POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Dottorato in Ingegneria Informatica e dei Sistemi – XXIV ciclo

Tesi di Dottorato

Knowledge extraction from unstructured data and classification through distributed ontologies

Giuseppe Rizzo

Tutore
prof. Angelo Raffaele Meo

Coordinatore del corso di dottorato
prof. Pietro Laface

March 2012

Summary

The World Wide Web has changed the way humans use and share any kind of information.

The Web removed several access barriers to the information published and has became an enormous space where users can easily navigate through heterogeneous resources (such as linked documents) and can easily edit, modify, or produce them. Documents implicitly enclose information and relationships among them which become only accessible to human beings. Indeed, the Web of documents evolved towards a space of data silos, linked each other only through untyped references (such as hypertext references) where only humans were able to understand. A growing desire to programmatically access to pieces of data implicitly enclosed in documents has characterized the last efforts of the Web research community. Direct access means structured data, thus enabling computing machinery to easily exploit the linking of different data sources. It has became crucial for the Web community to provide a technology stack for easing data integration at large scale, first structuring the data using standard ontologies and afterwards linking them to external data. Ontologies became the best practices to define axioms and relationships among classes and the Resource Description Framework (RDF) became the basic data model chosen to represent the ontology instances (i.e. an instance is a value of an axiom, class or attribute). Data becomes the new oil, in particular, extracting information from semi-structured textual documents on the Web is key to realize the Linked Data vision. In the literature these problems have been addressed with several proposals and standards, that mainly focus on technologies to access the data and on formats to represent the semantics of the data and their relationships.

With the increasing of the volume of interconnected and serialized RDF data, RDF repositories may suffer from data overloading and may become a single point of failure for the overall Linked Data vision. One of the goals of this dissertation is to propose a thorough approach to manage the large scale RDF repositories, and to distribute them in a redundant and reliable peer-to-peer RDF architecture. The architecture consists of a logic to distribute and mine the knowledge and of a set of physical peer nodes organized in a ring topology based on a Distributed Hash Table (DHT). Each node shares the same logic and provides an entry point that

enables clients to query the knowledge base using atomic, disjunctive and conjunctive SPARQL queries. The consistency of the results is increased using data redundancy algorithm that replicates each RDF triple in multiple nodes so that, in the case of peer failure, other peers can retrieve the data needed to resolve the queries. Additionally, a distributed load balancing algorithm is used to maintain a uniform distribution of the data among the participating peers by dynamically changing the key space assigned to each node in the DHT.

Recently, the process of data structuring has gained more and more attention when applied to the large volume of text information spread on the Web, such as legacy data, news papers, scientific papers or (micro-)blog posts. This process mainly consists in three steps: *i)* the extraction from the text of atomic pieces of information, called named entities; *ii)* the classification of these pieces of information through ontologies; *iii)* the disambigation of them through Uniform Resource Identifiers (URIs) identifying real world objects. As a step towards interconnecting the web to real world objects via named entities, different techniques have been proposed. The second objective of this work is to propose a comparison of these approaches in order to highlight strengths and weaknesses in different scenarios such as scientific and news papers, or user generated contents. We created the Named Entity Recognition and Disambiguation (NERD) web framework, publicly accessible on the Web (through REST API and web User Interface), which unifies several named entity extraction technologies. Moreover, we proposed the NERD ontology, a reference ontology for comparing the results of these technologies. Recently, the NERD ontology has been included in the NIF (Natural language processing Interchange Format) specification, part of the Creating Knowledge out of Interlinked Data (LOD2) project.

Summarizing, this dissertation defines a framework for the extraction of knowledge from unstructured data and its classification via distributed ontologies. A detailed study of the Semantic Web and knowledge extraction fields is proposed to define the issues taken under investigation in this work. Then, it proposes an architecture to tackle the single point of failure issue introduced by the RDF repositories spread within the Web. Although the use of ontologies enables a Web where data is structured and comprehensible by computing machinery, human users may take advantage of it especially for the annotation task. Hence, this work describes an annotation tool for web editing, audio and video annotation in a web front end User Interface powered on the top of a distributed ontology. Furthermore, this dissertation details a thorough comparison of the state of the art of named entity technologies. The NERD framework is presented as technology to encompass existing solutions in the named entity extraction field and the NERD ontology is presented as reference ontology in the field. Finally, this work highlights three use cases with the purpose to reduce the amount of data silos spread within the Web: a Linked Data approach to augment the automatic classification task in a Systematic Literature Review, an

application to lift educational data stored in Sharable Content Object Reference Model (SCORM) data silos to the Web of data and a scientific conference venue enhancer plug on the top of several data live collectors.

Significant research efforts have been devoted to combine the efficiency of a reliable data structure and the importance of data extraction techniques. This dissertation opens different research doors which mainly join two different research communities: the Semantic Web and the Natural Language Processing community. The Web provides a considerable amount of data where NLP techniques may shed the light within it. The use of the URI as a unique identifier may provide one milestone for the materialization of entities lifted from a raw text to real world objects.

Acknowledgements

Since I started the PhD I was fully absorbed by the need to learn. The ignorance has been the key for better understanding. What is presented in this work has been done thanks to all the people who shared with me this experience.

Thanks to Angelo Raffaele Meo and Antonio Servetti whom I am proud to call my supervisors. They guided me during these three years. Thanks to prof. Raffaele, for encouraging me several times. He patiently gave me right suggestion at right time. Definitely, he inspired for his always think positively. Thanks to Antonio, an always willing person whom gave me crucial feedbacks for all my works.

A special thanks to Federico Di Gregorio and Pierluigi Di Nunzio, whom I had the honor to work with. They introduced me in the Semantic Web community, giving crucial baselines for my studies. They patiently explained their findings about the Web and reserved to me time out of their job.

When I joined his research group, from the beginning I felt the incredible energy he spends for the research. His love to do this job is my model of work. Thanks to Raphaël Troncy, he constantly guided through every step of my last year of PhD.

Thanks to the following mates who shared with me the daily moments of my adventure. They gave me right assistance and right collaboration at the right time. At Politecnico: Biagio, Matteo, Antonio, Luca, Federico, Andrea. At EURECOM: Carmelo, Houda, and Ghislain.

All my works were born thanks to fruitful discussions and crucial reviews: thanks to all my co-authors and anonymous reviewers. And special thanks to all my students (at Politecnico and at EURECOM) for the discussions and the ideas we shared. What I learned during this years is also thanks to you.

I could not have asked for better friends than the following. Agnese, Alessandro, Chiara, Cristina, Enzo, Dario, Marco, Umberto, Vincenzo, the “Salentopoli” folks with whom I enjoyed the time out of research. I shared with them important life moments which always they make me happy just reminding them. Thanks all.

Everything I did is mainly because you supported me everyday. Your suggestions and think positively, even during my dark moments, shed the light to this experience. Thank you mom, dad, sisters Modesta, and Annalisa, brothers in love Uccio, and Giuseppe and my lovely niece Cristiana.

Finally, a lovely thanks to Eleonora. She has been the always present person who assisted me every second of this adventure even when my carrier was going far from her. Part of this work belongs to her. Thank you honey.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

A large amount of data is spread across the Web and an increasingly large number of users is making advanced use of it. This data corresponds to raw information, which can be used to enrich existing data linking them to existing resources, and then creating new knowledge. Historically, the Web has been a platform where documents have been collected and linked together through hyperlinks, becoming an enormous storage of unstructured data. However, the information hosts in documents is not easily processable by machines. Thus, the request to have fragmented documents increased. Pieces of information are easier to manage, especially if they are described with labels which belong to different types and are defined using common vocabularies or ontologies. This is the idea behind the trend called Web of Data. The specialization introduced by the Web of Data enables to structure resources in different items according to well known schemes, so that they can be read separately and if necessary combined with other pieces of data spread on the Web. If the datasets are interconnected, a user can freely navigate through the links creating new data views. Bizer *et al.* [13] coined the term Linked Data to define this emerging trend within the Semantic Web community. In the rest of this chapter we discuss the need to have data on the Web, that is freely accessible and published according to defined schemes that could make easy the process of using and linking them.

1.1 Data Publishing

As a step towards linking data across the Web, a crucial role to publish information is characterized by the Berners-Lee's principles [8]. These principles are the following:

1. use URIs as names for things;
2. use HTTP URIs, so that people can look up those names;

3. when someone looks up a URI, provide useful information, using the standards (RDF, SPARQL);

4. include links to other URIs, so that they can discover more things.

The first principle proposes the Uniform Resource Identifier (URI) [9] as the reference to point to Web documents, digital contents, also real world objects such as people, locations, relations and abstract concepts such as relationship, set of things. The second defines the HyperText Transfer Protocol (HTTP) [31] as the universal mechanism to access every kind of information spread on the Web. So, the combination of HTTP and URI allows users to identify on web resources by means of URIs and these resources can be looked up through the HTTP protocol. One of the key features of the Web of documents was the adoption of the HyperText Markup Language (HTML) document format. With the advent of the Web of data, it is still important to identify a common data model; indeed the third principle defines the Resource Description Framework (RDF) as the data model and suggests the SPARQL Protocol and RDF Query Language (SPARQL) as the language to query information from data repositories. The fourth principle invites data owners to create links to external resources (documents, real world objects or abstract concepts) which adhere to the above principles. These links become also a typed reference among resources.

On one hand the adoption of the guidelines creates a unified landscape of datasets linked each others, where structured information may be easily exchangeable. On the other hand they create a discontinuity from the past, when documents were published in data silos without any concerns about fragmentation, data representation and data licence. So, the critical aspect of these guidelines is the backward compatibility with previous technologies developed only for storing local information or for publishing it as "it is" without links to external resources. This is the scenario that several private companies and public governments are facing. Indeed, they have a big wealth of data, mostly stored in relational tables, tabular files such as Comma Separated Value (CSV), eXtensible Markup Language (XML), Calculator (CALC) or plain text. In the rest of this section we investigate mainly on how lifting existing structured data to the Linked Open Data Cloud (LOD). Subsequently, we describe the approach to extract Linked Data information from unstructured data.

1.1.1 Structured data

Structured data means data represented according to defined schema, such as tables, taxonomies, dictionaries or ontologies. Generally, structured data is easily interpreted by computing machinery and represents an important step towards data interoperability. To adhere to the Berners-Lee's principles, before publishing, although structured, this data has to be converted. Figure 1.1 details the conversion

process according to the source that is used as content provider: if the data comes from relational database, it is converted by means of RDBtoRDF systems such as Direct Mapping[1] or R2RML[2], instead if data comes from spreadsheets or similar it is converted by means of RDFizers[3]. These techniques may do the conversion process on the fly, mapping relational tables to RDF tuple, or may do the job as batch saving the information in a RDF archive. In both cases, the output of the conversion process creates a resource that is potentially ready to be consumed as Linked Data.

1.1.2 Unstructured data

Since its origins, the Web has accumulated several resource repositories, where information is mainly trapped in textual documents. By its nature, a text is rich in semantic that is clear and manageable to the reader but not to computing machinery. It becomes manageable only when the semantic is structured in tokens. Usually, a human being can extract this information and can represent it according to the pattern key and value. After this human extraction process, a text can be easily managed by machines. But human efforts are time consuming, so that many research communities have tried to tackle this problem using several automatic techniques. Leveraging from the Natural Language Processing (NLP) community findings performed in several years, the Linked Data community has exploited different techniques to tokenize the text in parts of speech (POS) describing the use of each token (e.g. identifying verb, noun, adjective, conjunction) and to extract semantic information unit, also called Named Entity (NE), from textual documents. Focusing mainly on the NE, first we introduce the definition proposed by Grishman [38]. A named entity is an information unit described by the name of a person or an organization, a location, a brand, a product, a numeric expression including time, date, money and percent found in a sentence. In addition to the named entity recognition challenge, the Linked Data community has proposed the classification of NEs through fine grained ontologies, well known by the Semantic Web community, such as DBpedia ontology [4], Freebase[4], and YAGO [90] and the disambiguation through URIs, possibly coming from the LOD, which describe real word objects.

Figure 1.2 details the workflow to lift the information trapped in a document to the LOD. According to the technology used to store this information (text, PDF, HTML, ODT), it is elaborated through NLP techniques and then NEs are extracted.

