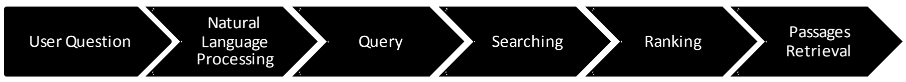


Figure 4.5:Question and Answering Flow.

As the Figure 4.4 shows, after the unstructured knowledge berried within the documents being indexed by an Information Retrieval engine, passages from documents can be searched.

It is shown in the Figure 4.5, Question and Answering flow, from the user question until the final passage as response to the used. The first step from this module is to perform a set of Natural Language tasks by the question analyzer in order to processes the user question, and convert into query term that will be useful for the IR system. Moreover, the terms are enriched with linguistic features that will support the enhancement of the passage retrieval, such as entity recognition and synonyms to the query terms. The entities are also searched in the ontology network, in order to provide further information to the IR system.

4.2 Expectations

The semantic web is a whole new research field. Ontology is a building block of the SW and its development is not an easy task. The success is dependant more on the creator art than technology. Common problems arise when defining the concepts of the ontology, this raises questions like:

- The ontology should be manually or automatically built?
- Is the ontology right for the problem?
- Do we have access to domain experts to build and\or validate the ontology?
- Should the knowledge be strongly domain-specific focused?
- How can we use the ontology mechanisms for inference?

Every time a new ontology is built related to a different concept, the same barriers appear. There is no definite process and can run into two completely different problems:

- Ontology too simple and too generic.
- Ontology so complex that makes their use impractical.

A balance is essential, where represented information is meaningful, reusable, and can be used with semantic web technologies, especially reasoners, to provide inference.

On today's Web, there are a growing number of data sets, published according to the Linked Data principals, the majority of them being part of the Linked Open Data cloud. The goal of Linked Data is to enable people to share structured data on the web. Taking the basic assumption behind linked data that the value of information increases the more it is interlinked with other data[49], a reliable way to integrate and exploit the existing information within linked data is sought.

An Information Retrieval Systems is useless if we can’t retrieve information. To take advantage of the ontology semantic power and knowledge inference mechanism and the free and readily available linked data information, a method that is capable of querying the system is required.

The work at hand deals with this problematic, particularly in terms of automating the process of creation and instantiation of the ontology and query construction.

The project will run as part of an overall project of creating a system for Acquisition and Management of Distributed and Dynamic Knowledge in Natural Language Interaction with support for knowledge development and consultancy projects in the Retail sector.

Taking the existing available retail knowledge, distributed across multiple documents, we try to create an automatic system to represent knowledge, instantiate and enhance information as ontology.

The time constrains put a strain in the objectives, but results are essential. After 16 weeks the following results are expected:

- Insight into current state of art both market ready in scientific research knowledge representation systems.
- Experience Semantic Web building blocks, taking into special account semantic languages, ontologies and tools.
- Generalize the handled problems to related applications.
- Generate an automatic system to represent knowledge.
- Instantiate and enhance information as ontology, using the power given by the SW to exploit the data.
- Provide a system that will help user in the pursuit of knowledge existent in the documents

To resume, as part of a global challenge to conceptualize, design and implement a proof of concept Knowledge Management System, with the support of Natural Language Processing and semantic information, following a philosophy of "Open Innovation".

This dissertation focuses on all the research and the system it originates. The system is a proof of concept of a method for generating a semantic representation of a document; enhance the volume of current information and their quality, using external data and metadata to improve the existing knowledge base.

4.3 Methodology

After a process of joint research, followed by an individual phase, a common set of technologies promised solution to the problem presented. The focus lies on the SW technologies, with particular focus on ontologies, properties, languages and associated tools.

49 http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

When the technological focus has been reached, the design of architecture began. After a team process of workload division and experimentation with technology was done. After having a architecture and solid experience with the technologies, the development process started.

The development process of this module is composed of 6 steps:

1. Build of a tuple knowledge base
2. Extract named entities
3. Enhance tuples with Linked data
4. Ontology inference

As described, the tuples represent the sentence syntax and semantic, inspired in the RDF subject-predicate-object information representation with some adaptations [ADE09]. It represents the relation between a subject, predicate and object elements, where each one can have or more modifier elements. From the analyses of the text document, with a syntactical parser for English is generated a parse tree with metadata of the sentence, followed by the extraction of the tuples elements using a heuristic parser dependent techniques. The result is a semantic representation of each sentence, following a customized tuple representation.

For each customized tuple, with Jena is created its OWL language representation. OWL uses a triple format knowledge base. With the tuple elements, when mandatory, new classes are built and respective individuals created. Data properties and Object properties are used as required. They represent the sentence structure.

Using the tuples and the AlchemyAPI, named entities are extracted, alias and basic co-reference solved. With the extracted entities, new classes are built and respective individuals created. The Data properties and Object properties are also used as required.

In the final step, for each tuple element, if the AlchemyAPI provides Linked Data the data is used to create new individuals Data properties and Object properties, to enhance the ontology.

Jena supports instance-based reasoners [JNINF]. That is, they work by using rules to propagate the if- and only-if- implications of the OWL constructs on instance data. They infer OWL instances for the existing classes, from the fashioned representation of triple knowledge base. It allows the inference of new knowledge.

The OWL classes and instances generated by the system are used as knowledge base by the system, to infer new answers to user queries. The answers are supported by ARQ a query engine for Jena that supports the SPARQL RDF Query language.

By adding new sentences, more tuples are created, more named entities recognized and more Linked Data added, interlinking more data and improving user queries results.

4.4 Development Enviroment

The work was developed in the operative system Windows XP. The creation, manipulation and publish of the ontology is done with Jena [Jena].

It provides a programmatic environment for manipulating RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. The ontology is manipulated in the OWL format [RFOWL]. It expresses everything that RDFS allows and much more.

The Jena inference subsystem, [JNRES] is designed to allow a wide range of inference engines or “reasoners” to be plugged into Jena. The primary use of this mechanism is to support the use of languages such as RDFS and OWL which allow additional facts to be inferred from instance data and class descriptions. The reasoner has a validation interface to detect when such constraints are violated by some data set.

ARQ is a query engine for Jena that supports the SPARQL RDF Query language [RDAT], a query language [RSPQL] and a protocol [SPQLP] for accessing RDF designed by the W3C RDF Data Access Working Group.

The AlchemyAPI [ALAPI] is capable of identifying people, companies, organizations, cities and other typed entities from text. To provide additional information describing the named entities detected, the AlchemyAPI provides comprehensive support for RDF and Linked Data to enriching the content. It is integrated into the project with the Java SDK.

The Protégé-OWL editor is an extension of Protégé that supports OWL, and is used to visualize classes, properties and individuals generated by the java code. It also allows the test of different reasoners such as pellet [PELLT] or Fact++ [FACTR].

5 Knowledge Discovery

With a work plan cut, the implementation process starts. This chapter gives a step by step description of the implementation.

Besides the semantic web view (Section 5.1), a description of languages (Section 5.1.1), and the relation with the ontology (Section 5.1.2) in the work is provided. Trailing is the actual implementation process (Section 5.2), where the namespace (Section 5.2.1), action and entity classes are created (Section 5.2.2), follow by the creation of class concept (Section 5.2.3), and all the individuals (Section 5.2.4).

Final a short resume and conclusions are provided in Section 5.3.

5.1 Base ontology and semantic Web

Semantic Web is a term coined Tim Berners-Lee [TBL98]. It is a vision for the future of the Web, in which information is linked up and have explicit meaning [WOT04], making it easier for machines to automatically process and integrate information available on the Web, providing a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.

According to the original vision, the porting of a document to Semantic Web involves the insertion of semantic notes (machine-readable metadata) to the original data, enabling machines to perform in a more intelligent way [TBL01]. No information is added, we only represent information in a more accessible format to access and infer.

5.1.1 Language

The Semantic Web initiative offers a set of standards (RDF, RDFS and OWL) for the representation and share and reuse of information [DCJ05].

OWL, web ontology language, is a representation of term vocabularies and relations between them. It’s based in DAML+OIL [DOL01] (a successor language to DAML and OIL), where refinements of design and use where done. OWL extends RDF and RDF Schema and employs a triple representation model. Its design principle includes developing a standard language for ontology representation, to enable semantic web, and consequently extensibility, modifiability and. The normative exchange syntax for OWL is RDF/XML [RDF04].

The expressive and reasoning power of description logic to the semantic web made it the chosen tool to use in the ontology built the knowledge database.

5.1.2 Ontology

With the language defined and a 3-tuple representation of the documents, the domain ontology, a meaningful representation of the documents content and structure, can be built.

The Ontology is a formal representation of domain knowledge. In general, domain experts work with strong domain specific knowledge. With the help of domain ontology, we can understand the relationships among the concepts in a text, and use the knowledge in various applications [GTR93]. But it's rather difficult, even for experts to come up with objective and generic knowledge to represent in the ontology, hence making it.

The creation and update is costly, but crucial, specially now with dynamic data. An automatic domain ontology acquisition is preferred. Most approaches use only nouns as the bricks for ontology building. These methods usually resort to clustering methods and disregard any ontological relations between other word classes.

The ontology needs to se semantically coherent and capable of represent significant concepts and pertinent relations, but at the same time provide a useful domain description. It's composed of classes, data properties, object properties and individuals.

One of the building pillars of this dissertation is the possibility of enhance the ontology with linked data, using shared structured data on the Web, to create typed links between data from different sources. Importing other resources from the web brings the entire set of assertions provided by their ontology into the current ontology. In order to make best use of this imported ontology they are coordinated with a namespace declaration, fully supported by OWL [OWL04].

Additionally we don't want to retrieve just explicit stated knowledge, with OWL reasoning capabilities we want to infer implicitly stated knowledge on the ontology [JNI10].

OWL makes use of the Open World Assumption [], under this open world assumption, if a statement cannot be proved to be true using current knowledge, we cannot draw the conclusion that the statement is false.

Finally the creation of the ontology and their enhancement leads to improve the capability of querying the information. As a query language, SPARQL is "data-oriented", it only queries the information held in the model [SPARQ]. SPARQL does not do anything other than take the description of what the application wants, in the form of a query, taking into account entailment regimes[ENT10] and returns that information, in the form of a set of bindings [PPA10] or an RDF graph [ARQ10]. ARQ is a query engine for Jena that supports the SPARQL RDF Query language.

5.2 Ontology Creation

The name "ontology" comes from Greek philosophy and means "*the study of the nature of being*". The term is used in the domain of Knowledge Representation "to categorize the kinds of things existing”. The aim is to fix a common vocabulary of terms able to describe as much knowledge about the world as possible from given domain, and to subdivide this knowledge in a coherent class hierarchy, so as to create a shared knowledge representation language.

Usually an Ontology is composed of the following:

- classes of objects
- instances
- relations between instances and classes

Existing ontology construction have severe limitations in creating ontologies from representative text collections. Considerable research has gone into developing ontologies and applying them to a variety of applications. The extraction of domain knowledge for developing these ontologies is often performed on a manual basis, depending of the Ontologycreator’s preferences and abilities.

Ontology creation is iterative process of modelling the given domain, by choosing the most important concepts and identifying the most relevant relations between them.Extracted knowledge is then organized into a domain ontology.

5.2.1 Namespace

XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references.

The base ontology and instances use the namespace with name “sample” and location “http://www.example.com/ontologies/”.

5.2.2 Entity and action

It is essential to create a simple representation, but at the same time reliable and robust. The system can’t have serious limitations that could hinder future improvement iterations. The semantic web provides a vast array of tools and technologies to solve these problems.

The simplest building block is the triple, conventionally written in the order subject, predicate. An *RDF triple* contains three components:

the subject, which is an resource or a blank node.

the predicate, also known as property, is an resource.

the object, which is resource, a literal or a blank node.

A resource is an attribute where the value of which is interpreted from a RDF URI reference object node.

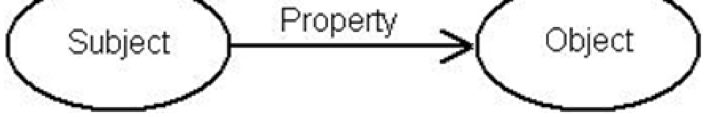


Figure 5.1: RDF triple.

As shown in figure 5.1, the expressions are known as triples in RDF terminology. The subject denotes the resource, and the property denotes traits or aspects of the resource and expresses a relationship between the subject and the object.

The first step of the automatic pipeline for ontology creation is complete. We have a custom built, automatic, tuple elements creation process. Each sentence is represented by one or more tuples.

The first step of the automatic pipeline for ontology creation is complete. We have a custom built, automatic, tuple elements creation process. Each sentence is represented by one or more tuples.

The tuples keep the majority of the sentence semantic and structure, as feasible. When creating the sentence ontological representation, the loss of information is also avoided, taking into account the tuple original representation limitations.

From the tuples, we can create the classes for entities, actions and their relations. It requires no human intervention. Classes provide an abstraction mechanism for grouping resources with similar characteristics.

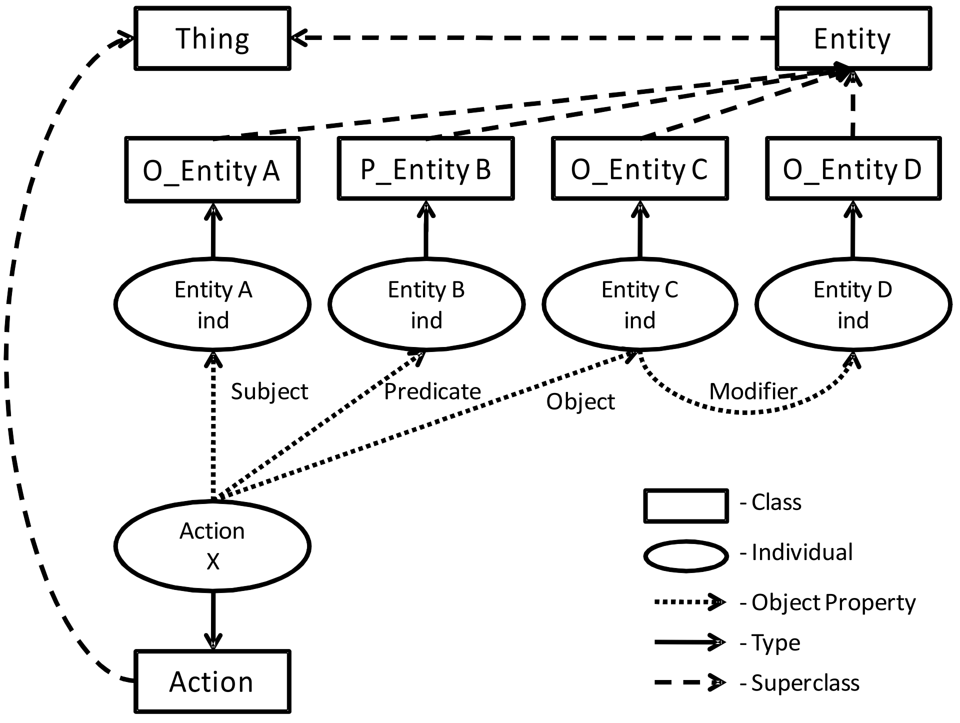


Figure 5.2: Simple tuple mapping.

The action is the tuple elements representation of the sentence structure and the notions relation. The class keeps the essential sentence structure.

In every tuple, elements represent an ontology notion, even modifiers. Entity is the system top class to represent sentences elements as basic notions.

For each element in the tuple we will create a new class representation, if the notion doesn’t exist. They are called Entity Elements.

New Entity Elements are created from new tuple elements and added to the ontology, they represent a new notion. The notion created is a subclass of entity. Some tuple elements have modifiers. Every modifier element is processed as any other tuple element. It is represented as subclass of entity.

Figure 5.2 shows a representation mock of a tuple with 4 different elements. Following the rules, 4 new unique entities, subclass of Entity are created.

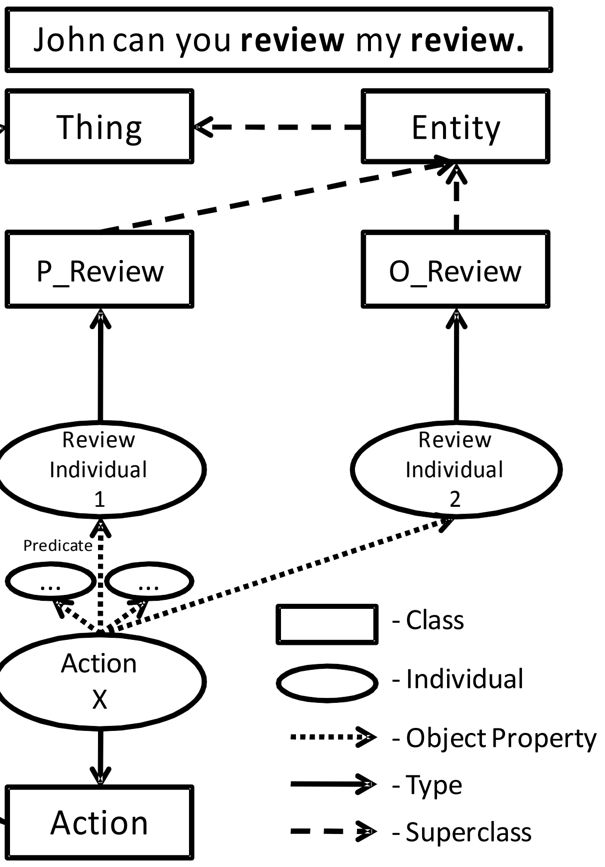


Figure 5.3: Elements representation collision.

Despite all the effort with NER, there are still problems with WSD. The tools are unreliable, and the science models incomplete. To reduce syntactic and semantic ambiguity, at least a bit, when creating a new class a different tag is given, a prefix in the name. Elements representing the action notions subject, object and their modifiers get "O_", the representation of predicate and their modifiers get "P_" preffix.

As shown in Figure 5.3, in "John can you **review** my **review.**" the first review is a verb in base form, but the second is a noun in singular form. This is a simple and neat way to avoid crossing of elements representation of subject and object, and their modifiers, with elements representation the predicate and their modifiers, who are more relevant to the relations between subject and object.

For the case shown, the base skeleton of the classes are:

```
<owl:Class rdf:about="&sample;Entity">
  <rdfs:label rdf:datatype="&xsd;string">Entity representation</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">Base class of entity. Keeps every
entity in a document</rdfs:comment>
</owl:Class>

<owl:Class rdf:about="&sample;Action">
  <rdfs:label rdf:datatype="&xsd;string">Action representation</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">Base class of Action. Represents
```

```
the tuple relations</rdfs:comment>
</owl:Class>
```

5.2.3 Concept

The term "Named Entity Recognition" (NER) refers to the process of recognizing information units such as names, including person, organizations, locations, times and quantities, etc, from unstructured text.

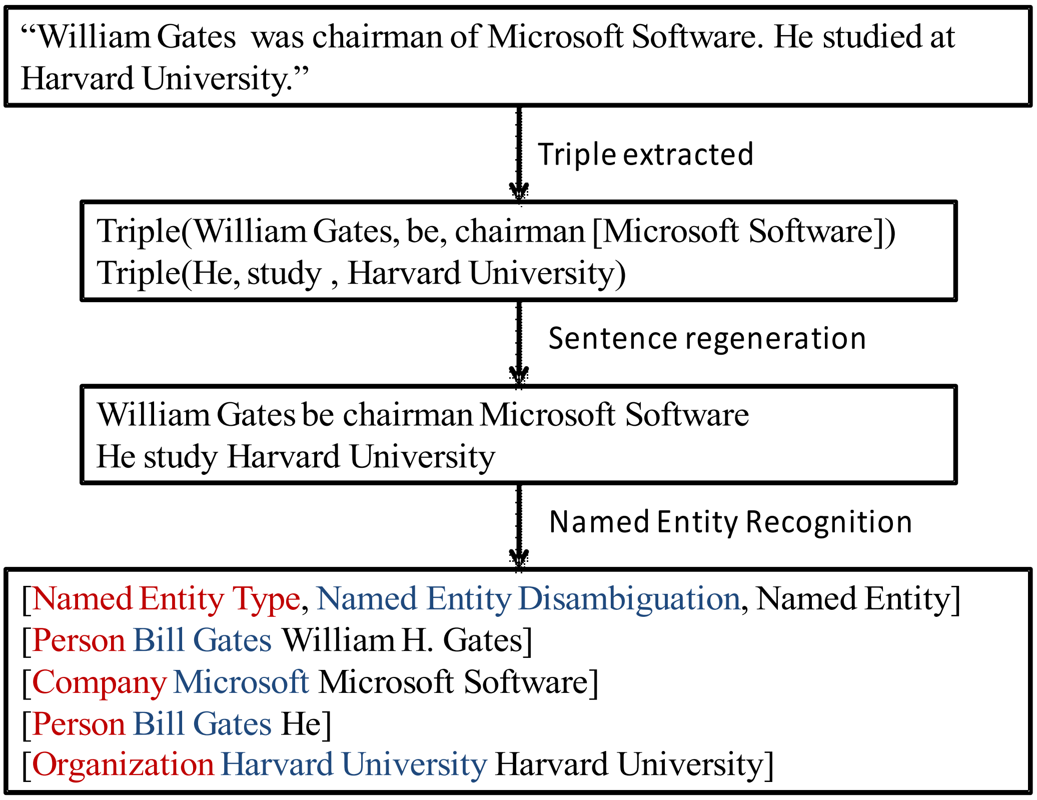

"William Gates was chairman of Microsoft Software. He studied at Harvard University."
Triple extracted
Triple(William Gates, be, chairman [Microsoft Software])
Triple(He, study , Harvard University)
Sentence regeneration
William Gates be chairman Microsoft Software
He study Harvard University
Named Entity Recognition
[Named Entity Type, Named Entity Disambiguation, Named Entity]
[Person Bill Gates William H. Gates]
[Company Microsoft Microsoft Software]
[Person Bill Gates He]
[Organization Harvard University Harvard University]

Figure 5.4: NER pipeline

The tuples extracted from each sentence don't provide extra information, but at the same time the majority of the semantic context and structure is not lost.

Doing Named Entity Recognition only for each tuple element would provide fewer results. They would be of feeble quality, since the surrounding text relations are not examined for contextual cues.

The determination of he/she/his/her/etc co-reference into named entities provides a more comprehensive view of processed texts, but it's impossible, if examination is done tuple by tuple.

To improve the NER process, and allow co-reference resolution, sentence regeneration[50] is done. Keeping the tuple creation order is essential to produce correct co-reference resolution. For each tuple a direct concatenation elements is done as shown in Figure 5.4.