[1] http://www.w3.org/TR/rdb-direct-mapping

[2] http://www.w3.org/TR/r2rml

[3] http://openstructs.org/resources/rdfizers

[4] http://www.freebase.com

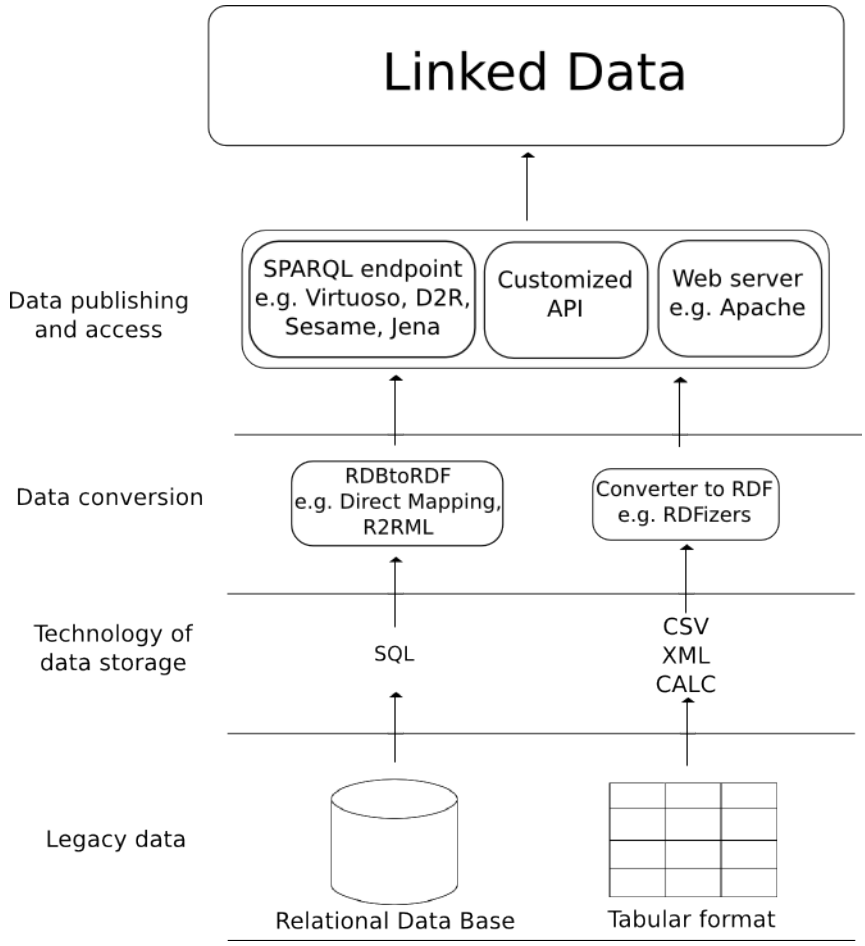

Figure 1.1. The workflow for lifting structured data, such as data stored in relational database of tabular files, to the LOD. The final goal of this approach is to publish web resources according to the Berners-Lee's rules.

Usually this information is disambiguated by means of URIs which describe real word objects and classified through ontologies. Depending on the technique used to store the information extracted, the list of named entities is published on the LOD by means of SPARQL endpoint, customized APIs, or through a web server.

1.2 Linking Data

The linking among data is one of the biggest advantages of the Linked Data vision. Data when isolated has small value, differently it gains value when it is linked to other data, which are produced and published independently by different individuals. The aggregation process is generated by applications able to create views on

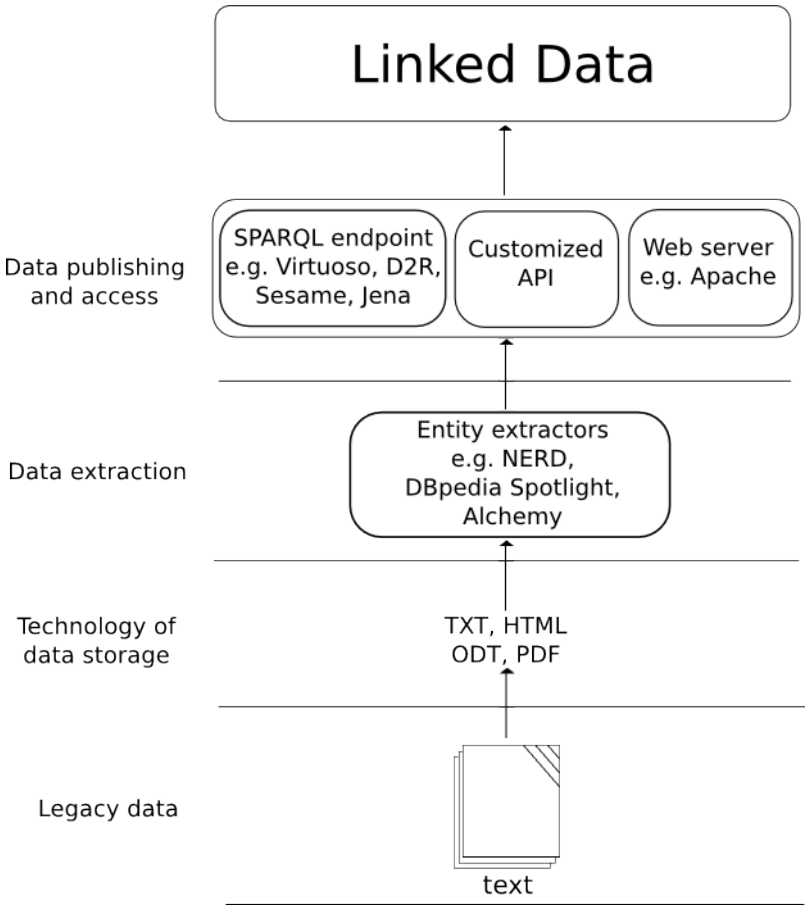

Figure 1.2. The workflow for lifting unstructured data, such as text to the LOD. The final goal of this approach is to publish the extracted information as web resources according to the Berners-Lee's rules.

the available raw data (in this approach the data layout is demanded to the final user of the data). The idea follows the Separation of Concerns (SoC) principle. The difference between data and presentation allows to navigate through resources, making customizable views which better respond to the need of who provides contents (author) and who uses, reuses and redistributes them. Data is presented without any information about the layout, so it is raw, but it still rich of semantics and ready to be used by computing machinery to easily extract information or, even, to aggregate it to other data instances. In this context computing machinery become the first class citizen of the Web, potentially able to jump from different data silos for inferring new knowledge. Data and relationships are serialized according to the RDF model. One of the most important improvement introduced by this model is the possibility to represent the type of the connection between data. The RDF model allows to define the nature, hence the type of the link, overcoming the

untyped connection existing in the Web of documents. Therefore, a Web in which data is both published and linked using RDF is a web where data is significantly more discoverable, and therefore more usable.

1.3 A cloud of Linked Open Data

When a dataset is published according to the Linked Data principles, it is ready to be part of the LOD. The LOD represents a fast growing space of information where an increasing number of individuals and organizations is contributing to maintain it by choosing to share their data with others. Numerous private institutions published their datasets, from which we can list Yahoo![5], Amazon[6] but also newspapers such as BBC[7], The New York Times[8] and research initiatives within various scientific disciplines such as DBpedia[9], Yago[10] and scientific journals such as IEEE[11], ACM[12]. As shown in Figure 1.3, the number of datasets is considerable important and Cyganiak *et al.* [28] estimated this number equal to 295. In this direction, several efforts have been spent by the public sectors of the U.S. government and U.K. government. They created public portals where data are published as public domain. Allowing users to freely extract knowledge from them, linking them and re-publishing them. The U.S. government published the portal `http://www.data.gov`, while the U.K. government `http://data.gov.uk`. Several other public institutions followed this direction, performing an incredible effort for providing new data that could be potentially linked to the LOD.

We are surrounding of data and many actors are playing a crucial role in it for publishing it. But, whenever this data is published, has to be well defined the licence in order to allow a user to consume in such a way this data. As Rizzo *et al.* highlighted in their work [81], open licences help to accomplish the fourth Linked Data principle, because the link to another data may exist only if it is discoverable and usable.

[5] `http://www.yahoo.com`

[6] `http://www.amazon.com`

[7] `http://bbc.com`

[8] `http://www.nytimes.com`

[9] `http://dbpedia.org`

[10] `http://www.mpi-inf.mpg.de/yago-naga/yago`

[11] `http://www.ieee.org`

[12] `http://www.acm.org`

1.4 Objective of this work

In the heterogeneous landscape of the Web, data becomes the new oil. Objective of this work is to tackle the problem of data overloading for large RDF repositories and to extract knowledge from legacy data and, potentially, to make it ready for publishing in the LOD. To summarize, this dissertation answers two research questions:

Research question 1: with the increasing of the volume of interconnected data described in RDF, large RDF repositories suffer from data overloading and become single point of failure. How to make sure that large RDF datasets are not single point of failure? How to distribute efficiently the RDF instances to enable a load balancing strategy? For instance, DBpedia is one of the largest RDF repository in the LOD, it has many duplicates across organizations since the official endpoint http://dbpedia.org/sparql is not reliable and it offers often single point of failure.

Research question 2: legacy data represent a source of knowledge, that is easily accessible to human beings. Often these data are published according to the Web of documents model and the conversion to the Web of Data model requires the use of automatic techniques such as named entity extraction. If the Web of Data will really reach a new scale using named entity extractors, how can we compare the performances of these extractors in order to highlight strengths and weaknesses of them? How can we leverage and combine the progress made by the NLP community?

1.5 Results

We propose a novel RDF architecture which is based on the assumption to be redundant and reliable even in case of point of failures. To obtain this result we implemented a distributed architecture composed of peer nodes organized in a ring topology based on a Distributed Hash Table (DHT) where each node provides an entry point that enables clients to query the knowledge base using atomic, disjunctive and conjunctive SPARQL queries. The consistency of the results is increased using data redundancy algorithm that replicates each RDF triple in multiple nodes so that, in the case of peer failure, other peers can retrieve the data needed to resolve the queries. Additionally, a distributed load balancing algorithm is used to maintain a uniform distribution of the data among the participating peers by dynamically changing the key space assigned to each node in the DHT. Experimental results show the ability of this architecture to tackle the problem of network load

balancing and redundancy. Moreover, we propose two use cases powered on this assumption to drive human beings to categorize media resources, exploiting the media fragment.

Then an overall comparison of 10 named entity extractors in pre-defined task, scenario and settings is detailed. We conducted mainly two types of experiments: qualitative and quantitative ones. For the former, we collected human ratings on the evaluation of precision for the extraction task per two different datasets and we compared them with the state of the art in the field. Then we conducted a quantitative experiment, running several experiments to assess the performances of these extractors in different scenario such as news articles, scientific paper and user generated contents. We created the NERD ontology a set of mappings established manually among the schemes provided by the extractors, we published the NERD web framework used in these experiments at http://nerd.eurecom.fr, built following the REST principles.

The extraction of data from unstructured sources of information and the aggregation of them extend the available information about textual resources for a computing machinery point of view. In this context, we propose approaches which rely on real educational data such as scientific papers, e-learning data, and scientific conference data (i.e. media contents, conference venue information and micro-blog posts).

1.6 Thesis structure

Before going into the details of this dissertation, we summarize its structure to give an overview of the research proposed.

Chapter 2, *Background*, overviews Semantic Web principles and knowledge extraction fundamentals. The main purpose is to summarize the current challenges about distributed ontologies and named entity extraction techniques. Furthermore, it highlights fields where the research needs to converge in order to go beyond the current state of the art.

Chapter 3, *Distributed ontologies for semantic annotation*, shows a proposal to distribute ontology instances in a P2P network, and to reduce the single point of failure due to the centralized approach. The amount of data is substantially increasing and repositories where this information is stored provide bottlenecks, becoming single point of failure for the entire Web architecture. An approach to tackle this problem is proposed in this chapter. Moreover, it details an annotation tool powered on the top of it which exploits the media fragment and drives human beings during the media categorization process.

Chapter 4, *Knowledge extraction from unstructured data*, highlights the importance to extract automatically structured information from unstructured data silos through NLP techniques. Then it shows an overall approach which focuses on weaknesses and strengths of 10 extractors in different test scenario depending by the content sources. It proposes the NERD framework composed of a REST API, web User Interface, and the NERD ontology which has became a reference ontology in the context of the NIF project.

Chapter 5, *Use cases of Linked Data applications for managing educational data*, leveraging on NLP techniques, three different proposals are advanced, respectively, mining structured data from scientific papers, lifting educational data from data silos to the Web of Data, and extracting Linked Data entities from generated contents related to a conference venue.

Chapter 6, *Conclusions* shows the motivations about our research and future targets in the field of knowledge extraction and distributed ontology for the Linked Data community.

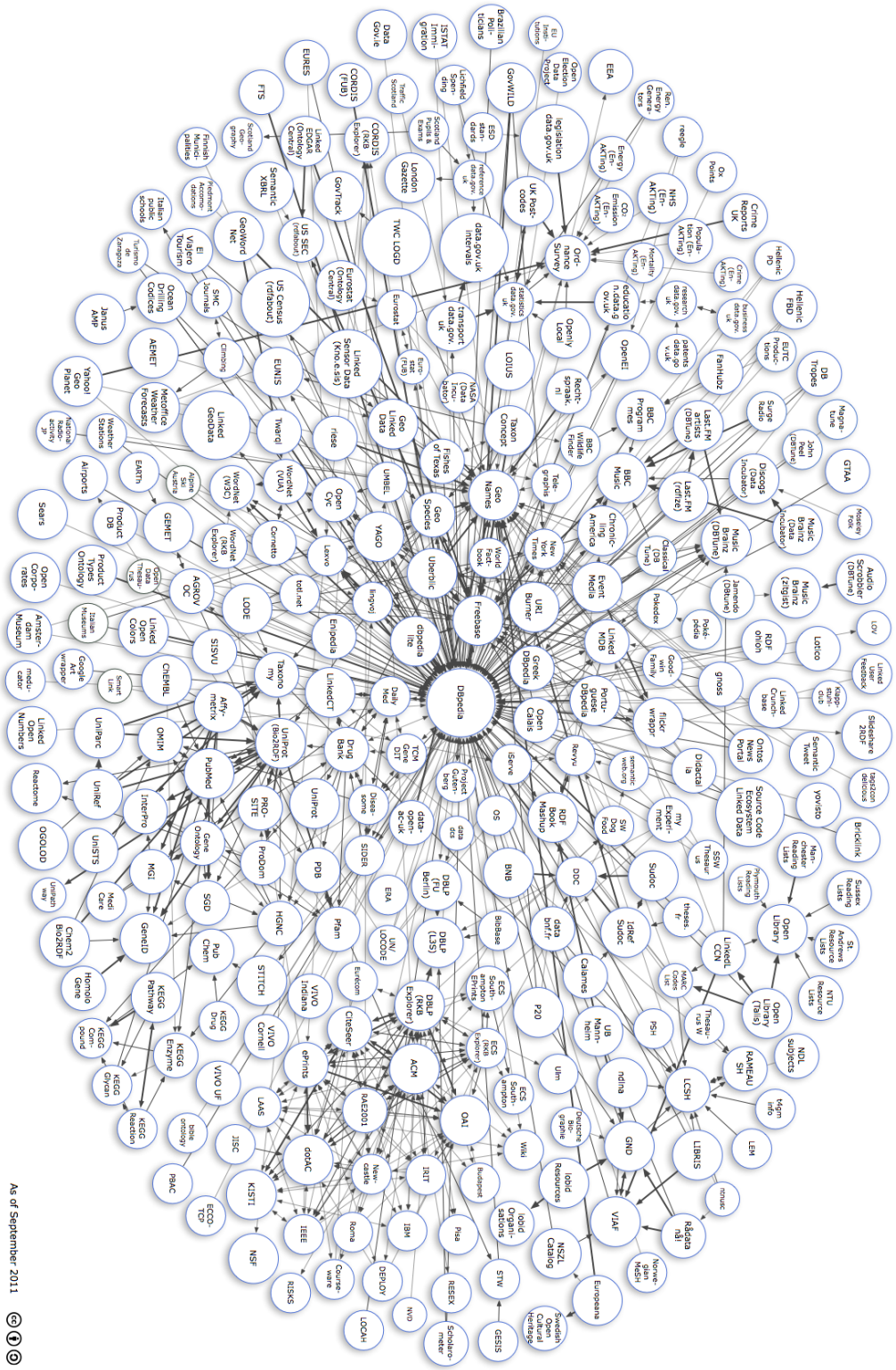

Figure 1.3. The Linked Open Data Cloud (LOD) contains 295 datasets of different purposes.

Chapter 2

Background

Key for the Semantic Web community is the possibility to create new knowledge by sharing structured information originated by multiple sources using a common framework based on standard data formats and network protocols. In such a scenario the information is frequently distributed on a very large number of nodes. Beside, a considerable amount of nodes publish not structured information on the Web. It becomes crucial to manage the large amount of heterogeneous data and to automatically extract useful information.

Although the main objective of the Web has remained the same during the years (to be a public archive of data), its application has been changed different times. An important change happened with the introduction of the Web of Data. Before this transition, the Web was an enormous space where textual documents were collected and published. That information was mainly human readable and the human beings were first class citizen of this space. After the change, machines became a first class citizen, but the amount of data human readable already published on the Web was enormous (just considering legacy data of public institutions, etc.). Hence, in such a context that the Semantic Web community joints the Natural Language Processing community in order to automatically extract structured information and to publish it according to the Linked Data principles.

In the rest of this chapter, we detail the fundamentals about the Semantic Web, first describing the RDF data model and the RDF Schema (section 2.1), and the ontology modelling (section 2.2). RDF repositories have been largely adopted, hence it has been crucial the role of distributed solutions, able to balance the knowledge base in order to avoid single point of failure (section 2.3). The nature of the information stored in an ontology is structured, e.g. described using defined vocabulary. But, the Web is an enormous space where heterogeneous information is stored. A crucial challenge is represented by the knowledge extraction task (section 2.4). A joint effort from the Semantic Web community and the Natural Language Processing community is presented by the findings about the disambiguation process of named

entities through web resource, as detailed in section 2.5.

2.1 Resource Description Framework

The Resource Description Framework (RDF) [55] is a language for describing Web resources, such as individuals, real world objects, properties and value of such properties. For instance, we consider the person *Giacomo Leopardi* who is disambiguated by an URI in the web space. Following the DBpedia [14] ontology instance representation[1], we use the URI `http://dbpedia.org/resource/Giacomo_Leopardi` which identifies univocally the resource of the individual *Giacomo Leopardi* within the Web. Following the semantic humans use to define real world objects, we can state Giacomo Leopardi is a Person. Therefore we define the type of a real world object Giacomo Leopardi assigning the class Person to the URI which describes univocally the Giacomo Leopardi instance. Finally, we can list some personal details about *Giacomo Leopardi*, for instance we detail the birthday and the full name. Figure 2.1 provides an example schema of the relations underlined above.

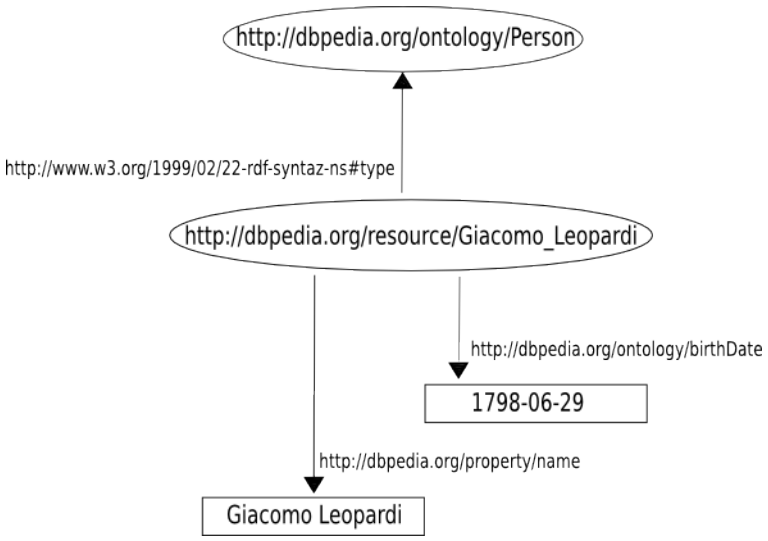

Figure 2.1. A graph representation of the information about Giacomo Leopardi.

The RDF representation is serialized according to the eXtensible Markup Language (XML) syntax [18]. Below, we present the serialization of the example shown in Figure 2.1 using the RDF/XML serialization.

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

[1]The URIs used in these examples are actually deferencable within the Web space.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:dbpprop="http://dbpedia.org/property/"
         xmlns:dbpedia-owl="http://dbpedia.org/ontology/">
 <dbpedia-owl:Person
         rdf:about="http://dbpedia.org/resource/Giacomo_Leopardi">
  <dbpedia-owl:birthDate>1798-06-29</dbpedia-owl:birthDate>
  <dbpprop:name>Giacomo  Leopardi</dbpprop:name>
 </dbpedia-owl:Person>
</rdf:RDF>
```

The RDF language is machine processable and it links web resources across the Web, by means of URIs. It differs from HTML mainly for two aspects: first RDF URIs can point to any identifiable thing, including abstract concepts or things not identifiable within the Web, such as a person (Giacomo Leopardi in the previous example). The second difference is RDF properties have URIs, to precisely identify the relationships that exist between the linked items. Since the reference was used in the Web of documents (i.e. HTML documents), all the relationships were untyped, through the RDF language they assume a type.

2.1.1 RDF data model

The RDF data model describes information as directed graph, where arcs and nodes are labelled. The RDF data model is extensible and flexible, through it can represent heterogeneous information, using different schemes [6]. The RDF data model is described in detail as part of the W3C RDF Primer [61].

In RDF, the description of a resource is represented as a triple, i.e. a set of three elements. A triple contains a *subject*, a *predicate*, and an *object*, also declared as *(s,p,o)*. A triple definition follows the natural language human beings use to describe actions and events. We detail an example to describe the analogy between the two forms of communications. Let's define the statement: *Giacomo Leopardi wrote Il sabato del villaggio*, *Giacomo Leopardi* is the subject, while *wrote* is the predicate, and *Il sabato del villaggio* is the object. The subject of a triple is the URI identifying the described resource. The predicate indicates what kind of relationship exists between subject and object (in the example it defines an action) and it is defined according to a vocabulary or taxonomy. Finally, the object can either be a value such as string, integer, date or an URI. The former is called literal, the latter link. According to such a finding, the triple becomes a Literal triple or an RDF link. Figure 2.2 shows the graph representation of this statement.
We focus on the differences that exist between the two types of statements:

Literal triple has a string, number, or data as the object. It is used to describe the property of objects. In the example above, the literal describes the name

Figure 2.2. A graph representation of the statement: *Giacomo Leopardi wrote Il sabato del villaggio.*

of the poetry Leopardi wrote. A literal can be plain or typed: a plain literal is a string combined with an optional language tag (such as English, Italian). A typed literal is a string combined with a datatype URI. The datatype URI identifies the datatype of the literal. Datatype URIs for common datatypes such as integers, floating point numbers and dates are defined by the XML Schema datatypes specification [12]. The common data type used is *string*.

RDF link describes the relationship between two resources. The triple is composed of three URIs, each of them points to an existing web resource. We distinguish from internal and external link. With the former you can point resource which is within the local Linked Data source, while the latter is a URI that describes an external resource.

As mentioned, an RDF triple can be represented as a labelled graph composed of URIs. Taking into account that URIs are globally identifier within the LOD, we can consider a set of triples as giant graph, as highlighted by Tim Berners-Lee in [7].

2.1.2 RDF Schema

The RDF provides a data model to represent relationships among resources, using typed properties and values. However, as Figure 2.1 introduces, the RDF data model does not describe the nature of the relationships, but it only details them. Indeed, in the same example we used a class *dbpedia-owl:Person* and *dbpedia-owl:birthDate*, and used properties such as *dbpprop:name* to describe the relationships. RDF itself provides no means for defining such domain specific classes and properties. Instead, these classes and properties are described as an RDF vocabulary, using extensions to RDF provided by the RDF Schema [19]. However, the RDF Schema does not provides the facilities needed to model the conceptualization of the relationships of a domain context, but instead provides the facilities needed to describe such classes and properties, and to indicate which classes and properties are expected to be used together. To summarize, the RDF Schema provides a type system for RDF.

The RDF Schema facilities are themselves provided in the form of an RDF vocabulary; that is, as a specialized set of predefined RDF resources with their own special meanings. The resources in the RDF Schema vocabulary have URIs with the prefix `http://www.w3.org/2000/01/rdf-schema#` (conventionally associated with the prefix *rdfs:*). Vocabulary descriptions (schemes) written in the RDF Schema

are compliant RDF graphs. Hence, this allows backward compatibility with all the software that are not aware of the RDF Schema, but they can still interpret it as a legal RDF graph consisting of various resources and properties, but they do not "understand" the additional built-in meanings of the RDF Schema terms. To understand these additional meanings, RDF software have to be written to process an extended language that includes not only the *rdf:* vocabulary, but also the *rdfs:* vocabulary, together with their built-in meanings.

A basic step in any kind of description process is identifying the different types of things to describe. The RDF Schema refers to these "kinds of things" as classes. A class in RDF Schema corresponds to a generic concept of a Thing, something like the notion of a class in object-oriented programming languages. RDF classes can be used to represent almost any category of thing, such as Web pages, people, document types, databases or abstract concepts. Classes are described using the RDF Schema resources *rdfs:Class* and *rdfs:Resource*, and the properties *rdf:type* and *rdfs:subClassOf*. Let's consider the example shown in Figure 2.1. The URI `http://dbpedia.org/resource/Giacomo_Leopardi` points to an individual and it refers to a person, actually *Giacomo Leopardi*. This individual belongs to a class Person, so we can define formally the type of this URI as:

```
dbpedia-owl:Person   rdf:type      rdfs:Class  .
```

In the example, the property *rdf:type* is used to indicate that a resource is an instance of a class. For the sake of brevity, we used the rdfs schema instead of owl (as it happens for the real definition). In such a way, we extend the meaning of the class Person, making it compliant with the OWL Full (see section 2.2). This serialization shows how the class Person is linked to the vocabulary it uses: *dbpedia-owl* (further explanations are in Section 2.2). Now, we suppose that the Person class is a class inherited by the Thing class: we model this relationship through the *rdfs:subClassOf* property.