50 Custum built process to recreate sentence semantics, from the tuples.

```
<?xml version='1.0' encoding='UTF-8'?>
<!ELEMENT results (entities|language|url|usage|status)*>
<!ELEMENT entity (disambiguated|text|count|relevance|type)*>
<!ELEMENT disambiguated (yago|opencyc|umbel|freebase|dbpedia|name)*>
<!ELEMENT disambiguated (website|semanticCrunchbase|crunchbase)*>
<!ELEMENT status (#PCDATA)>
<!ELEMENT usage (#PCDATA)>
<!ELEMENT url EMPTY>
<!ELEMENT language (#PCDATA)>
<!ELEMENT entities (entity)*>
<!ELEMENT type (#PCDATA)>
<!ELEMENT relevance (#PCDATA)>
<!ELEMENT count (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT dbpedia (#PCDATA)>
<!ELEMENT freebase (#PCDATA)>
<!ELEMENT umbel (#PCDATA)>
<!ELEMENT opencyc (#PCDATA)>
<!ELEMENT yago (#PCDATA)>
<!ELEMENT crunchbase (#PCDATA)>
<!ELEMENT semanticCrunchbase (#PCDATA)>
<!ELEMENT website (#PCDATA)>
```

Figure 5.5: AlchemyAPI result schema.

The AlchemyAPI identifies information units, in the sentence, returning the result in the XML format, where the original text, type and disambiguation are held (Figure 5.5).

When we create the Entity class, we check if there is Named Entity information. In the NER XML we look for the text element [WSM04, XML08], inside entity element, with the same name as the tuple element, using XPath [XPA99]. Each new Named Entity Type (NET), leads to the creation of a new class representation. They are created as subclass of Concept.

The class Concept groups Named Entity type information into relevant groups, providing extra information to the entity notions.

The process is repeated till all tuples are processed. The concept representation class is:

```
<owl:Class rdf:about="&sample;Concept">
  <rdfs:label rdf:datatype="&xsd;string">Action representation</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">Base class of Concept. Represents
named entity types and relations with entities</rdfs:comment>
</owl:Class>
```

5.2.4 Individual creation

Each class has associated a unique set of individuals and each tuple element is represented by and unique single individual. The individual creation process runs simultaneous with the class creation process. Individuals represent objects in domains of interest [OWL04].

Properties are URI references [URI98] and Individuals have a relationship denoted by the properties. Action is related with the respective entity, using properties *subject*, *predicate* and *object*. The properties have Domain Action, Range Entity, are functional [FUN04] and share the base ontology namespace.

Property modifier relates entities. They have Domain and Range Entity. The properties are functional and have the base ontology namespace.

Every time a tuple is processed, a new action individual is created. The individual has as superclass Action, denotes as being of the type Action.

The tuple elements (subject, predicate, object and every modifier) originate always one single unique individual. The individual has as superclass Entity Element. Due to the ontology inference mechanisms it has also superclass Entity and Thing. The individual is acknowledged as *Entity individual*.

OWL individuals don’t have "unique names”. On the web, such an assumption is not possible. For example, the same person could be referred to in many different ways. For this reason OWL does not make this assumption. Unless an explicit statement is being made that two URI references refer to the same or to different individuals.

The avoid namespace collisions, each individual has a unique hash attached as prefix name. The individual has the property label, with namespace RDFS, used to provide a human-readable version of a resource's name.

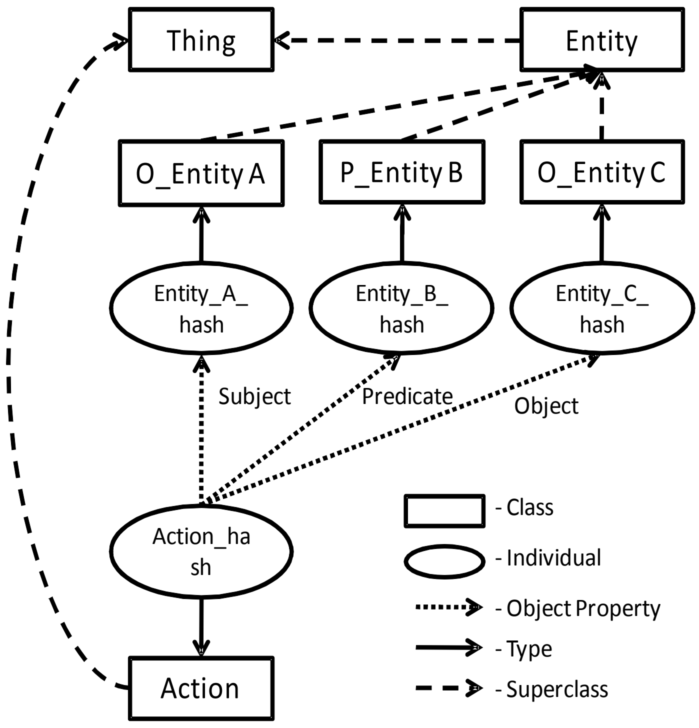


Figure 5.6:Individual relation.

The individual of type action is related to individuals of the type entity, by the object properties subject, predicate and object [OPR04] (figure 5.6).

This relation represents the simplest sentence representation, a sentence with a noun related by the predicate with other noun or adjective.

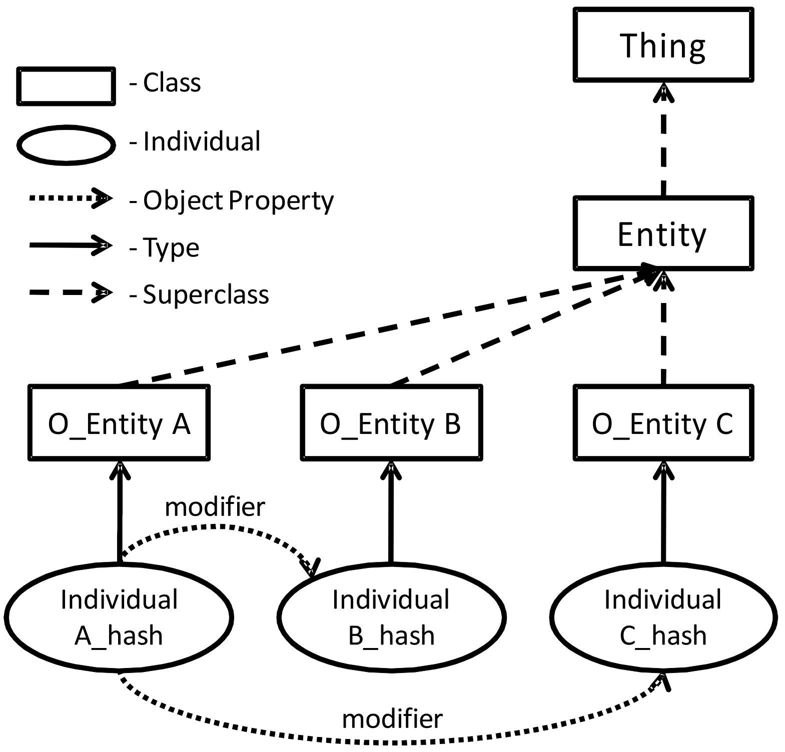


Figure 5.7:Individual modifiers.

Some tuple elements have modifiers; they provide extra information about the entity and are essential to keep sentence semantics. This leads to improve the ontological sentence representation and overall ontology semantics.

Every modifier element is also a subclass of entity. One individual can have one or more modifiers to an existing entity individual, related by the functional property named *modifier*. Each individual of type entity can have one or more modifiers (figure 5.7).

Besides knowing the Entity individual name, representation in the sentence and relation to other Entity individuals, no extra information is provided; it's time to start enhancing knowledge. Every time a new individual of type Entity is created, the enhancement process is attempted.

The first footstep is *Named Entity Disambiguation*. We already run AlchemyAPI to find anfd create Named Entity Types in Concept class, but Alchemy provides much more, it can expose semantic hidden content .

After running AlchemyAPI, for each tuple element, the NER XML return is one of 3 possible:

- NER doesn't return nothing, finds no entity
- NER process returns only the Named Entity Type and solves co-reference.
- NER process returns the Named Entity type, disambiguation information and solves co-reference.

If the NER process doesn't return information, no named entity was found, and at the moment no information enhancement is possible, no more individuals are created. With future improvements, AlchemyAPI might provide results.

[Named Entity Type, Named Entity Disambiguation, Named Entity]
[Person PersonX person]
[Company CompanyX company]
[Person PersonX He]

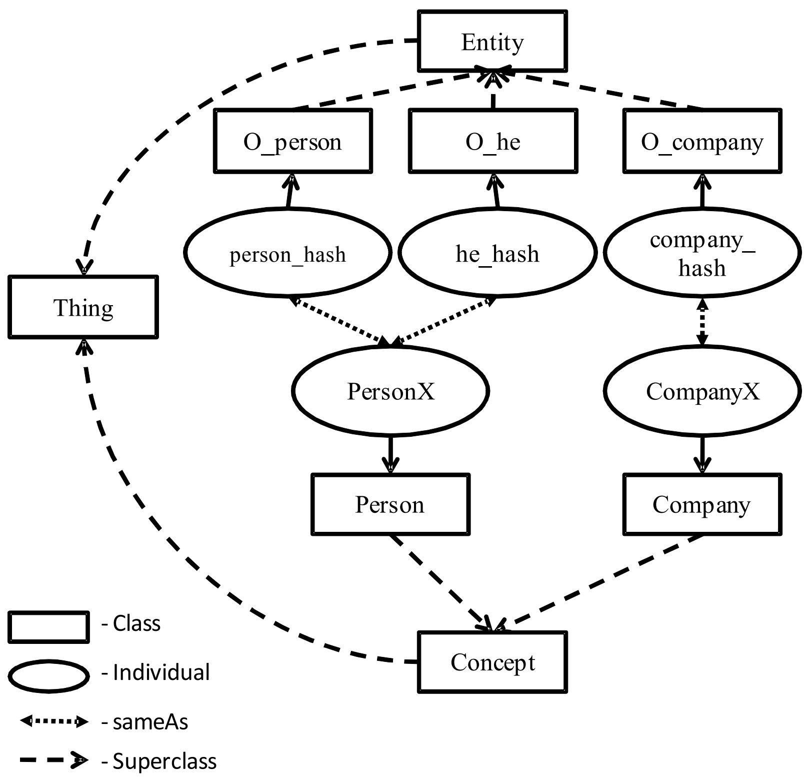


Figure 5.8: Entity desambiguation.

If the NER process returns the Named Entity Type, with co-reference solved, but no disambiguation information is available, a new individual is create with the same name as the Tuple element (and in case of a co-referring the resolution name) and has the same superclass or type as the NET class. The individual is acknowledged Concept individual, as shown in Figure 5.8.

The built-in OWL property *sameAs* links a Concept individual to an Entity individual. The sameAs statement indicates that two URI references actually refer to the same thing: the individuals have the same "identity" or meaning.

The return with the most appealing information is the third, besides the Named Entity type, it also provides disambiguation information. The new individual is created with the same name as the disambiguated name and has the same superclass or type as the NET class. The OWL property sameAs links the created Concept individual to the equivalent Entity individual.

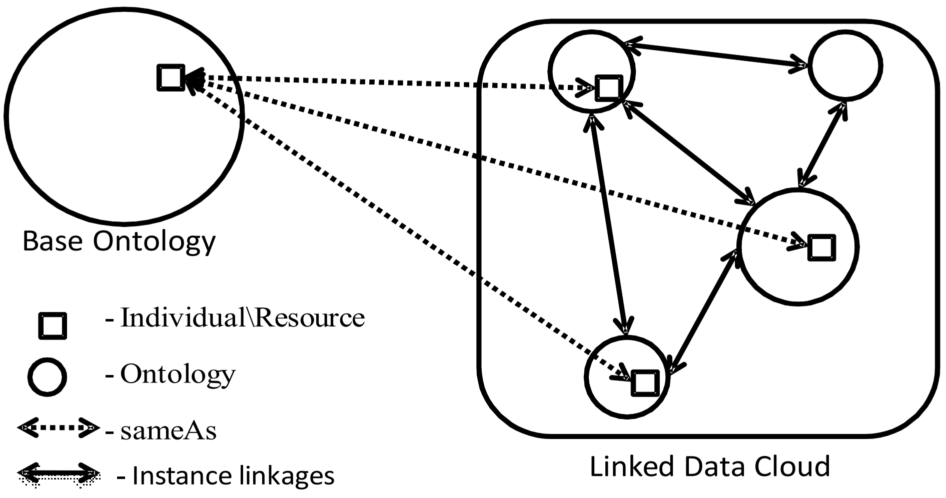


Figure 5.9: Linked Open Data resource.