```
dbpedia-owl:Person   rdf:type          rdfs:Class  ;
                     rdfs:subClassOf   owl:Thing  .
```

The *rdf:Property* is used when we define the property of an attribute. Following the same example, we have:

```
dbpedia-owl:Person   rdf:type          rdfs:Class  ;
                     rdfs:subClassOf   owl:Thing  .
dbpprop:name         rdf:type          rdf:Property  .
```

To define the relationship between a class and a property, we use the *rdfs:domain*, while the *rdfs:range* defines the value type which the property has; the previous example becomes:

```
dbpedia-owl:Person   rdf:type          rdfs:Class  ;
                     rdfs:subClassOf   contact:Thing  .
dbpprop:name         rdf:type          rdf:Property  .
```

```
dbpedia-owl:birthDate  rdf:type          owl:DatatypeProperty ;
                       rdf:type          owl:FunctionalProperty ;
                       rdfs:domain       dbpedia-owl:Person ;
                       rdfs:range        xsd:date .
```

The RDF Schema type system is similar in some aspects to the type systems of object-oriented programming languages. However, the RDF differs from most programming language type systems in several important aspects. One important difference is that instead of describing a class as having a collection of specific properties, an RDF Schema describes properties as applying to specific classes of resources, using domain and range properties. This enables to have a property that may be used by other classes and the modification on that is propagated on all the other classes. So, property descriptions are, by default, independent by the class definition.

2.2 Ontology

Both RDF and RDF Schema allow to define relationships among resources and to describe them, but they do not provide any methodology to define the formal explicit description of classes, properties, attributes, and restrictions on attributes. As we introduced in section 2.1.2, a vocabulary provides entry definitions of relationships among classes. A vocabulary becomes a set of classes, relationships and restrictions. The Artificial Intelligence (AI) community introduced the concept of ontology, as "an explicit formal specifications of the terms in the domain and relations among them" [41]. Then, the Web community adopted it for extending RDF Schema with more expressive constructs aimed at facilitating agent interaction on the Web, i.e. moving "de facto" towards a machine-readable Web, including definitions of basic concepts in a domain and relations among them. Several attempts were spent to formalize a language to accomplish the use of ontologies to define a common vocabulary for researchers who need to share information in a domain. We can list DARPA Agent Markup Language (DAML) [43], which was the first proposal of ontology modelling in the Web community and the Web Ontology Language (OWL) [40] adopted by the World Wide Web Consortium (W3C)[2] as the current standard language. The OWL represents a natural evolution of DAML and it extends XML, XML Schema, RDF and RDF Schema with additional representational constructs and restrictions. Therefore, the OWL allows to represent that two classes are disjoint, that number of values of certain properties are limited, classes are equal, defines much more properties for the attributes, define the symmetry of properties and enumerated classes. The OWL comes in three variations:

[2] http://www.w3c.org

OWL Lite : supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies.

OWL DL : supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and reliability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logic, a field of research that has studied the logics that form the formal foundation of OWL.

OWL Full : is meant for users who want maximum expressiveness and the syntactic freedom of the RDF Schema with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

2.3 Distributed Ontology

An ontology is a hierarchical structured set of terms for describing a domain that can be used as a skeletal for describing a domain. Often, the term ontology defines also the data it hosts, i.e. instances. Using an analogy with the database literature, we can define an instance is the equivalent of a record in a database table. A record, generally speaking, follows the structure of the table where it is stored. A bag of records, belonging to different tables, define the content of the database, or better, the knowledge base. Going back to the Semantic Web literature a set of instances, which follow the schema provided by an ontology, create the knowledge base. Although the ontology definition defines only the relationship level among classes, often that word is used to define the materialization of the classes, i.e. the instances. In this section we refer as ontology to the stack which contains both the structure definition and the knowledge base.

Many efforts have been made to build an infrastructure composed by several centralized peers aimed at sharing knowledge and support consistency and availability of retrieved data. Jena [23] and Sesame [20] are examples of centralized storage systems which enable external applications to retrieve data using the SPARQL [2],

a graph-matching query language used to retrieve semantic knowledge from RDF repositories. A set of this storage servers can cooperate together to share knowledge, but the major issue related to this approach is that the results may change over time, as a function on the availability of network nodes. That is, if a node becomes unavailable it is not backed up by any other node of the network. Thus any node may be a single point of failure for the entire architecture [87].

To overcome these centralized constraints, many approaches in literature consider a distributed scenario, based on a P2P network. A P2P approach is proposed in Edutella [70] using the Gnutella protocol [54]. The overlay is composed of a set of Super Peers (SP) [71], each of them connected to a large number of simple peers. The SPs are nodes with high network bandwidth and sustainable computation power. These SPs form the backbone, manage a local set of RDF data and are responsible for routing and querying. Although this approach is developed for a distributed scenario, it also presents the single point of failure issue for the SPs, because when a SP is not available, the consistency of the retrieved knowledge is compromised.

The P2P paradigm offers an interesting alternative to existing information system infrastructures. The most important features are: *i)* scalability in terms of number of nodes and distribution, *ii)* direct access to data at the source which guarantees freshness in contrast to centralized repositories, *iii)* robustness and resilience against attacks and churn by exploiting self organization principles, and *iv)* simplified deployment because peers can be used and no special infrastructure is required to join the network (e.g. a new data repository can be added to a P2P network without any particular administrative task or declaration of adherence to a common schema) [16]. A P2P system (we refer to the P2P database system in the database literature) is conceived as a collection of autonomous local repositories which interact (e.g. establish correspondences or exchange query and update requests) in a P2P style. That is, local repositories are autonomous peers with equal rights and are linked to only a small number of neighbors: it allows the decentralization of the control. Furthermore, the term repository indicates that a single peer is a collection of instances, where these instances are shared to other peers allowing symmetric communication [37]. To summarize, P2P system enables two main features: autonomy (self-organization when insertion and leave of node events happen) and heterogeneity.

Although the link from the database community to the Semantic Web community is strongly enough to allow the knowledge sharing, a P2P architecture for an RDF knowledge base has been introduced by the RDFPeers [21] project, based on a DHT, which is developed on the top of Multi Attribute Addressable Network (MAAN) [22], an application layer based on Chord [89]. Each peer in this network is responsible for a segment of the whole key space, thus satisfying the "node equality" principle in terms of required network bandwidth, storage and query dependent computational load. Each RDF field is hashed and a copy of the triple is stored in the nodes

responsible for the segments where the hashes fall. Any query can then be resolved by hashing one of the constraints and contacting the node responsible for that hash ID. This approach provides load balancing based on the hash function, which assigns an ID to each input data, but it does not enforce the uniformity of such resulting values, so we may end up having a very loaded sector in the network, while others are lightly loaded. On the long run, in fact, the non-uniform distribution of the semantic statements, e.g. the high frequency of *rdf:type* or *dc:title* statements, will concentrate a lot of data in the particular sectors matching those hash ID. To overcome this issue, RDFPeers uses a successor probing algorithm which randomly analyzes a sub-set of all ranges and chooses the heaviest to be divided. In addition no active redundancy system ensures the consistency of the retrieved RDF data.

2.4 Knowledge extraction

Semantic Web applications rely on RDF and OWL to represent the knowledge that is used to describe specific contexts. As we described in the previous sections, these formal languages are machine understandable. Indeed, they enable a space where computing machinery are the first class citizen. Humans, instead, use a different way of communication and its foundation relies on the use of Natural Language (NL) to express semantics. This gap between the two different languages has been filled by Information Extraction (IE) approaches developed by the Natural Language Processing (NLP) research community [27]. The goal of the Information Extraction (IE) is to find desired pieces of information, such as concepts (hierarchy of terms which are used to point to shared definitions), entities (name, numeric expression, date) in natural language texts and print them in a form that is suitable for the automatic querying and processing.

Most of the information stored in a digital form are hidden in natural language texts for computing machinery. A promising approach to programmatically access on such knowledge uses IE techniques to reduce the natural language texts to tabular structures, from which it is possible to retrieve text tokens as answers to queries. Many IE techniques rely on predefined templates and pattern-based extraction rules or machine learning techniques to identify certain entities in text documents. Usually texts use limitless vocabularies, structures, and composition styles to define approximately the same content, making it hard for any IE technique to cover all writing pattern variations. Other approach has been proposed and they range from dictionary based and probabilistic approach. The first approach is often used when a small set of terms has to be identified in a text, instead the latter covers a large part of this techniques and usage (see section 2.5).

More important, traditional IE techniques lack the domain knowledge required to create relationships between the extracted entities. To fill this gap, the NLP

community adopted ontologies to drive the annotations. So far, for each token extracted within the text, it is classified automatically according to a hierarchy of terms (i.e ontology). Tokens mainly are key words or named entities and they differ from the identification purpose: a key word is one of the words found in a text, instead a named entity is mainly a name or an expression of time (we can define a named entity is a specialization of a key word).

Recent research trends show how the named entity recognition (NER) task has assumed an incredible role in the Semantic Web community. Mainly because, the recognition of named entities in a natural language text identify real word objects ready to be consumed in the LOD. This is crucial for legacy data which hosts huge amount of human readable information but it is inaccessible due to the time needed to extract it. So, if the output process of such techniques produces named entities, they can be disambiguated through web URIs which identify real world objects. This is the last challenge of this process and several efforts have been spent to address it.