Whenever an entity is successfully disambiguated, Linked Data resource location is built-in in API response. The desambiguation includes "sameAs" RDF links to Identifiable Resources (Data Sources) in the Linked Data cloud (Figure 5.9).

[Named Entity Type, Named Entity Disambiguation, Named Entity]
[EntT EntX ent]
[EntT EntX entx]

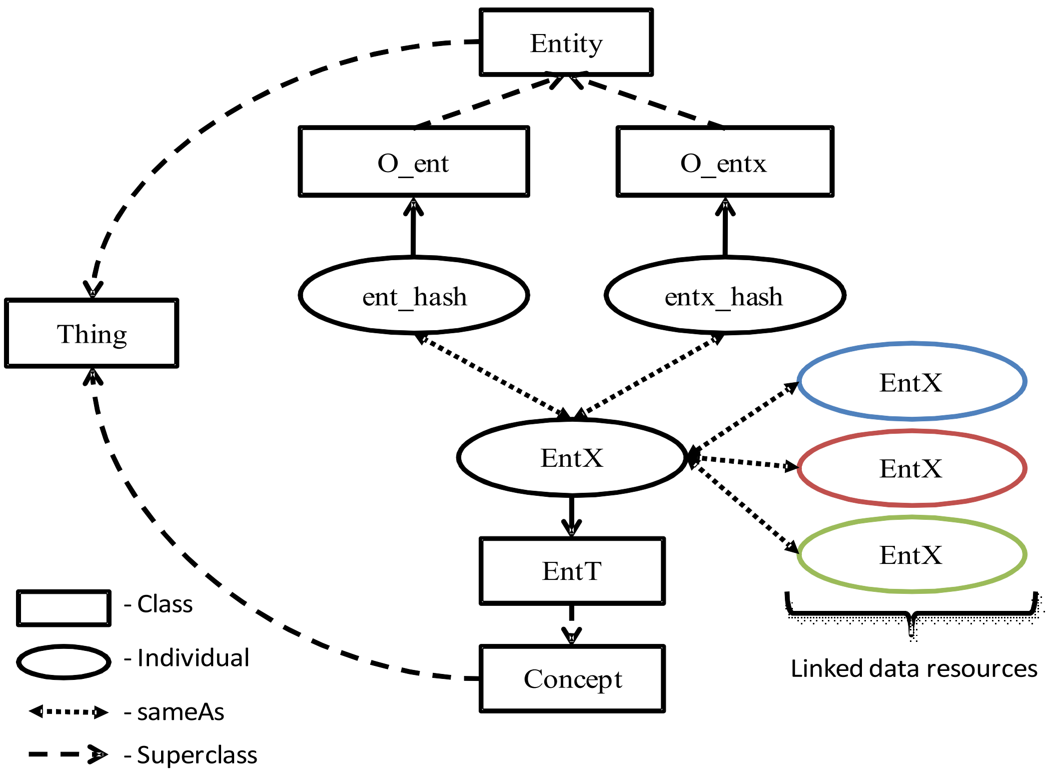


Figure 5.10: Linked Open data resource desambiguation.

The Concept individual is connected to the Linked Data cloud resource, by slash[51] URI, as a way to expand the current knowledge database.

The inclusion of the link to the LOD resource in the Concept individual allows the reuse of well know mix of vocabularies available in the semantic web (Figure 5.10), adapting the data to the application needs, allowing further expansion of the application usefulness.

The LOD resources have useful data to integrate in the current KD. SPARQL Protocol and RDF Query Language (SPARQL) are to the semantic Web as SQL is to a relational database. With Jena and ARQ support, sophisticated queries against distributed LOD resources or any other RDF databases can be done, to capture fragments[52] of information and integrate them in the knowledge domain, the system is bonded.

Using a custom query is built in the Sparql language, taking into account the domain, KD needs and LOD resource namespace, endpoint[53] and Jena ARQ API. The data acquired from the query is integrated into the Concept individual, while keeping the original namespace unchanged, since the LOD guidelines state:

- Do not define new vocabularies from scratch.
- Make use of other people terms.

One or more Concept individual links to LOD resources are queried.

The query result and relations to the individual are integrated in the base ontology, keeping original namespaces as possible. New relations are created when needed. As example a LOD resource regarding a person won't provide the grandfather relation, but will when possible provide father. To obtain the grandfather relation, the query will search for the father of the resource father. The grandfather resource will keep original namespace, but grandfather property will keep the base ontology namespace.

5.3 Conclusions

An automatic process for ontology creation and enhancement was presented. After applying the process to tuples, who can be created from full texts in documents, a representation in OWL is created and merged with enriched data, obtained from named entity resolution, co-reference and alias resolution and LOD inclusion. The OWL representation provides a pragmatic interpretation of its content.

The reasoner engine allows the queries used with the model to return not only those statements that were present in the original data representation, but also additional statements than can be derived from the data using the rules or other inference mechanisms implemented by the reasoner.

51 303 redirect type, used when the RDF and HTML convey the same information in different forms.

52 Triples.

53 Standard *SPARQL* protocol service as defined in the SPROT specification.

6 Test Case

This chapter provides a working sample of the tackled problem, helping in the understanding and interpretation of the problem.

A basic sentence can be reduced to three basic elements: subject, predicate and complement, similar to the tuple <subject, predicate and object>. Subject usually indicates a person, object or idea, executed by the verb. The predicate refers the action; it is the verb of the sentence and is the relation between subject and complement. Complement also know as object in the tuple indicates the element affected by the action, i.e., the predicate.

Let's take as example the following sentence "Bill Gates was chairman of Microsoft", the relation with name "was", relates subject "Bill Gates", with complement "chairman". The complement has also a modifier, "president".

When applied to the sentence the algorithm for tuple extraction, the result will be the following:

- Subject – "Bill Gates"
- Predicate – "be"
- Object - "chairman", with modifier "Microsoft"

Now starts the project related to this document, with the base ontology already created, the results are:

The resulting action would store the tuple subject, predicate and object.

```
<sample:Action rdf:about="&sample;Action_111bd81a-97ff-45fd-9512-5b2cead9ba33">
    <rdf:type rdf:resource="&sample;Action"/>
    <sample:object rdf:resource="&sample;Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231"/>
    <sample:subject rdf:resource="&sample;chairman_931ea3e6-b50d-428a-86ba-2763c220eb29"/>
    <sample:predicate rdf:resource="&sample;be_46ea77c2-3d30-44a8-8a6c-e815ff47af5b"/>
</sample:Action>
```

The subject, predicate, object and modifier classes result would be:

```
    <owl:Class rdf:about="&sample;O_Bill_Gates">
            <rdfs:subClassOf rdf:resource="&sample;Entity"/>
    </owl:Class>

    <owl:Class rdf:about="&sample;O_chairman">
```

```
        <rdfs:subClassOf rdf:resource="&sample;Entity"/>
</owl:Class>

<owl:Class rdf:about="&sample;P_be">
        <rdfs:subClassOf rdf:resource="&sample;Entity"/>
</owl:Class>

<owl:Class rdf:about="&sample;O_Microsoft">
        <rdfs:subClassOf rdf:resource="&sample;Entity"/>
</owl:Class>
```

The individuals generated are:

```
<owl:Thing rdf:about="&sample;Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231">
<rdf:type rdf:resource="&sample;O_Bill_Gates"/>
<rdfs:label xml:lang="EN">Bill Gates</rdfs:label>
<owl:sameAs rdf:resource="&sample;BillGates"/>
</owl:Thing>

<owl:Thing rdf:about="&sample;chairman_931ea3e6-b50d-428a-86ba-2763c220eb29">
<rdf:type rdf:resource="&sample;O_chairman"/>
<rdfs:label xml:lang="EN">chairman</rdfs:label>
<sample:modifier rdf:resource="&sample;Microsoft_46ea77c2-3d30-44a8-8a6c-e815ff47af5b"/>
</owl:Thing>

<owl:Thing rdf:about="&sample;be_46ea77c2-3d30-44a8-8a6c-e815ff47af5b">
<rdf:type rdf:resource="&sample;P_be"/>
<rdfs:label xml:lang="EN">be</rdfs:label>
</owl:Thing>

<owl:Thing rdf:about="&sample;Microsoft_46ea77c2-3d30-44a8-8a6c-e815ff47af5b">
<rdf:type rdf:resource="&sample;O_Microsoft"/>
<rdfs:label xml:lang="EN">Microsoft</rdfs:label>
<owl:sameAs rdf:resource="&sample;Microsoft"/>
</owl:Thing>
```

The subject, predicate, object and modifers are created. the relations between the are kept, and connections of type sameAs to concept individuals are created. The next step is the named entity recognition, concept creation and enrichment of data. First lets check the OWL code for the concepts.

```
<owl:Class rdf:about="&sample;Company">
<rdfs:subClassOf rdf:resource="&sample;Concept"/>
</owl:Class>

<owl:Class rdf:about="&sample;Person">
<rdfs:subClassOf rdf:resource="&sample;Concept"/>
</owl:Class>
```

The concept created refer to the person “Bill Gates” and the company “Microsoft”. Each concept contain one or more individuals, related to entities. The Linked Open Data extracted, depends on the endpoint used and the information requested by the query. The query is adapted to the required information, so to improve the gathered element from each LOD resource, a special query is created for each named entity type.

Using Dbpedia endpoint[54], resourceNamespace with name equal to the LOD resource name and the following query, constructed for every concept of type "Company":

```
PREFIX ont: <http://dbpedia.org/ontology/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT *
WHERE
{
resourceNamespace ont:foundationPerson ?founder .
resourceNamespace ont:industry> ?industry .
resourceNamespace ont:product> ?product .
resourceNamespace ont:location ?location .
resourceNamespace prop:slogan> ?slogan .
}
```

The combination of the AlchemyAPI, plus the query result, when inserted into the ontology, for the example of the company "Microsoft"is the following:

```
<owl:Thing rdf:about="&sample;Microsoft">
<rdf:type rdf:resource="&sample;Company"/>
<ontology:foundationPerson rdf:resource="&resource;Bill_Gates"/>
<ontology:foundationPerson rdf:resource="&resource;Paul_Allen"/>
<ontology:industry rdf:resource="&resource;Computer_software"/>
<ontology:industry rdf:resource="&resource;Consumer_electronics"/>
<ontology:industry rdf:resource="&resource;Video_game_console"/>
<ontology:product rdf:resource="&resource;Microsoft_Game_Studios"/>
<ontology:product rdf:resource="&resource;Zune"/>
<ontology:product rdf:resource="&resource;Microsoft_Windows"/>
<ontology:product rdf:resource="&resource;Microsoft_Office"/>
<ontology:product rdf:resource="&resource;Microsoft_Servers"/>
<ontology:product rdf:resource="&resource;Microsoft_Visual_Studio "/>
<ontology:product rdf:resource="&resource;Microsoft_Expression_Studio"/>
<ontology:product rdf:resource="&resource;Microsoft_Dynamics"/>
<ontology:product rdf:resource="&resource;Windows_Live"/>
<ontology:product rdf:resource="&resource;Bing_(search_engine)"/>
<ontology:product rdf:resource="&resource;Windows_Phone"/>
<ontology:location rdf:resource="&resource;Redmond,_Washington"/>
<ontology:product rdf:resource="&resource;Windows_Phone"/>
<prop:slogan>Your potential. Our passion.</prop:slogan>
<owl:sameAs rdf:resource="http://dbpedia.org/resource/Microsoft"/>
<owl:sameAs rdf:resource="http://rdf.freebase.com/ns/guid.9202a8c04000641f8000000000026344"/>
<owl:sameAs rdf:resource="http://umbel.org/umbel/ne/wikipedia/Microsoft"/>
<owl:sameAs rdf:resource="http://sw.opencyc.org/concept/Mx4rvVjegpwpEbGdrcN5Y29ycA"/>
<owl:sameAs rdf:resource="http://mpii.de/yago/resource/Microsoft"/>
<owl:sameAs rdf:resource="http://www.crunchbase.com/company/microsoft"/>
<owl:sameAs rdf:resource="http://cb.semsol.org/company/microsoft.rdf"/>
</owl:Thing>
```

The query created for type person is:

```
PREFIX ont: <http://dbpedia.org/ontology/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT *
WHERE
{
resourceNamespace ont:foundationPerson ?founder .
resourceNamespace ont:birthDate> ?birthdate .
resourceNamespace ont:birthPlace> ?birthPlace .
```

54 http://dbpedia.org/sparql

```
resourceNamespace ont:occupation ?occupation .
resourceNamespace prop:name> ?name .
}
```

The combination of the AlchemyAPI, plus the query result, when inserted into the ontology, for the example of the person “Bill Gates”is the following:

```
<owl:Thing rdf:about="&sample;Bill_Gates">
<rdf:type rdf:resource="&sample;Person"/>
<ontology:birthDate> 1955-10-28  </ontology:birthDate>
<ontology:birthPlace rdf:resource="&resource;Seattle,_Washington"/>
<ontology:occupation rdf:resource="&resource;Bill___Melinda_Gates_Foundation”/>
<ontology:occupation rdf:resource="&resource;Cascade_Investment”/>
<ontology:occupation rdf:resource="&resource;Microsoft”/>
<ontology:occupation rdf:resource="&resource;Berkshire_Hathaway”/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__Chairman_of_Microsoft”/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__Co-
Chair_of_Bill___Melinda_Gates_Foundation”/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__CEO_of_Cascade_Investment”/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__Director_of_Berkshire_Hathaway”/>
<prop:name>Bill Gates </prop:name>
<prop:name>Gates, William Henry, III  </prop:name>
<owl:sameAs rdf:resource="http://www.microsoft.com/presspass/exec/billg/default.mspx”/>
<owl:sameAs rdf:resource="http://dbpedia.org/resource/Bill_Gates”/>
<owl:sameAs rdf:resource="http://rdf.freebase.com/ns/guid.9202a8c04000641f8000000000009e99”/>
<owl:sameAs rdf:resource="http://umbel.org/umbel/ne/wikipedia/Bill_Gates”/>
<owl:sameAs rdf:resource="http://sw.opencyc.org/concept/Mx4rvYwpvpwpEbGdrcN5Y29ycA”/>
<owl:sameAs rdf:resource="http://mpii.de/yago/resource/Bill_Gates”/>
<owl:sameAs rdf:resource="&sample;Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231"/>
</owl:Thing>
```

Both individual share the same namespace as the base ontology, they also keep the LOD resource's links, very useful for any future update or change. Also each new resource, if it's a member of the LOD initiative, it can be used to further improve the knowledge representation, it’s only necessary to adapt or create new queries. From a complete view of the example consult Annex C.

The ontology tools automatically infer new knowledge thanks to the reasoner and the entailment regimes. The complete list of the example can be seen in Annex D.

The knowledge resulting from the representation is huge.

We know from the inferred results that the “Bill Gates” referred in the sentence has the real name “William Henry Gates the III”, was born “1955-10-08” in “Seattle Washington”, one of his occupations is CEO of “Cascade Investment”.

We know from the representation that the “Microsoft” referred in the sentence, has as founder “Paul Allen” and “Bill Gates”, the same “Bill Gates” referred in the text. It created products such as “Zune” and “Microsoft Windows”. The industry attention is focused in “Computer Software”, “Video Game Consoles” and “Consumer Electronics”.

From a simple sentence lots of useful information was gathered, all without human intervention. As more sentences are added, more relations are created and more information generated. This is the power of the Semantic Web.

7 Conclusions

This chapter presents the conclusions drawn from the project. The retrospective of the work brings together scope, objectives and results. It also presented an analysis of the accomplished objectives originally proposed and where further work should be done to improve this project.

7.1 Restrospective

The market competition put a big strain into current evolution and quality of services. A motionless company is a fading company. Research is a building block of every vigorous business, but confidential work, with no external interaction is a past trend, especially in IT. The world of retail is one of the drivers of evolution in the area, and therefore, also one of the most demanding.

As the firms look to advance their technology, keeping input\output streams of external and internal ideas, lead to unidentified and improbable research paths. Open and networked innovation is the key building block for a healthy research environment.

Wipro Retail by nature tries to develop their own support tools, from ground up or by expanding market solutions. Due to the research nature, little is done into the subject and no similar systems exist.

The realization proved to be very challenging, due to the technological and novelty of the associated concepts, but the inclusion in Wipro Retail, work conditions and individual support, gave way to personal growth and good work realization.

7.2 Objective satisfaction

This report objective was materialized as part of *Prymas*, a knowledge managing and sharing solution. Using some agile methodologies, *Prymas* gathered a lot of attention and receptivity inside and outside of the company.

During development, parallel objectives set were also meet; the knowledge sharing led to unknown inputs, both internally and externally, mainly in research directions, work focus and available tools.

The communication lines lead us from classic knowledge representation systems into ontologies and Semantic Web data manipulation. It directed us into automatic ontology creation and the Linked Open Data initiative.

Discussions with multiple researchers, especially those who pursue related subjects helped outline the bleeding edge research state and lead to access to personal tools, experimentations and results. Personal opinions were also very useful when decisions were taken or alternatives required.

Even after the system and architecture definition, constant community input guided to improvements. Still the primary objective was the study of the state of art, both in technological and scientific approaches, to implement a proof of concept method for automatic acquire, structure and enhancement of information, in order to create a knowledge representation. The objective is fully meet; a coarse model is implemented.

From the syntactical and semantic analysis a representation is created, <resource, attribute, value> tuple, with modifiers, who will lead to the creation of concepts, entities and taxonomy structure.

The instance creation creates a hypothesis as its end result. The instances are set of seemingly unrelated facts, but when instances and classes are created in OWL, and suffer a semantic enhancement, the knowledge goes beyond a set of seemingly unrelated facts, it is connected.

When the inference engine is coupled with the query engine, the amount of inferred knowledge is very satisfying, unidentified relations in the documents are made available. But were the system really shines is with linked data integration.

The amount and quality of information provided by LOD is unmatched, the ontology incorporation straightforward. This gives the ontology new related and structured knowledge, outside the documentation domain, to incorporate and query as required.

Taking into the account the project complexity, author previous knowledge and experience, time constrains, the creation, interrogation and enhancement of an ontology, created from a document, while enlisting internal and external opinion was fully meet.

The projects gave a relevant view of the state of semantic web, current applications and futures uses, essencial to the company future research.

As a side note the blog got good traffic (blog of the week, with a good daily average) and enrolment from community, who is very dispersed around the internet.

7.3 Future Work

Natural language processing, is still limited, especially when the processing is done on a semantic level, the results can be fuzzy, even incorrect.

The tuple format loses some semantic information and context, compared to the original sentence. Co-reference, word sense disambiguation and alias resolution produce decent results, but incomplete. The Semantic Web already gave the first steps, the concept is well defined, but the tools and support data is still very immature. There is also a good degree of uncertainty on how to explore SW.

The current tools, methods and knowledge representation are very coarse. With time methods can be reviewed and improved and KR adapted. The tools and research approaches are improving at a outstanding pace, two years ago, it would be impossible to reproduce the same work, one year ago the results would be much more incomplete.

Conclusions

The tools and research approaches are improving at a outstanding pace: two years ago- would be impossible to reproduce the same work; one year ago- the results would be very feeble.

Stronger and better relations between ontology instances lead to more knowledge and impressive inferred results. Combined with better queries, the efficiency can improve.

Still the best way to improve is to test the system under real world condition, with real users, and a constant stream of feedback, to fine tune the system.

References

[MUC95] 1995 Message Understanding Conference Proceedings (MUC-5), May 2010, http://cs.nyu.edu/cs/faculty/grishman/muc6.html

[MUC01] 2001 Message Understanding Conference Proceedings (MUC-7), May 2010, http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_toc.html

[AMS05] A.U. Michael R. Solomon, "Knowledge Management as Competitive Advantage in the Textile and Apparel Value Chain," 2005.

[ADE09] Aaron Defazio, Natural language question over triple knowledge bases, 2009.

[AAK06] Agbago, Akakpo; Kuhn, R. and Foster, G., Truecasing for the Portage System. Proc. Of International Conference on Recent Advances in Natural Language Processing, 2006.

[ALAPI] AlchemyAPI, April 2010, http://www.alchemyapi.com.

[AED10] AlchemyAPI Entity Desambiguation, April 2010, http://www.alchemyapi.com/api/entity/disamb.html

[AET10] AlchemyAPI Entity Types, April 2010, http://www.alchemyapi.com/api/entity/types.html

[BMC02] B. McBride, Jena IEEE Internet Computing, 2002.

[TBL01] Berners-Lee, Tim; James Hendler and Ora Lassila, "The Semantic Web". Scientific American Magazine. 2001.

[BRT98] Bray, T., RDF and Metadata, 1998.

[CHW03] Chesbrough, H.W., Open Innovation: The new imperative for creating and profiting from technology. Boston: Harvard Business School Press, 2003.

[CSS92] Coates-Stephens, Sam,The Analysis and Acquisition of Proper Names for the Understanding of Free Text. Computers and the Humanities, 1992.

[DLF07] D. Rusu, L. Dali, B. Fortuna, M. Grobelnik, and D. Mladenić, Triplet extraction from sentences, Knowledge Creation Diffusion Utilization, 2007.

[DOL01] DAML+OIL Reference Description. April 2010 Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. W3C Note 18 December 2001.

[DPS07] David Nadeau, Peter D. Turney, and Stan Matwin. Semi-supervised named entity recognition: Learning to recognize 100 entity types with little supervision, 2007.

[DCJ05] Dave Reynolds, Carol Thompson, Jishnu Mukerji, and Derek Coleman, An Assessment of RDF/OWL Modelling, HPL Technical Report 2005-189, 2005.

[DMB02] Dimitrov, Marin; Bontcheva, K.; Cunningham H and Maynard, D., A Light-weight Approach to Co-reference resolution for Named Entities in Text. In Discourse Anaphora and Anaphor Resolution Colloquium, 2002.

[FAF02] F. Arvidsson and A. Flycht-Eriksson, Ontologies I, 2002.

[FACTR] Fact++ reference, April 2010, http://owl.man.ac.uk/factplusplus/.

[FAA03] Freitas, A. A. A survey of evolutionary algorithms for data mining and knowledge discovery. In Advances in Evolutionary Computing: theory and Applications, A. Ghosh and S. Tsutsui, Eds. Natural Computing Series, 2003.

[FEB82] Frank B. Ebersole, TI: Stalking the Rigid Designator, Philosophical Investigations, 1982.