2.5 Named entity recognition and disambiguation

The Named Entity (NE) recognition and disambiguation problem has been addressed in different research fields such as NLP, Web mining and Semantic Web communities. All of them agree on the definition of a named entity, which was coined by Grishman *et al.* as an information unit described by the name of a person or an organization, a location, a brand, a product, a numeric expression including time, date, money and percent found in a sentence [39]. One of the first research papers in the NLP field, aiming at automatically identifying named entities in texts, was proposed by Rau [76]. This work relies on heuristics and definition of patterns to recognize company names in texts. The training set is defined by the set of heuristics chosen. This work evolved and was improved later on by Sekine *et al.* [85]. A different approach was introduced when Supervised Learning (SL) techniques were used. The big disruptive change was the use of a large dataset manually labeled. In the SL field, a human being usually trains positive and negative examples so that the algorithm computes classification patterns. SL techniques exploit Hidden Markov Models (HMM) [11], Decision Trees [84], Maximum Entropy Models [17], Support Vector Machines (SVM) [3] and Conditional Random Fields (CRF) [57]. The common goal of these approaches is to recognize relevant key-phrases and to classify them in a fixed taxonomy. The challenges with SL approaches is the unavailability of such labeled resources and the prohibitive cost of creating examples. Semi-Supervised Learning (SSL) and Unsupervised Learning (UL) approaches attempt to solve this problem by either providing a small initial set of labeled data to train and seed the system [47], or by resolving the extraction problem as a clustering one. For instance, a user can try to gather named entities from clustered

groups based on the similarity of context. Other unsupervised methods may rely on lexical resources (e.g. WordNet), lexical patterns and statistics computed on large annotated corpus [1].

The NER task is strongly dependent on the knowledge base used to train the NE extraction algorithm. Leveraging on the use of DBpedia [14], Freebase[3] and YAGO [90] ontologies, recent methods, coming from Semantic Web community, have been introduced to map entities to relational facts exploiting these fine-grained ontologies. In addition to detect a NE and its type, efforts have been spent to develop methods for disambiguating information unit with a URI. Disambiguation is one of the key challenges in this scenario and its foundation stands on the fact that terms taken in isolation are naturally ambiguous. Hence, a text containing the term `London` may refer to the city `London in UK` or to the city `London in Minnesota, USA`, depending on the surrounding context. Similarly, people, organizations and companies can have multiple names and nicknames. These methods generally try to find in the surrounding text some clues for contextualizing the ambiguous term and refine its intended meaning. Therefore, a NE extraction workflow consists in analyzing some input content for detecting named entities, assigning them a type weighted by a confidence score and by providing a list of URIs for disambiguation. Initially, the Web mining community has harnessed Wikipedia as the linking hub where entities were mapped [44, 56]. A natural evolution of this approach, mainly driven by the Semantic Web community, consists in disambiguating named entities with data from the LOD cloud. In [64], the authors proposed an approach to avoid named entity ambiguity using the DBpedia dataset.

Interlinking text resources with the Linked Open Data cloud becomes an important research question and it has been addressed by numerous services which have opened their knowledge to online computation. Although these services expose a comparable output, they have their own strengths and weaknesses but, to the best of our knowledge, few research comparisons have been spent to evaluate them. The creators of the DBpedia Spotlight service have compared their service with a number of other NER extractors (OpenCalais, Zemanta, Ontos Semantic API[4], The Wiki Machine[5], AlchemyAPI and M&W's wikifier [66]) according to an annotation task scenario. The experiment consisted in evaluating 35 paragraphs from 10 news articles in 8 categories selected from the *The New York Times* and has been performed by 4 human raters. The final goal was to create wiki links and to provide a disambiguation benchmark (partially, re-used in this work). The experiment showed how DBpedia Spotlight overcomes the performance of other services under evaluation,

[3] http://www.freebase.com/

[4] http://www.ontos.com

[5] http://thewikimachine.fbk.eu/

but its performances are strongly affected by the configuration parameters. Authors underlined the importance to perform several set-up experiments and to figure out the best configuration set for the specific disambiguation task. Moreover, they did not take into account the precision of the NE and type.

In [83] Rizzo *et al.* proposed a qualitative comparison attempt, highlighting the precision score for each extracted field from 10 news articles coming from 2 different sources, *The New York Times* and *BBC*[6] and 5 different categories: business, health, science, sport, world. Due to the news articles length, they addressed a low Fleiss's kappa agreement score: many output records to evaluate affected the human rater ability to select the correct answer. Indeed, to avoid this problem, Mendes *et al.* proposed a dataset composed of pieces of news articles (paragraphs). Although this approach biases the extraction results for the One Entity per Document extractor (see section 4.2), we consider it a valid approximation for the evaluation agreement. In this dissertation, we advance these initial experiments by providing a full generic framework powered by an ontology and present detailed qualitative and quantitative experiments, focusing on the extraction performances with different type of text: user-generated content, scientific text and news articles.

[6] http://www.bbc.com

Chapter 3

Distributed ontologies for semantic annotation

With the increasing of the volume of ontology instances among the Linked Data cloud, RDF repositories suffer from data consistency and performance problems due to node failures and data fragmentation in different locations. In such a scenario information is frequently distributed on a very large number of nodes over the whole Web. When the information is distributed, multiple peers are involved in a single query and results may change over time, i.e., if a network node is not available, the retrieved data set does not include the related information. Also, if the information is randomly distributed over the whole network the query response time increases proportionally with the number of nodes. In interactive services these problems are critical because the user expects coherent results in a reasonable response time. In this chapter we address data consistency and data availability issues introducing redundancy through a dedicated architecture plugged on the top of the Web architecture were nodes are managed by a distributed load balancing logic. Moreover, we propose an annotation tool, powered on the top of the redundant architecture, which allows users to annotate multimedia contents, editing fragments and creating novel composition, just point to the fragment annotations. The annotation tool has been developed for both audio material and video contents.

3.1 Architecture

Nodes are self-organized in a P2P (peer-to-peer) network and each peer stores each own ontology instances and potentially the data generated from other peers [79]. Taking as a basic unit of data the RDF triple, the problem is to find a distribution algorithm to efficiently store multiple copies of each triple and to retrieve it in predictable time independently of the temporary unavailability of the original source.

Each peer is authoritative for the information stored into its own RDF database and, according to a specific distribution algorithm, it sends multiple copies of each RDF triple to other nodes in the network. At the same time each server provides storage space for RDF triples originated by other peers in the network. At any time any server can decide to join or leave the network, thus adding or marking as invalid some of the distributed information. Although the accessibility of a given node in the network can not be guaranteed at all times, to satisfy the data consistency and availability requirements, the information owned by the node is available at least within a specific time interval (RDF triple's time-to-live). RDF triples are indexed by multiple hash values calculated on the subject, predicate and object. Literal and typed literal values are excluded and each triple can yield from one to three hash values. Hash values are then used to locate a bucket node in a single ring according to the Mercury protocol [10]. We assign to each node a segment of the key space which can grow or reduce in order to provide uniform triple distribution, even if data is concentrated in a short key range, so every node in the network is responsible for nearly the same amount of data. Using periodical routing messages, peers exchange information on the local triple distribution in order to monitor the range with the highest load. Then, when a new node joins the network, the system uses that node to split the most loaded range so that the two nodes become authoritative for half of the triples previously contained. As can be seen in Figure 3.1, each server is

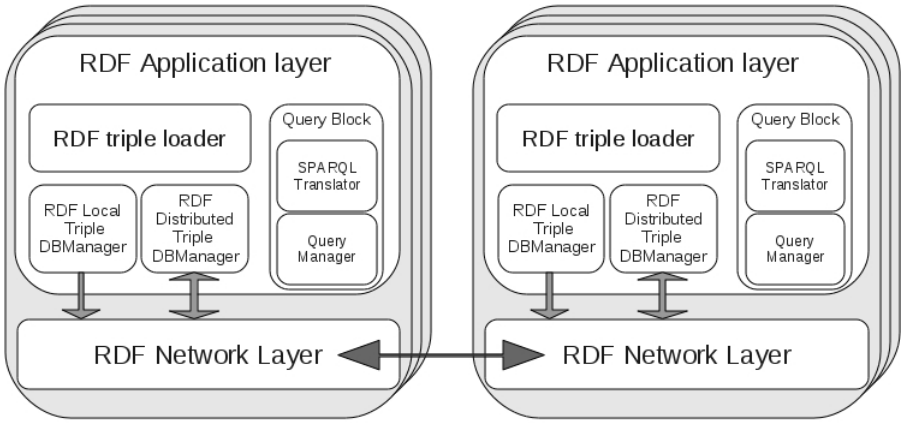

Figure 3.1. Reliable P2P RDF architecture.

organized in a stack composed by a network layer and an application layer. The underlying network layer manages the routing information in the P2P network and also manages the join/leave events and the load balancing. The application layer is composed of the RDF triple loader, used to read the RDF documents and store the information into the local RDF triple database, a distributed RDF triple database used to store information shared with the other nodes and a query block used to

interpret the SPARQL queries.

3.1.1 Message routing

Following the idea proposed in Figure 3.2 each node has a DHT composed of two sibling links connected to the previous and next node in a clockwise direction and k long distance links chosen so that the distance from the source node to the target node follows a harmonic law. This scheme provides good efficiency so that we route a generic message between two nodes in $O(\frac{log^2(n)}{k})$ [60], where k is the number of long distance links [53], that is $log(n)$ according to the Mercury protocol. A generic

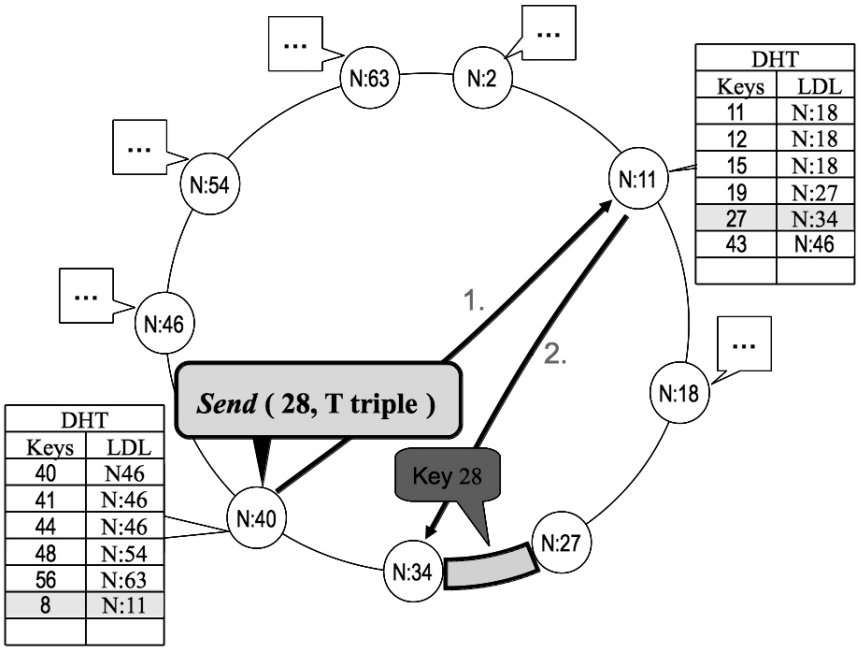

Figure 3.2. A routing of one triple into the distributed network composed of nine nodes in a 6-bit identifier space.

node m builds k long distance links with the following procedure; let I denote the unit interval [0,1] (that corresponds to the DHT ring with unit perimeter), for each such link the node draws a number $x \in I$ from the harmonic probability distribution function $p_n(x) = \frac{1}{n*log(x)}$, if $x \in [\frac{1}{n}, 1]$, where n is the number of nodes in the network. Then it establishes a long distance link with a node that is distant $p_n(x)$ from itself on the DHT ring. As shown in Figure 3.3, if it happens that the procedure uses inconsistent DHT information, the message routing algorithm can incur in routing loops as described in [35]. This can be avoided having each hop i checking the ID of the node from which it has just received the message to be forwarded (the source) so that, if the ID value of the target node is between the ID of the source node and

the ID of the previous (i) node, then it forwards the packet via the backward link. In this case the routing cost is bounded to $O(n)$.