[GW71] Gaisi Takeuti, W. M. Zaring, Introduction to Axiomatic Set Theory, Springer GTM 1, 1971.

[GAW93] Gale, W. A., K. W. Church, and D. Yarowsky, A method for disambiguating word senses in a large corpus. Computers and the Humanities, 1993.

[GDA93] Gary Davies, Is Retailing What The Dictionaries Say It Is, Manchester Business School, Manchester, International Journal of Retail & Distribution Management, Volume 21, Número 2, 1993.

[GNN87] GENESERETH, M.R., Nilsson, N., Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers, 1987.

[GCC05] Gliozzo, A., C. Giuliano, and C. Strapparava, Domain kernels for word sense disambiguation. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, 2005.

[GTR93] Gruber, T.R., A translation approach to portable ontologies, 1993.

[HCU04] Hamish Cunningham, Information Extraction, Automatic., Encyclopedia of Language and Linguistics, 2006.

[HFE05] Huang, Fei, Multilingual Named Entity Extraction and Translation from Text and Speech, 2005.

[IPP03] Ian Horrocks, Peter F. Patel-Schneider, Frank van Harmelen, Journal of Web Semantics, 2003.

[JIC03] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson, The Jena Semantic Web Platform: Architecture and Design, HP Laboratories Technical Report HPL, 2003.

[JLE04] J. Leskovec, "Learning Sub-structures of Document Semantic Graphs for Document Summarization", Linguistic Analysis, 2004.

[BDS91] J. S. Baron, B. Davies, D. Swindley, Dictionary of Retailing, Macmillan, 1991.

[JAK03] Jeen Broekstra, Arjohn Kampman, Inferencing and Truth Maintenance in RDF Schema Exploring a naive practical approach, 2003.

[ARQ10] Jena ARQ tutorial, March 2010, http://jena.sourceforge.net/ARQ/Tutorial/.

[JNI10] Jena Inference API, May 2010, http://jena.sourceforge.net/inference/.

[JNOWL] Jena Ontology API, May 2010, http://jena.sourceforge.net/ontology/

[JTC05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning, Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In proceedings of the 43nd Annual Meeting of the Association for Computational, 2005.

[KRP82] Kripke, Saul,Naming and Necessity. Harvard University Press, 1982.

[LLW05] Lacy, Lee W., OWL: Representing Information Using the Web Ontology Language. 2005.

[LXM04] Li, Xin., Morie, P. and Roth, D., Identification and Tracing of Ambiguous Names: Discriminative and Generative Approaches. Proc. National Conference on Artificial, 2004.

[MKO06] M. Konchady. Text Mining Application Programming. Charles River Media; 1 edition, 2006.

[MMC08] Marie-Catherine de Marne and Christopher D. Manning, Stanford typed dependencies manual September 2008, 2008.

[MGY03] Mann, Gideon S. and Yarowsky, D., Unsupervised Personal Name Disambiguation. Proc. Conference on Computational Natural Language Learning, 2003.

[MBR00] McBride, Brian, Jena: Implementing the rdf model and syntax specification. Technical report, Hewlett Packard Laboratories, 2000.

[NDT05] Nadeau, David and Turney, P., A Supervised Learning Approach to Acronym Identification. Proc. Canadian Conference on Artificial Intelligence, 2005.

[FUN04] OWL features, April 2010, http://www.w3.org/TR/2004/REC-owl-features-20040210/#FunctionalProperty.

[OPR04] OWL Properties, April 2010, http://www.w3.org/TR/owl-ref/#Property.

[RFOWL] OWL W3C reference, April 2010, http://www.w3.org/2004/OWL/.

[PPB04] P. Buitelaar, P. Cimiano, and B. Magnini, Ontology Learning from Text: An Overview, 2004.

[PMA04] Pasca, Marius, Acquisition of Categorized Named Entities for Web Search. In Proc. Conference on Information and Knowledge Management, 2004.

[PELLT] Pellet Inference Engine, April 2010, http://clarkparsia.com/pellet/.

[PHM08] Philip H. Mitchell, Discovery-Based Retail: Unlock Your Store's Potential , Bascom Hill Publishing Group, 2008.

[PTK01] Poibeau, Thierry and Kosseim, L., Proper Name Extraction from Non-Journalistic Texts. In Proceedings of Computational Linguistics in the Netherlands, 2001.

[RLF91] Rau, Lisa F., Extracting Company Names from Text. Proc. Conference on Artificial Intelligence Applications of IEEE, 1991.

[RDS04] RDF Grammar reference, April 2010, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[RDF04] RDF Syntax reference, April 2010, www.w3.org/TR/REC-rdf-syntax/

[RPV04] Roberto Navigli e Paola Velardi. Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites, 2004.

[RBE04] Russell, Bertrand, Knowledge by Acquaintance and Knowledge by Description, In The Problems of Philosophy, 2004.

[SMJ86] Salton, G. and McGill, M. J. Introduction to Modern Information Retrieval, McGraw-Hill, Inc. 1986.

[SDM05] Sánchez, David and Moreno, A. (2005) Web Mining Techniques for Automatic Discovery of Medical Knowledge. Proc. Conference on Artificial Intelligence in Medicine.

[SSN04] Sekine, Satoshi and Nobata, C. (2004) Definition, dictionaries and tagger for Extended Named entity Hierarchy. Proc. Conference on Language Resources and Evaluation.

[WOT04] Semantic Web Web ontology requirerments, April 2010, http://www.w3.org/TR/webont-req.

[RDATA] Semantic Web Data Access, April 2010, http://www.w3.org/2001/sw/DataAccess.

[TBL98] Semantic Web design issues, April 2010, http://www.w3.org/DesignIssues/Semantic.html.

[RDATA] Semantic Web Data Access, April 2010, http://www.w3.org/2001/sw/DataAccess.

[SLB10] Shah, R., Lin, B., Gershman, A. & Frederking, R., SYNERGY: A Named Entity Recognition System for Resource-scarce Languages such as Swahili using Online Machine Translation. In Proceedings of International Conference on Language Resource and Evaluation Workshop on African Language Technology, 2010.

[BAC97] Sir Francis Bacon. Religious Meditations, Of Heresies. 1597.

[SJF84] Sowa, John F. (1984), Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA, 1984.

[SBA91] Sowa, John F. Borgida, Alexander, Principles of Semantic Networks: Explorations in the Representation of Knowledge, 1991.

[SPA10] Sparql Query system, April 2010, http://www.w3.org/TR/2010/WD-sparql11-query-20100126/.

[PPA10] Sparql Property Paths, April 2010, http://www.w3.org/TR/2010/WD-sparql11-property-paths-20100126/.

[ENT10] SPARQL Entailmente regimes, April 2010, http://www.w3.org/TR/2010/WD-sparql11-entailment-20100126.

[SPQLP] SPARQL protocol reference, April 2010, www.w3.org/TR/rdf-sparql-protocol/.

[RSPQL] SPARQL reference, April 2010, http://www.w3.org/TR/rdf-sparql-query/.

[SRI99] Srihari, Rohini and Li, W., Information Extraction Supported Question Answering. Proc. Text Retrieval Conference, 1999.

[URI98] T. Berners-Lee, R. Fielding and L. Masinter, RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax, 1998.

[TGR93] T. Gruber, A translation approach to portable ontology specifications, 1993.

[GAT10] The gate team, Developing Language Processing Components with GATE Version 5 (a User Guide) For GATE version 5.0,2009

[WLW05] Wang, Lee, Wang, C., Xie, X., Forman, J., Lu, Y., Ma, W.-Y. and Li, Y, Detecting Dominant Locations from Search Queries. Proc. International ACM SIGIR Conference, 2005.

[WDR53] W. D. Ross, Aristotle's Metaphysics, 1953.

[WSM04] W. Scott Means, XML in a Nutshell, Third Edition, O'Reilly Media, 2004.

[XPA99] XML XPATH reference, April 2010, http://www.w3.org/TR/xpath.

[XML08] XML reference, April 2010, http://www.w3.org/TR/2008/REC-xml-20081126/.

[YQG07] Yin, S., Qiu, Y., and Ge, J., Research and Realization of Text Mining Algorithm on Web. In Proceedings of the 2007 international Conference on Computational intelligence and Security Workshops, 2007.

[YTH04] Youyong Zou, Tim Finin, and Harry Chen, F-OWL: an Inference Engine for the Semantic Web, 2004.

Annex A: Named Entity Recognition

First transcript, from What is Prymas[55]:

"A report in the Wall Street Journal indicates that the Facebook, along with MySpace, Digg, and half dozen other social-networking sites, have been sharing users' personal data with advertisers, without users' knowledge or consent.

After some recent vouched fears, some have been realized. All of those promises that sites such as MySpace and Facebook have made regarding the safety of user personal information has been proven to be untrue.

After Facebook release of OpenGraph, it made promisses of safety and privacy to users. The sent data included user names or ID numbers tied to personal profiles being viewed when users click on ads. Facebook goes as far as providing the user name and complete profile to the advertiser, who's add was clicked.

This action was first noticed (PDF) in August 2009 by researchers from Worcester Polytechnic Institute and AT&T Labs, who brought it up with the sites in question. Until the Wall Street Journal contacted them the action continued."

For a full report of the case complexity go here.

Second transcript:

"Therese Rein was born in 1958. Therese Rein is a Australian Business woman. Therese Rein is the wife of Kevin Reid."

Alchemy results transcript 1:

Company: Facebook, Myspace, Digg, AT&T labs

Print media: Wall Street Journal

Facility: Worcester Polytechnic Institute

Links to disambiguate in linked data were given.

55 http://whatisprymas.wordpress.com/2010/05/22/facebook-and-friends-caught-red-handed/

Alchemy results transcript 2:

Person: Kevin Rein, Therese Rein

No links to disambiguate in linked data were given.

Open Calais Results transcript 1:

Company: Wall Street Journal,AT&T Labs,Facebook,MySpace

Organization: Worcester Polytechnic Institute

Product: OpenGraph

PublishedMedium: the Wall Street Journal

Technology: PDF

Open Calais Results transcript 2:

Person: Kevin Rein, Therese Rein

Position: Business woman

Annex B: Base Ontology

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY sample "http://www.example.com/ontologies/sample.owl#" >
]>

<rdf:RDF xmlns="file:/C:/Documents%20and%20Settings/ccost/Desktop/test/sample1.owl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:sample="http://www.example.com/ontologies/sample.owl#">
   <owl:Ontology rdf:about=""/>

   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   // Object Properties
   ///////////////////////////////////////////////////////////////////////////////////////
    -->

   <!-- http://www.example.com/ontologies/sample.owl#modifier -->
   <owl:ObjectProperty rdf:about="&sample;modifier">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
      <rdfs:domain rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>

   <!-- http://www.example.com/ontologies/sample.owl#object -->
   <owl:ObjectProperty rdf:about="&sample;object">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="&sample;Action"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>
   <!-- http://www.example.com/ontologies/sample.owl#predicate -->
   <owl:ObjectProperty rdf:about="&sample;predicate">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="&sample;Action"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>

   <!-- http://www.example.com/ontologies/sample.owl#subject -->
   <owl:ObjectProperty rdf:about="&sample;subject">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="&sample;Action"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>

   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   // Classes
   ///////////////////////////////////////////////////////////////////////////////////////
    -->

   <!-- http://www.example.com/ontologies/sample.owl#Concept -->
            <owl:Class rdf:about="&sample;Concept">
            <rdfs:label rdf:datatype="&xsd;string">Action representation</rdfs:label>
      <rdfs:comment rdf:datatype="&xsd;string">Base class of Concept. Represents named entity types and relations
with entities</rdfs:comment>
            </owl:Class>


 <!-- http://www.example.com/ontologies/sample.owl#Action -->
 <owl:Class rdf:about="&sample;Action">
 <rdfs:label rdf:datatype="&xsd;string">Action representation</rdfs:label>
 <rdfs:comment rdf:datatype="&xsd;string">Base class of Action. Represents the tuple relations</rdfs:comment>
 </owl:Class>

 <!-- http://www.example.com/ontologies/sample.owl#Entity -->
 <owl:Class rdf:about="&sample;Entity">
 <rdfs:label rdf:datatype="&xsd;string">Entity representation</rdfs:label>
 <rdfs:comment rdf:datatype="&xsd;string">Base class of entity. Keeps every entity in a document</rdfs:comment>
 </owl:Class>

</rdf:RDF>
```

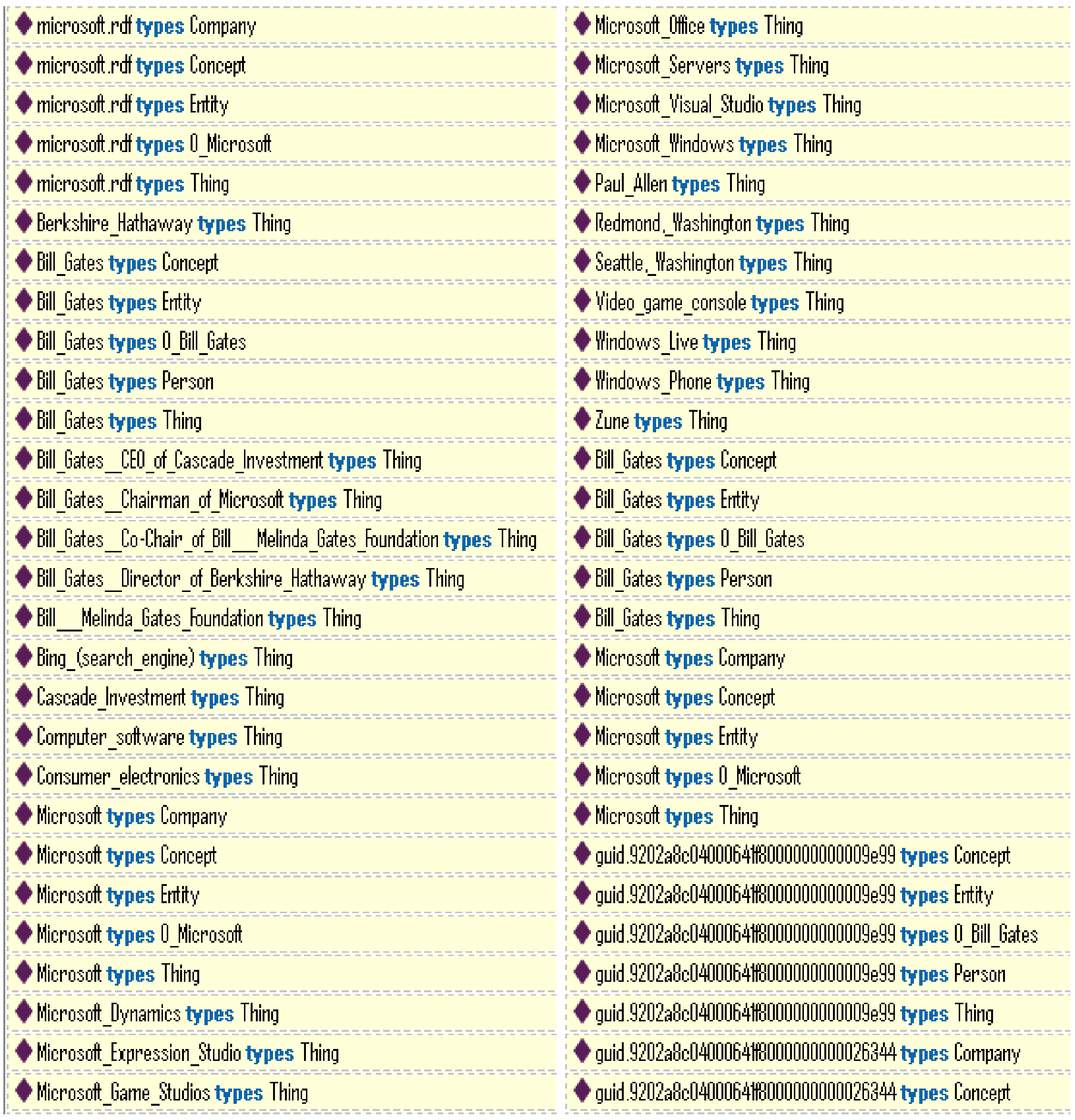
Annex C: Test Results 1

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY ontology "http://dbpedia.org/ontology#" >
 <!ENTITY resource "http://dbpedia.org/resource/" >
 <!ENTITY prop "http://dbpedia.org/property/" >
 <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
 <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
 <!ENTITY sample "http://www.example.com/ontologies/sample.owl#" >
]>

<rdf:RDF xmlns="file:/C:/Documents%20and%20Settings/ccost/Desktop/test/sample1.owl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:resource="http://dbpedia.org/resource/"
     xmlns:ontology="http://dbpedia.org/ontology#"
     xmlns:prop="http://dbpedia.org/property#"
     xmlns:sample="http://www.example.com/ontologies/sample.owl#">
   <owl:Ontology rdf:about=""/>

   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   // Object Properties
   ///////////////////////////////////////////////////////////////////////////////////////
    -->

   <!-- http://www.example.com/ontologies/sample.owl#modifier -->
   <owl:ObjectProperty rdf:about="&sample;modifier">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
      <rdfs:domain rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>
   <!-- http://www.example.com/ontologies/sample.owl#object -->
   <owl:ObjectProperty rdf:about="&sample;object">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="&sample;Action"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>

   <!-- http://www.example.com/ontologies/sample.owl#predicate -->
   <owl:ObjectProperty rdf:about="&sample;predicate">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="&sample;Action"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>

   <!-- http://www.example.com/ontologies/sample.owl#subject -->
   <owl:ObjectProperty rdf:about="&sample;subject">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="&sample;Action"/>
      <rdfs:range rdf:resource="&sample;Entity"/>
   </owl:ObjectProperty>

   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   // Classes
   ///////////////////////////////////////////////////////////////////////////////////////
    -->

         <!-- http://www.example.com/ontologies/sample.owl#Concept -->
      <owl:Class rdf:about="&sample;Concept">
      <rdfs:label rdf:datatype="&xsd;string">Action representation</rdfs:label>
      <rdfs:comment rdf:datatype="&xsd;string">Base class of Concept. Represents named entity types and
relations              with entities</rdfs:comment>
       </owl:Class>

 <!-- http://www.example.com/ontologies/sample.owl#Action -->
 <owl:Class rdf:about="&sample;Action">
 <rdfs:label rdf:datatype="&xsd;string">Action representation</rdfs:label>
      <rdfs:comment rdf:datatype="&xsd;string">Base class of Action. Represents the tuple
relations</rdfs:comment>
      </owl:Class>

   <!-- http://www.example.com/ontologies/sample.owl#Entity -->
      <owl:Class rdf:about="&sample;Entity">
      <rdfs:label rdf:datatype="&xsd;string">Entity representation</rdfs:label>
      <rdfs:comment rdf:datatype="&xsd;string">Base class of entity. Keeps every entity in a
document</rdfs:comment>
      </owl:Class>
    <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   // Instances
   ///////////////////////////////////////////////////////////////////////////////////////
    -->

<sample:Action rdf:about="&sample;Action_111bd81a-97ff-45fd-9512-5b2cead9ba33">
<rdf:type rdf:resource="&sample;Action"/>
<sample:object rdf:resource="&sample;Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231"/>
<sample:subject rdf:resource="&sample;chairman_931ea3e6-b50d-428a-86ba-2763c220eb29"/>
<sample:predicate rdf:resource="&sample;be_46ea77c2-3d30-44a8-8a6c-e815ff47af5b"/>
</sample:Action>

<owl:Class rdf:about="&sample;Company">
<rdfs:subClassOf rdf:resource="&sample;Concept"/>
```

```
</owl:Class>

<owl:Class rdf:about="&sample;Person">
<rdfs:subClassOf rdf:resource="&sample;Concept"/>
</owl:Class>

<owl:Class rdf:about="&sample;O_Bill_Gates">
<rdfs:subClassOf rdf:resource="&sample;Entity"/>
</owl:Class>

<owl:Class rdf:about="&sample;O_chairman">
<rdfs:subClassOf rdf:resource="&sample;Entity"/>
</owl:Class>

<owl:Class rdf:about="&sample;P_be">
<rdfs:subClassOf rdf:resource="&sample;Entity"/>
</owl:Class>

<owl:Class rdf:about="&sample;O_Microsoft">
<rdfs:subClassOf rdf:resource="&sample;Entity"/>
</owl:Class>

<owl:Thing rdf:about="&sample;Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231">
<rdf:type rdf:resource="&sample;O_Bill_Gates"/>
<rdfs:label xml:lang="EN">Bill Gates</rdfs:label>
<owl:sameAs rdf:resource="&sample;Bill_Gates"/>
</owl:Thing>

<owl:Thing rdf:about="&sample;chairman_931ea3e6-b50d-428a-86ba-2763c220eb29">
<rdf:type rdf:resource="&sample;O_chairman"/>
<rdfs:label xml:lang="EN">chairman</rdfs:label>
<sample:modifier rdf:resource="&sample;Microsoft_46ea77c2-3d30-44a8-8a6c-e815ff47af5b"/>
</owl:Thing>

<owl:Thing rdf:about="&sample;be_46ea77c2-3d30-44a8-8a6c-e815ff47af5b">
<rdf:type rdf:resource="&sample;P_be"/>
<rdfs:label xml:lang="EN">be</rdfs:label>
</owl:Thing>

<owl:Thing rdf:about="&sample;Microsoft_46ea77c2-3d30-44a8-8a6c-e815ff47af5b">
<rdf:type rdf:resource="&sample;O_Microsoft"/>
<rdfs:label xml:lang="EN">Microsoft</rdfs:label>
<owl:sameAs rdf:resource="&sample;Microsoft"/>
</owl:Thing>

<owl:Thing rdf:about="&sample;Bill_Gates">
<rdf:type rdf:resource="&sample;Person"/>
<ontology:birthDate> 1955-10-28  </ontology:birthDate>
<ontology:birthPlace rdf:resource="&resource;Seattle,_Washington"/>
<ontology:occupation rdf:resource="&resource;Bill___Melinda_Gates_Foundation"/>
<ontology:occupation rdf:resource="&resource;Cascade_Investment"/>
<ontology:occupation rdf:resource="&resource;Microsoft"/>
<ontology:occupation rdf:resource="&resource;Berkshire_Hathaway"/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__Chairman_of_Microsoft"/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__Co-Chair_of_Bill___Melinda_Gates_Foundation"/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__CEO_of_Cascade_Investment"/>
<ontology:occupation rdf:resource="&resource;Bill_Gates__Director_of_Berkshire_Hathaway"/>
<prop:name> Bill Gates </prop:name>
<prop:name>Gates, William Henry, III  </prop:name>
<owl:sameAs rdf:resource="http://www.microsoft.com/presspass/exec/billg/default.mspx"/>
<owl:sameAs rdf:resource="http://dbpedia.org/resource/Bill_Gates"/>
<owl:sameAs rdf:resource="http://rdf.freebase.com/ns/guid.9202a8c04000641f8000000000009e99"/>
<owl:sameAs rdf:resource="http://umbel.org/umbel/ne/wikipedia/Bill_Gates"/>
<owl:sameAs rdf:resource="http://sw.opencyc.org/concept/Mx4rvYwpvpwpEbGdrcN5Y29ycA"/>
```