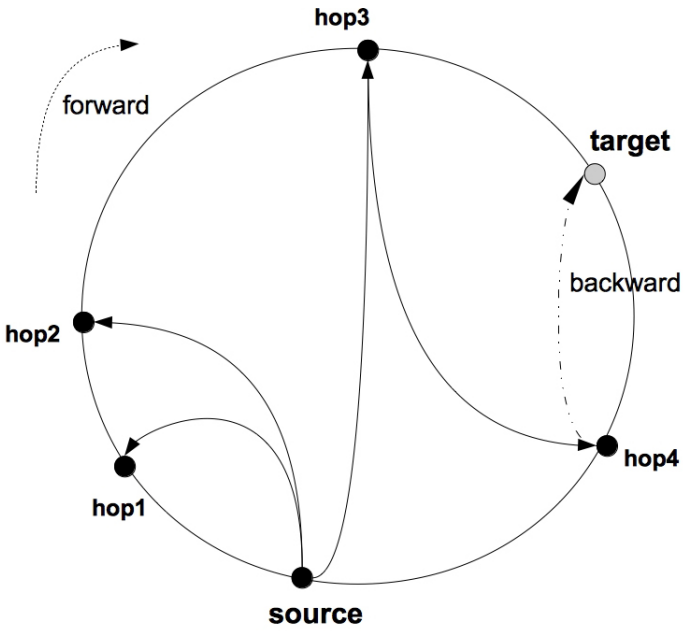

Figure 3.3. A backward-routing (source toward target) of a message in a scenario with inconsistent DHT information.

3.1.2 Load estimation procedure

In order to maintain a uniform distribution of the data among the peers, some of the algorithms proposed in this work rely on the estimation of the load of the network nodes. For this purpose we use a distributed algorithm that relies only on the communication between neighbouring nodes in the network graph and that adopts a consensus algorithm for the election of the most loaded node [30]. It has been shown in the literature that the convergence speed of consensus problems is a function of the topology of the underlying graph. While this bound can be easily calculated for fixed networks, it requires further considerations for a dynamic network topology such as a time-varying P2P network. However, it has also been shown that, as long as the union of all infinitely occurring graph instances is connected, there is a distributed consensus that will eventually converge [95] and the good convergence speed can be achieved imposing particular constraints on the different topologies. The considered P2P architecture is composed of a cloud of nodes with a regular topology where distant peers are connected by means of a number of additional short paths. Such “long distance links” allow the characterization of the network topology

as a "small world", a concept firstly introduced in [93] a phenomenon observed in many real world graphs such as large-scale social, biological and technological networks. A reasonable conjecture is that the small world graphs should result in good convergence speed for consensus problems because their low average pair wise path length should speed the diffusion of information in the system [46]. Based on the small world principle we consider a typical consensus problem to determine a convergence criterion for the election of the most loaded node. Consider a network of integrator agents: $\dot{x_i} = n_i$, where n_i is a generic node within the network and let $G = (V, E)$ be a graph with the set of nodes $V = v_1, v_2, ..., v_n$, the set of edges, $E = V \times V$ and in each node only communicates with its neighbouring nodes $N_i = \{j \in V : \{i, j\} \in E\}$. Following the idea proposed in [73] the linear dynamic system:

$$\dot{x_i}(t) = \sum_{j \in N_i} (x_j(t - \tau) - x_i(t - \tau)) \tag{3.1}$$

solves a consensus problem, where τ is the time delay for all links that we suppose, for the sake of simplicity, constant. More precisely, let $a_i, ..., a_n \in \Re$ be n constants, then with the set of initial states $x_i(0 = a_i)$, the state of all agents asymptotically converges to the average value:

$$\bar{a} = \frac{1}{n} \times \sum_{i=1}^{n} a_i. \tag{3.2}$$

A necessary and sufficient conditions for the stability of Equation 3.1 is:

$$\tau < \tau_{max} = \frac{\pi}{2 \times \lambda_n} \tag{3.3}$$

where λ_n is a measure of robustness to delay for reaching a consensus in a network. As a consequence, to guarantee the convergence of the algorithm for the selection of the most loaded node, we define a bound on the maximum delay of the load estimation algorithm limiting the recursive search on neighbour nodes to a depth given by a TTL value. Thus, we extend the result in Equation 3.1 as:

$$\dot{x_i}(t) = \sum_{j \in N_i} (x_j(t - \tau - TTL) - x_i(t - \tau - TTL)) \tag{3.4}$$

As shown in Figure 3.4 in order to compute a realistic load histogram of the network, each node periodically contacts its neighbours, listed in the DHT table, using load estimate packets.

The packet format is shown in Figure 3.5 and is composed of: (1) Time-to-Live (TTL), which defines the number of hops to discover and it represents the length of the short path, (2) Timestamp, which identifies when estimate has been calculated, (3) consensus-range-begin and (4) consensus-range-end, which describe

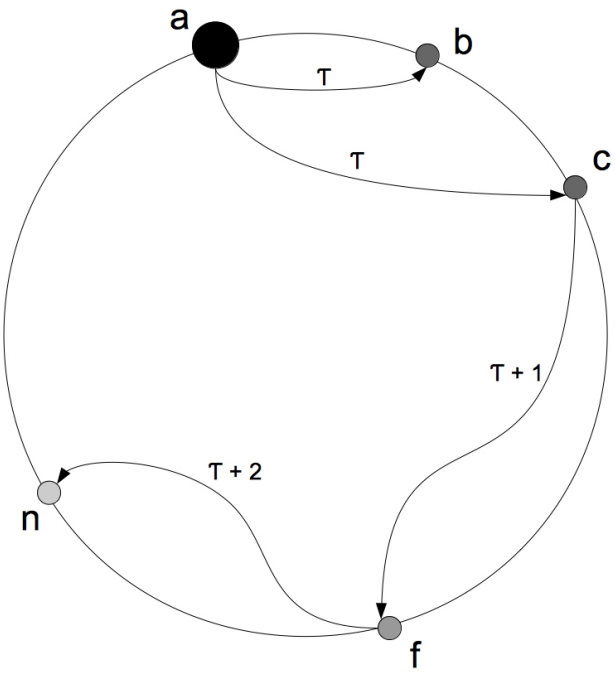

Figure 3.4. Each node contacts our directly neighbours (nodes which are indexed from the DHT) in τ seconds. Let's define n the distant factor, a generic node contacts a not directly neighbours in a $n \times \tau$ seconds.

the leftmost and rightmost key of sample estimate of the projection load estimate of the current node, (5) number of nodes and (6) average-load, which identifies the total number of triples divided by the number of total nodes, and (7) most-load-node, which describes the most loaded node in the macro-range. In order to reduce the bandwidth overhead we propose to append the load estimate information to the routing messages used to manage the network. Nodes recursively propagate these

TTL	Timestamp	consensus range begin	consensus range end	number of nodes	average load	most loaded node

Figure 3.5. Load estimation packet format.

packets using their long distance links and the TTL field is decremented by 1 for every hop. When the TTL value becomes equal to zero, the packet is sent backward and it returns to the original sender, i.e. the source node. In this way, each node can receive a feedback on the load estimation done by each of its neighbours with a delay that can guarantee the convergence of the distributed consensus algorithm. To overcome the initial state problem, the consensus algorithm performs the network

load capture periodically every t seconds and then it only takes the estimate of the nodes which directly referenced to the source node. Furthermore, because of the dynamic nature of the network, old load estimates risk to become counterproductive. Thus estimates older than a given threshold must be deleted or updated. In order to perform this, a timestamp reference is used: when this value becomes older than a threshold then a new load estimation procedure is performed. As a result of the load estimation process nodes can calculate the distribution of the load among the peers and the number of peers in the network.

3.1.3 Join and leave events

When a new node A wants to join the P2P network, it contacts a known node B that, as a function of the estimated load information, redirects A to the most loaded peer it is aware of, C. The joining node A is then inserted in the ring as the predecessor of C. As a consequence, the data range previously managed only by C is evenly split between C and A, i.e., A acquires half of the triples that have been on C, according to the principle of consistent hashing [49]. Obviously, groups of triples stored under the same hash could not be split and therefore should be considered as grains among the triples. Since the triples are stored according to the alphabetic order, C flushes each data whose hash is greater than the ID assigned to the new node.

The leave event can require two different actions since we may want to preserve the consistency of the data or we may want to remove portion of the knowledge from the network. In the former case the event requires that the leaving node transfers its triples to its successor (following the principle of consistent hashing) that has to update its responsibility range. In addition, the leaving node forwards through its long-distance links [65] a leave message so that these peers can update their DHT to match the new state of the network. In the latter case, the leaving node may also want to remove from the network the triples it is owner of, so it requests the deletion of these triples and their redundant copies. In the rest of the paper we consider as a leave event only the case that can also be forced, for example, by a link failure or a temporary node unavailability.

3.1.4 Redundancy

A redundancy algorithm is proposed to address the issues related to temporary node unavailability (or node disconnection). Triples are in fact replicated in more than one peer according to their hash function, using the following equation:

$$\sum_{i=0}^{i=t-1} hash(x) + \frac{\omega * i}{t}, \tag{3.5}$$

where x is a URI field of a statement, ω is equal to 2^{hash_length} and t identifies the number of replicas. Figure 3.6 shows an example of the redundancy mechanism where each triple has an additional copy of another peer with $t = 2$. This algorithm

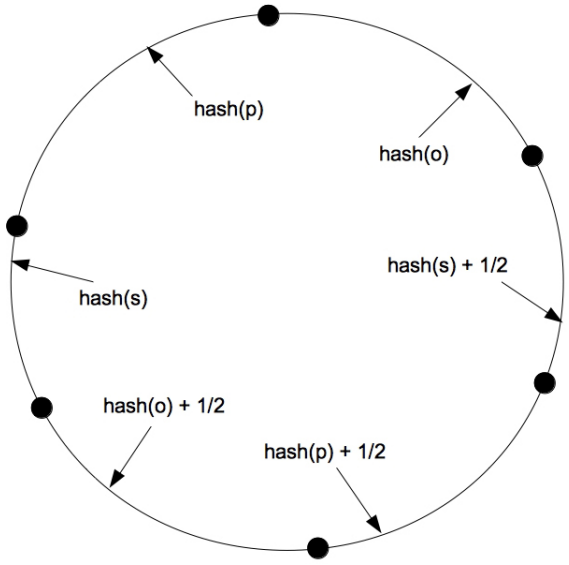

Figure 3.6. Redundant copies of a given triple (s,p,o) with t=2.

does not group all triples replica of a node in one peer. When the network must substitute a node that has left, it uses the above formula to transfer the redundant triples of the dead node to its successor. In this case the node that receives the triples does not need to replicate them again.

3.1.5 Query resolving

In the proposed system any client outside the P2P network can use the SPARQL standard, the graph-matching query language used to retrieve semantic knowledge, on any P2P node to query the network knowledge base. The distributed RDF architecture can then return the same results in a predictable time independently from the particular node used to perform the query. The basic idea that we use for querying the knowledge base is to hash the exact value of at least one triple field. Then, these hashes identify the nodes which store the requested triple-set. In the case of multiple exact fields available in a query, we use a query parsing algorithm for rewriting the query to minimize the cost of answering it. This cost is affected by at least three elements: the size of the knowledge base (in particular the number of nodes which are involved in the query), the strategy followed to combine the data and the order or plan in which data is processed. In our context, i.e., a highly dynamic

P2P scenario, new techniques to identify an efficient query evaluation strategy are needed.

For what concerns the query parsing algorithm, the SPARQL Query Graph Model (SQGM) [42] has been proposed as a method to build an optimizer such that it can elaborate a right strategy for combining the middle results. However, this approach relies on a central node for building a global graph and it requires the collection and transfer of a huge amount of data across the P2P network. On the contrary, we suggest using a distributed algorithm since we have a global knowledge base composed of a pool of P2P nodes which make unfeasible to share a global graph as in the SQGM proposal. In [88] a most efficient technique has been presented that constructs optimized heuristics as a function of the number of occurrences of an element within the knowledge base, i.e., as a function of the element selectivity. Although this implementation has been developed for a centralized environment, we extend it for a distributed scenario where the network is able to collect the selectivity information of all elements owned by the P2P nodes. To evaluate the performance of query execution, we consider the query propagation time between nodes. We assume an upperbound delay for the propagation time between two nodes, which is called t, and that, for simplicity, we suppose constant and equal for all nodes. Thus, we express the total query time as:

$$QT = \sum_{s} ii = 1^{n} C_i + R \tag{3.6}$$

where C_i is equal to the sum of the computing time of all nodes and R is equal to the routing time.

Atomic query

An atomic query pattern is a triple in which the subject, predicate and object can each be a variable or an exact value. In this case, the query block just uses the hash function on the query pattern exact values to identify the node which manages the requested triple-set. An example of an atomic query is represented by the form $(?s, p, o)$, where the question mark indicates the unbound or variable value. Following the idea proposed in [88] we define the selectivity of a triple pattern as $s(t) = s(s) \times s(p) \times s(o)$, where $s(t)$ denotes the selectivity for the triple pattern t , $s(s)$ the selectivity for the subject s, $s(p)$ the selectivity for the predicate p, and $s(o)$ the selectivity for the object o. The (estimated) selectivity is a real value in the interval [0,1] and corresponds to the (estimated) number of triples selectivity for the subject s matching a pattern, i.e., the number of triples matched, normalized by the total number of triples in the RDF data-set. The selectivity of a unbound triple field, i.e., $s(s)$ or $s(p)$ or $s(o)$, is generally equal to 1.0 , otherwise the selectivity of an exact value is equal to T_e/T , where Te denotes the number of occurrences