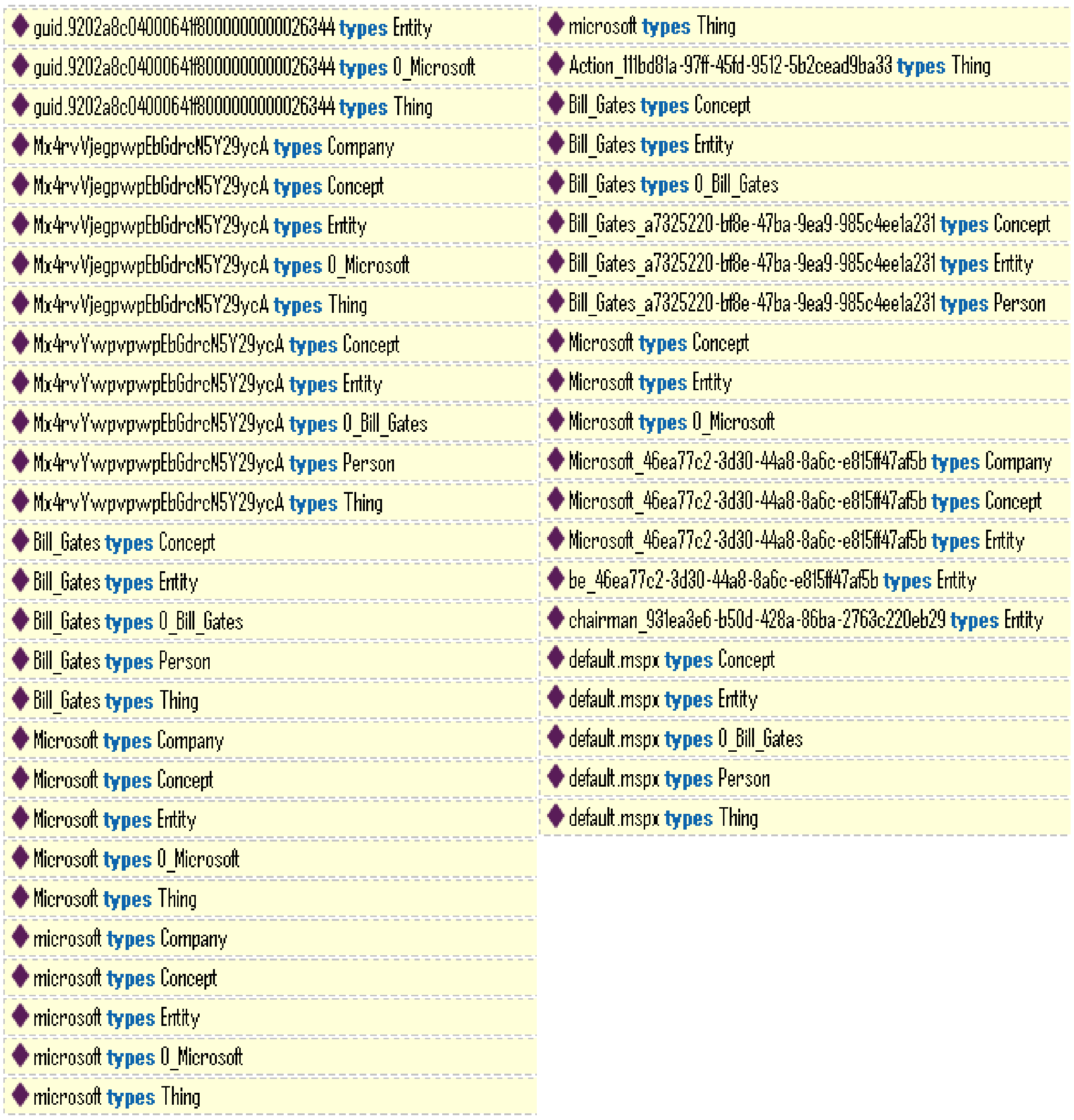
```
<owl:sameAs rdf:resource="http://mpii.de/yago/resource/Bill_Gates"/>
<owl:sameAs rdf:resource="&sample;Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231"/>
</owl:Thing>

<owl:Thing rdf:about="&sample;Microsoft">
<rdf:type rdf:resource="&sample;Company"/>
<ontology:foundationPerson rdf:resource="&resource;Bill_Gates"/>
<ontology:foundationPerson rdf:resource="&resource;Paul_Allen"/>
<ontology:industry rdf:resource="&resource;Computer_software"/>
<ontology:industry rdf:resource="&resource;Consumer_electronics"/>
<ontology:industry rdf:resource="&resource;Video_game_console"/>
<ontology:product rdf:resource="&resource;Microsoft_Game_Studios"/>
<ontology:product rdf:resource="&resource;Zune"/>
<ontology:product rdf:resource="&resource;Microsoft_Windows"/>
<ontology:product rdf:resource="&resource;Microsoft_Office"/>
<ontology:product rdf:resource="&resource;Microsoft_Servers"/>
<ontology:product rdf:resource="&resource;Microsoft_Visual_Studio"/>
<ontology:product rdf:resource="&resource;Microsoft_Expression_Studio"/>
<ontology:product rdf:resource="&resource;Microsoft_Dynamics"/>
<ontology:product rdf:resource="&resource;Windows_Live"/>
<ontology:product rdf:resource="&resource;Bing_(search_engine)"/>
<ontology:product rdf:resource="&resource;Windows_Phone"/>
<ontology:location rdf:resource="&resource;Redmond,_Washington"/>
<ontology:product rdf:resource="&resource;Windows_Phone"/>
<prop:slogan>Your potential. Our passion.</prop:slogan>
<owl:sameAs rdf:resource="http://dbpedia.org/resource/Microsoft"/>
<owl:sameAs rdf:resource="http://rdf.freebase.com/ns/guid.9202a8c04000641f8000000000026344"/>
<owl:sameAs rdf:resource="http://umbel.org/umbel/ne/wikipedia/Microsoft"/>
<owl:sameAs rdf:resource="http://sw.opencyc.org/concept/Mx4rvVjegpwpEbGdrcN5Y29ycA"/>
<owl:sameAs rdf:resource="http://mpii.de/yago/resource/Microsoft"/>
<owl:sameAs rdf:resource="http://www.crunchbase.com/company/microsoft"/>
<owl:sameAs rdf:resource="http://cb.semsol.org/company/microsoft.rdf"/>
</owl:Thing>
</rdf:RDF>
```

Annex D: Test Results 1 Entailments



microsoft.rdf types Company
microsoft.rdf types Concept
microsoft.rdf types Entity
microsoft.rdf types 0_Microsoft
microsoft.rdf types Thing
Berkshire_Hathaway types Thing
Bill_Gates types Concept
Bill_Gates types Entity
Bill_Gates types 0_Bill_Gates
Bill_Gates types Person
Bill_Gates types Thing
Bill_Gates__CEO_of_Cascade_Investment types Thing
Bill_Gates__Chairman_of_Microsoft types Thing
Bill_Gates__Co-Chair_of_Bill___Melinda_Gates_Foundation types Thing
Bill_Gates__Director_of_Berkshire_Hathaway types Thing
Bill___Melinda_Gates_Foundation types Thing
Bing_(search_engine) types Thing
Cascade_Investment types Thing
Computer_software types Thing
Consumer_electronics types Thing
Microsoft types Company
Microsoft types Concept
Microsoft types Entity
Microsoft types 0_Microsoft
Microsoft types Thing
Microsoft_Dynamics types Thing
Microsoft_Expression_Studio types Thing
Microsoft_Game_Studios types Thing
Microsoft_Office types Thing
Microsoft_Servers types Thing
Microsoft_Visual_Studio types Thing
Microsoft_Windows types Thing
Paul_Allen types Thing
Redmond,_Washington types Thing
Seattle,_Washington types Thing
Video_game_console types Thing
Windows_Live types Thing
Windows_Phone types Thing
Zune types Thing
Bill_Gates types Concept
Bill_Gates types Entity
Bill_Gates types 0_Bill_Gates
Bill_Gates types Person
Bill_Gates types Thing
Microsoft types Company
Microsoft types Concept
Microsoft types Entity
Microsoft types 0_Microsoft
Microsoft types Thing
guid.9202a8c04000641f80000000000009e99 types Concept
guid.9202a8c04000641f80000000000009e99 types Entity
guid.9202a8c04000641f80000000000009e99 types 0_Bill_Gates
guid.9202a8c04000641f80000000000009e99 types Person
guid.9202a8c04000641f80000000000009e99 types Thing
guid.9202a8c04000641f80000000000026344 types Company
guid.9202a8c04000641f80000000000026344 types Concept

Test Results 1 Entailments



- guid.9202a8c0400064ff8000000000026344 **types** Entity
- guid.9202a8c0400064ff8000000000026344 **types** 0_Microsoft
- guid.9202a8c0400064ff8000000000026344 **types** Thing
- Mx4rvVjegpwpEbGdrcN5Y29ycA **types** Company
- Mx4rvVjegpwpEbGdrcN5Y29ycA **types** Concept
- Mx4rvVjegpwpEbGdrcN5Y29ycA **types** Entity
- Mx4rvVjegpwpEbGdrcN5Y29ycA **types** 0_Microsoft
- Mx4rvVjegpwpEbGdrcN5Y29ycA **types** Thing
- Mx4rvYwpvpwpEbGdrcN5Y29ycA **types** Concept
- Mx4rvYwpvpwpEbGdrcN5Y29ycA **types** Entity
- Mx4rvYwpvpwpEbGdrcN5Y29ycA **types** 0_Bill_Gates
- Mx4rvYwpvpwpEbGdrcN5Y29ycA **types** Person
- Mx4rvYwpvpwpEbGdrcN5Y29ycA **types** Thing
- Bill_Gates **types** Concept
- Bill_Gates **types** Entity
- Bill_Gates **types** 0_Bill_Gates
- Bill_Gates **types** Person
- Bill_Gates **types** Thing
- Microsoft **types** Company
- Microsoft **types** Concept
- Microsoft **types** Entity
- Microsoft **types** 0_Microsoft
- Microsoft **types** Thing
- microsoft **types** Company
- microsoft **types** Concept
- microsoft **types** Entity
- microsoft **types** 0_Microsoft
- microsoft **types** Thing
- microsoft **types** Thing
- Action_111bd81a-97ff-45fd-9512-5b2cead9ba33 **types** Thing
- Bill_Gates **types** Concept
- Bill_Gates **types** Entity
- Bill_Gates **types** 0_Bill_Gates
- Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231 **types** Concept
- Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231 **types** Entity
- Bill_Gates_a7325220-bf8e-47ba-9ea9-985c4ee1a231 **types** Person
- Microsoft **types** Concept
- Microsoft **types** Entity
- Microsoft **types** 0_Microsoft
- Microsoft_46ea77c2-3d30-44a8-8a6c-e815ff47af5b **types** Company
- Microsoft_46ea77c2-3d30-44a8-8a6c-e815ff47af5b **types** Concept
- Microsoft_46ea77c2-3d30-44a8-8a6c-e815ff47af5b **types** Entity
- be_46ea77c2-3d30-44a8-8a6c-e815ff47af5b **types** Entity
- chairman_931ea3e6-b50d-428a-86ba-2763c220eb29 **types** Entity
- default.mspx **types** Concept
- default.mspx **types** Entity
- default.mspx **types** 0_Bill_Gates
- default.mspx **types** Person
- default.mspx **types** Thing