of a triple element and T the total number of triples in the entire knowledge base. Thus, a low selectivity value means a high element selectivity and viceversa. A general representation of an ontology, which use the RDF notation for representing its data, is generally composed of an absolute number of subjects at least one order of magnitude greater than the absolute number of predicates [72]. According to Equation 3.6,

$$\sum_{i=1}^{n} C_i = s(e) \times C_s + C_t, \tag{3.7}$$

where $s(e)$ is the selectivity of the element and C_s and C_t are, respectively, the computing time of the source node and of the target node. In addition, since R depends linearly on the number of nodes interested in the query times and the routing protocol guarantees that a generic node can be τ and the routing protocol guarantees that a generic node can be reached in $a = O(log(n)/k)$, we can state that R is upper-bounded by $a \times \tau$. As a consequence, the computation time of the target node depends to a great degree on the number of triples owned by a node and on the number of potential selectable triples. For this reason, to minimize the computation time required, the query is sent to the node with the higher selectivity. A different case is represented by the queries in the form $(?s, ?p, ?o)$. This is the most general and the most expensive query which matches all triples. Since there is no restriction whatsoever on this triple pattern, we have to contact all nodes, which takes $O(n)$ routing hops for a network with n nodes. For this reason, the $QT = C_s + n \times \tau$, where C_s is the source node computing time.

Disjunctive query

A disjunctive query pattern is an atomic query with a list of constraints. It is then evaluated almost as an atomic query, but the node responsible for such triple-set has to return only the data that matches the specified constraints. For example, consider these two queries:

1. $(?s, foaf : name, ?o)$, ?o="Picasso" OR ?o="Modigliani"
2. $(?s, media : length, ?o)$, ?o > "54" AND ?o < "99"

They are divided into two separate atomic triple patterns and they are sent to two different nodes. As described previously the single queries are sent to the nodes with highest selectivity, the results are then collected by the source node. Let N the number of atomic queries generated, following Equation 3.6 we have:

$$QT = \sum_{i=1}^{n} C_i + \tau \times d, \tag{3.8}$$

where d is upper-bounded by $O(N + \frac{log_s(n)}{k})$ and C_i is the value of the computation time of the i^{th} node.

Conjunctive query

A conjunctive query pattern, or simply a joint query, is the conjunction of a list of query patterns. Consequently, we need to join two or more sets of triples that are spread on different nodes. We adopt the query chain algorithm (QC), described in [58]. For example, consider the query $(?x, foaf : name, "JohnnyLeeOutlaw")$ and $(?x, foaf : mbox, ?mbox)$, this schema splits the whole query in sub-queries and solves them separately as an atomic query. The result of the sub-queries are then collected and joined to form the requested triple-set. Assume a node j that poses a conjunctive query q which consists of triple patterns $q_1, ..., q_k$. Each triple pattern of q will be evaluated by a different node; these nodes form the query chain for q. The order we use to evaluate the different triple patterns is decided by the query block, which rewrites the query in order to optimize the number of triples selected by the first nodes. Thus, a source node issues the most constraint query to the first node. Then, this one issues the rest of the query and the result of the first sub-query to second node. The last node in this chain computes the triples required and sends the results to the source node. In this way we reduce the number of triples exchanged between the source node and the nodes involved into the query. Let consider N the number of sub-queries, from Equation 3.6 we have:

$$QT = \sum_{i=1}^{n} C_i + \tau \times c, \tag{3.9}$$

where c is upper-bounded by $N \times O(\frac{log_s(n)}{k})$.

3.2 Evaluation

To asses the performance of the proposed architecture we developed a specific simulator that reproduces the behaviour of a medium size network with three hundred nodes and one million of triples in the data-set. First, we focus on the analysis of the load balancing technique; we study the distribution of the triples among the peers and then the network bandwidth overhead introduced by this feature. Two basic scenarios are considered, a) static, where nodes sequentially join the network monotonically increasing its size, b) dynamic, where nodes join and leave the network. In each join event the new node inserts 3,334 new triples, which is the average value of triples in each node of the existing network.

3.2.1 Load balancing

Figure 3.7 shows the experiment where 300 nodes are sequentially inserted in the P2P network and the RSD (Relative standard deviation) value is reported after each

insertion. When a node joins the network, 3,334 triples are added to the system and the new node becomes the predecessor of the most loaded one inheriting half of its triples. The plot shows that, apart from the very beginning of the simulation when few nodes are present, the relative standard deviation is kept almost constant ensuring a good load balancing among the peers. Variations are mostly due to the uneven distribution on the P2P network of the new 3,334 triples that come with the joining nodes. Figure 3.8 shows the effect on the RSD of ten series of ten join

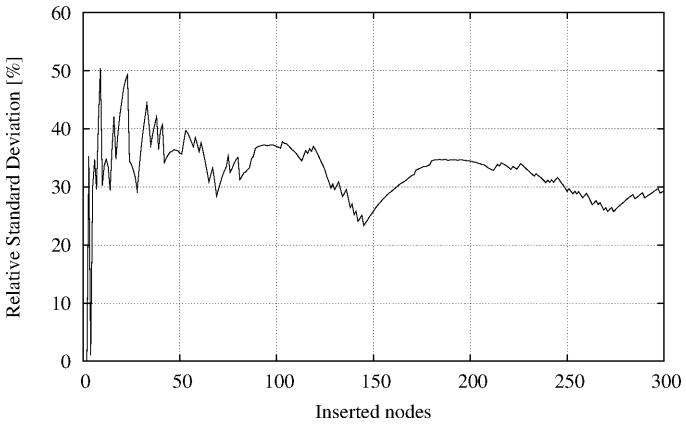

Figure 3.7. RSD of the number of triples managed by each peer in the static scenario experiment.

and ten leave events, after the insertion of 300 nodes. Peaks correspond to the leave events when the leaving node triples must be moved to the successor node drastically increasing its load with respect to the other peers. Valleys are instead related to the join events when a new peer joins the network and helps splitting the triple range managed by the most loaded peer.

Since each joining node carries a relative large amount of new random triples the RSD delta after a series can either be positive or negative depending on the new triple hashes. For example series I, II, III, IV (from event 300 to event 380) increase the RSD, while series V decreases it. However series that increase the RSD tends to occur more frequently, because, as previously shown in Figure 3.8, the bandwidth saving load balancing algorithm applied to the insertion events can only slowly reduce the RSD.

3.2.2 Network bandwidth

For what concerns the network bandwidth used by the proposed algorithm to guarantee load balancing and triple redundancy, the effects of data exchange in the two previous scenarios (static and dynamic) are shown in Figure 3.9 and Figure 3.10. The number of triples exchanged are normalized with respect to the number of

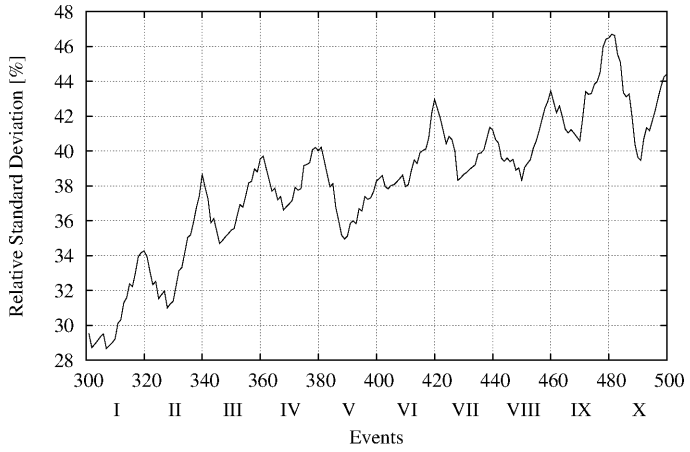

Figure 3.8. RSD of the number of triples managed by each peer in the dynamic scenario experiment.

triples each node inserts in the system (e.g. 3,334). So, when a new node joins the network most of its triples move into the ranges managed by other peers and half of the triples of the most loaded peer moves into the node hash range. When a node leaves the network its triples move to the node successor.

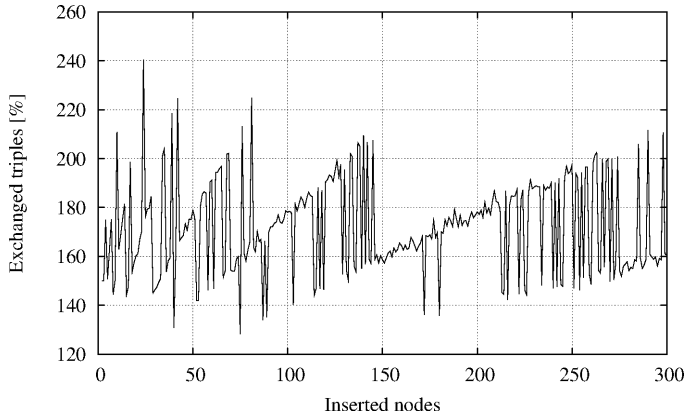

Figure 3.9. Network bandwidth usage in the static scenario.

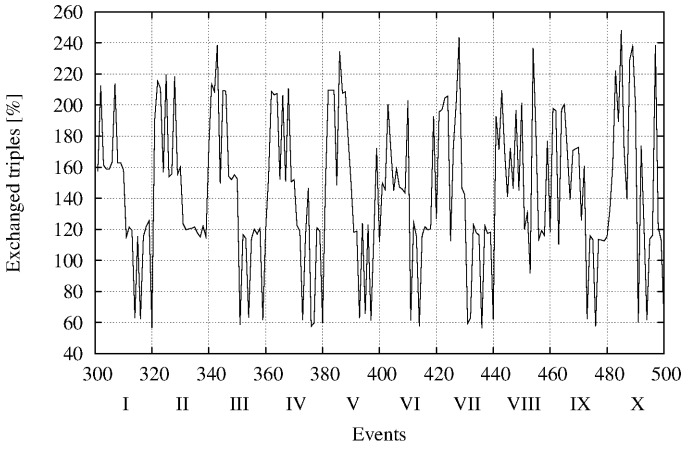

Figure 3.10. Network bandwidth usage in the dynamic scenario.

3.3 Ontology driven multimedia annotations

We aim to create an efficient and scalable architecture for semantic annotations of multimedia contents and to implement this architecture to augment the user-interaction for cataloging, retrieving and visualizing of composed multimedia contents. The first step is to adopt a specialized ontology that will hold those information. In this scenario [29, 51] demonstrate that "logically extensive" upper-level ontologies are extremely hard to agree on, to build, maintain, understand and use. This seems to give enough evidence that a lightweight upper level ontology is what semantic annotations need as a ground for representing the annotation concept and for describing the community interactions among the users. In the scenario of global definition of semantic for managing Authoring Tagging System in online communities, one of the main work is Semantically-Interlinked Online Communities (SIOC) Core Ontology [15]. Authors proposed new scenarios for online community site data and lets developers build innovative semantic applications on top of the existing Social Web, by means of the SIOC ontology.

Many researchers investigated in ontology creations for representing the semantic of media elements [34, 96]. They obtained different results which addressed one particular topic, e.g. to represent taxonomy of a kind of media or to identify event sequences or concepts. These outlined the difficult to create a standard video description domain. One of our goal is to preserve the semantic meaning of the multimedia content, wrapping it in a reference of the above ontology and augmenting its information by means of user annotations. For such reason we focus on the representation of a media as simple item, which is described by means of tag concept. First of all, we map our usage-scenario in the above ontology; in particular we need to represent multimedia contents and user interactions in a common way, by means of

ontology entities and inferences. We use the Item concept as the envelop of our main element (multimedia stream), after this we use the community concept to exploit the Friend of a Friend (FOAF) inferences among users. The information needed to represent a tag process is stored in a Tag entity. The inference between the multimedia content (Item) and the Tag is provided by means of the topic inference. To summarize, we use a sub-set of the overall entities contained in SIOC ontology to manage the relation between media elements and their annotations. In particular, the Item represents multimedia content descriptions (e.g. start-time, end-time, duration, streaming protocol, streaming source locator, etc.), the Tag holds category selection list and descriptions. Each user may perform the Post of any new resource (multimedia content) and may infer the Tag information, by means of topic inference. We adopt Item as a generic class to describe a multimedia item, allowing mappings with most specialized ontologies. As described previously, the ontology represents an upper-level domain description, which is composed of entities linked using relations. These data are mapped in triples *(s,p,o)* and stored in a Resource Description Framework (RDF) repository. In particular, our data-base holds semantic information about audio and video streams: streaming sources locators, per-user annotated media fragments and per-user augmented composite streams. Streaming source locators are used to identify and access standard audio and video streams over the Internet. Users can tag, annotate and split into fragments such streams using tools provided by the Authoring System. Also, they can import any semantic information embedded in the stream itself using a standard data format like RDF or link to knowledge available on the originating server using a SPARQL endpoint. Figure 3.11 shows the architecture. The Representation State Transfer (REST) ap-

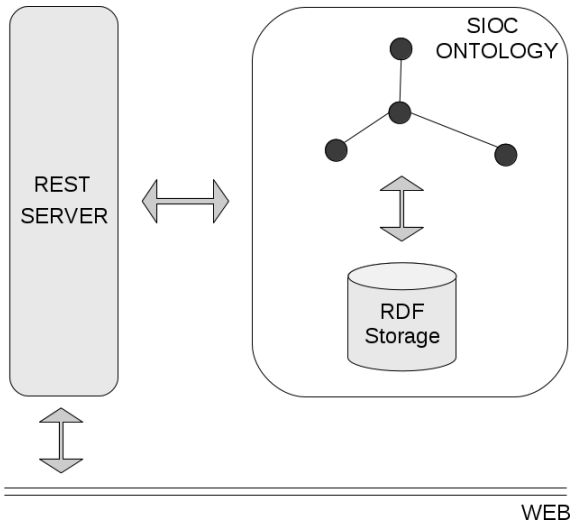

Figure 3.11. The architecture of our Authoring System aimed to get the user demand and to show the inferred data according to the ontology schema.

plication server manages web resources between the outlined system with customers and it provides an interface for accessing the semantic data. The data is stored according to the SIOC ontology schema in the distributed RDF storage. A central system, Authoring System, is in charge to interact with clients through HTTP and to infer from the distributed RDF storage the RDF data needed for the communication. We created an annotation tool, which is composed of a client side and a server side adapter, powered on the top of the Authoring System, able to show to drive users through the annotation and to store all the user generated data. Then we applied this annotation tool on two use cases: a video editing scenario and a sonic material editor, where users can edit in real time the resources just using the web experience. Figure 3.12 shows the agents involved in this scenario and shows a mix of new stream, which is composed of many parts of contents created by the community and localized within the Web. A user client, through the annotation tool, communicates with the Authoring System by means of a HTTP/SPARQL channel. The Authoring System indexes the required resources and sends to the user a list of meta-information according to the demand. This envelop included the URI reference where the user client may stream the multimedia content, with additional information like descriptions and time-line parameters. The user client communicates with multimedia content servers by means of RTSP/RTP (or similar streaming protocol).

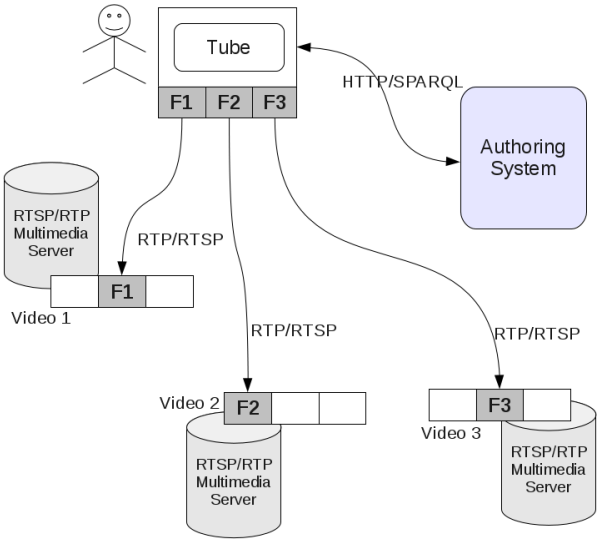

Figure 3.12. The interaction from the end-user and our Authoring System aimed to discover the meta-data information indexed into our repository.

3.3.1 Media annotation tool

Multimedia contents have a rich semantic structure. They communicate a message, designed by the media maker, which ranges from nonsensical or abstract to symbolic (e.g. a piece of film music supporting a clear narrative). A majority of media production nowadays is done with software tools, which give rise to various new opportunities to monitor the production process. The annotation tool is a web tool for annotating any kind of media resource. It can load any online available OWL ontology and guide users through the annotation process with a simple user interface. When starting the annotation process the user is allowed to select the ontology that deals with the aspect of the resource he wants to make statements about. In this context we present an open web framework that can improve the user experience during the creative process, by means of ontology driven annotation. Communities, that use the free text annotation method, are affected by a set of problems, like polysemy, synonymy, data scarcity, spelling errors and plurals. Polysemous tags can return undesirable results. For example, in a music collection when a user searches for the tag *love*, results could contain both love songs and songs that were tagged as such because user liked them very much. Tag synonymy is also an interesting problem. Even though it enriches the vocabulary, it also presents inconsistencies among terms used in the annotation process. According to [62], *bass drum* sounds can be annotated with the *kick drum* tag, but these sounds will not be returned when searching for *bass drum*. To avoid this problem, sometimes users tend to add redundant tags to facilitate the retrieval (e.g. using *synth*, *synthesis* and *synthetic* for a given sound). The tool provides an intuitive Web user interface that lets users choose one of the classes in the ontology. Finally, the annotations are converted to the RDF syntax. The annotation tool consists of a client side and a server side component.

Client side

The client-side component is a graphical user interface consisting of boxes, menus and input fields to let the user navigate the classes provided by the ontology. It also allows to choose one or more classes, and specifying the value for the attributes of a class, if present. According to the SoC (separation of concerns) guidelines, we developed our tool using HTML for page markup, CSS for graphical style and JavaScript to handle the program logic and user interactions. The jQuery framework[1] was used to manipulate the Document Object Model (DOM) and the jQuery UI[2] utilized for the GUI components like autocomplete, datepickers and complex

[1] http://jquery.com/

[2] http://jqueryui.com/

behaviour handlers like draggable and droppable. The tool has been developed with attention to modular programming. In order to allow other developers to reuse the code, our annotation tool was divided into three reusable modules: owl.js, owl-ui.js and owl-ui.media.js.

- owl.js: requests an interpretation of a specified ontology from the server side component and converts this to an internal data model.
- owl-ui.js: is responsible for the creation of the annotation tool panel, composed of menus and dynamic textboxes. It requires the owl.js library to populate the user interface widgets with the information retrieved from the ontology.
- owl-ui.media.js: creates an interface to annotate audio files. It allows the user to listen to files and to select a sub part of a media in order to annotate it. Then, it allows opening of the annotation tool panel generated by the owl-ui.js library in order to annotate a media with ontology classes.

These libraries can be embedded into any web page, making it particularly easy for a developer to add the annotation feature to his own web application. Furthermore, it would be relatively easy to develop special user interface components for annotating other types of documents, like video or text.

Server side

The second part of our tool is the server-side component. It is a SPARQL Protocol and RDF Query Language (SPARQL) endpoint which makes queries over the ontology and retrieves all classes, properties and attributes. The response is generated in JavaScript Object Notation (JSON) format (but it is possible to request different output formats, like raw text and XML) and it is returned to the client side. JSON notation is used because it is a lightweight data-interchange format, it is readable by humans and can be easily converted from text to JavaScript object. The SPARQL endpoint runs on a Linux machine with the Apache[3] web server running and the PHP[4] language available. Furthermore, the endpoint uses the Redland RDF[5] libraries to interpret the data from the ontologies and they are a key component of our framework.

When the annotation tool is initialized, it makes a synchronous call to the SPARQL Endpoint hosted by a server machine and it sends three main parameters: the URL of the ontology to query, the SPARQL query to execute and the

[3] http://www.apache.org

[4] http://www.php.net

[5] http://librdf.org

format of the response. The annotation tool receives a response, by default a JSON object, containing each class and subclass of the ontology. Processing this data our tool creates a JavaScript structure of objects containing the complete ontology hierarchy of classes. At this point the annotation tool creates the user interface populated with data retrieved from the ontology. The resulting GUI widget includes a textbox, in which dynamic suggestions are provided by means of the autocomplete feature. The user can traverse the class hierarchy through a tree menu to choose a concept related to the resource he is annotating. When the user selects a class from the menu or from the textbox he is presented with a new widget where the user can assign a value to each attribute of the class. The tool chooses the right widget for each possible attribute type. The annotations are collected into a stack and when the annotation process is completed, the user confirms the annotation. The tool generates an RDF representation of the annotations and subsequently sends it to a server where it could be stored in a triple store and retrieved later. Thanks to namespaces and URIs that identify uniquely a resource, the generated RDF/XML annotation holds the complete semantic description and the information the user has associated with the resource.

3.3.2 Real-time web video editing

As a use case of the annotation tool, we developed a web application for editing multimedia items coming from different media streaming server [80]. A web user is able to compound new Items which are made merging the semantic annotations of different Items. An important role is represented by the Item time-line. A user may select a part of the Item, which is called fragment and is characterized by a start-time and an end-time, and may create an annotation. In this way, the Item may be made from many fragments or annotations. Then, a user may divide the Item in a sequence of new elements, which are meaningful for him in a semantic way. Augmented information are expressed by means of triples and sent or gathered from the user through HTTP/SPARQL standard protocol. Additionally, the gathered information can be assembled by users to define new virtual media streams that, from the consumer point of view, are played as a sequence of audio and video fragments that are joined to form a novel creative composition. The audio and video footage of this new composition is not stored anywhere in the network. It is provided in real time by the Authoring System, which will use streaming source locators to transparently join right media fragments following a user's play request. For such reason, the multimedia content remains the same and it is not copied in any clients, but it is stored only in one single source. Then, our Authoring System limits the network bandwidth consumption (it only exchange the RDF data) and it prevents the data redundancy among the different users. The informational content, related to original media streams available in the Authoring System, increases implicitly

with the number of annotated fragments. Those are created by the user-interaction through the mix of semantic relations. Even though this architecture is strictly dependent from the original source (the source change may require a new index schema for all fragments), on the other hand, it provides a consistency check on data represented by annotations. In addition, this kind of information becomes available for all users and, in a trusted network, the global information grow continuously with community interactions. Figure 3.13 shows how the annotation web interface looks like.

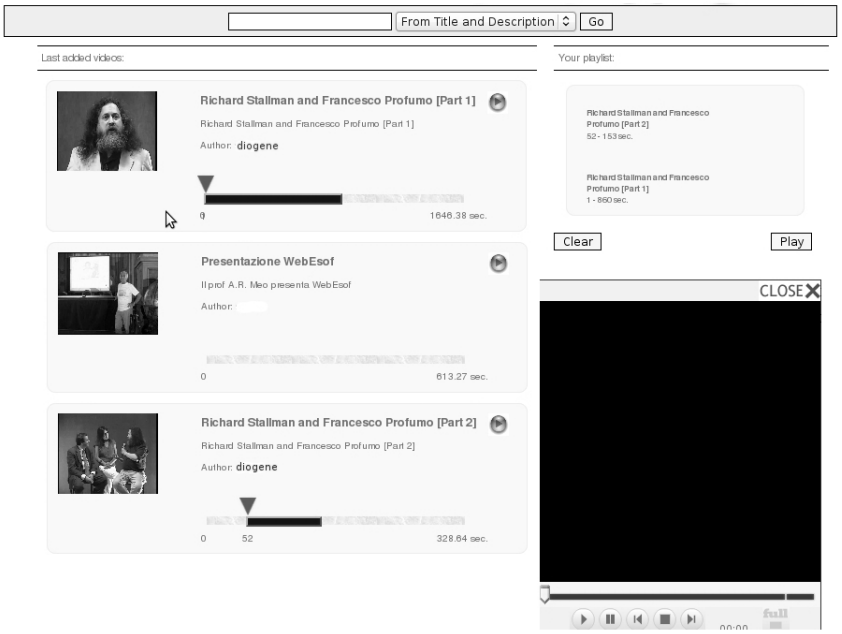

Figure 3.13. The web interface developed for the human beings to edit video items collected from different streaming sources.

3.3.3 Sound making through ontology

In the context of helping sound maker to create sound composition, we developed web-based audio sequencer [78]. We used the standard mark-up language designed for the Web, HTML, graphical customization allowed by CSS stylesheets and we handled the business logic and user interactions with JavaScript. We also tried to exploit the multimedia capabilities of the new version of the HTML standard, but our project required advanced audio synchronization features that HTML5 Audio does not yet provide. We had to fall back to Adobe Flash technology that is responsible for handling audio playback.

- The Audio Engine Layer is responsible for the playback of the audio. It retrieves audio files from the Web, then it synchronizes tracks and handles the virtual timeline. In addition, it has features to mute and solo a track and change its volume. It also permits looping a section of the composition and includes a metronome. It communicates with the Communication Layer described below.

- The Communication layer controls bidirectionally the Audio Engine Layer and the Graphical Interface of the application. When a user performs an action, it is handled by the Communication Layer that transmits the instructions to the Audio Engine Layer. It also receives events from the Audio Engine Layer and updates the User Interface.

- The Graphical User Interface handles all interactions with the user through drag functionalities, buttons, sliders and editable text fields. When a user interacts with the GUI, the Communication Layer propagates the action to the Audio Engine.

Through this application users can mix sounds available over the web simply using the URL of the audio resources. It implements the basic functionalities of every sequencer, like audio playback, visual tracks synchronization and looping, so that users can create their own audio composition. We also integrated searching of the Creative Commons[6] licensed sound repository Freesound[7], so that users can retrieve sounds from a large repository.

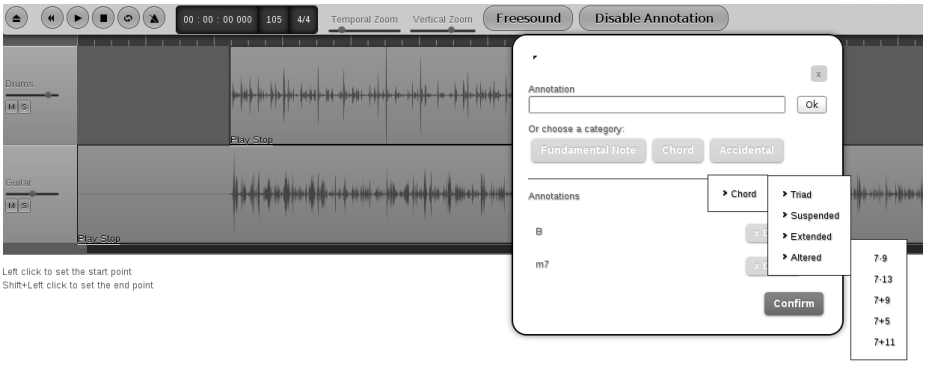

Figure 3.14. Integration between the Web audio sequencer and the annotation tool. In this case the user loaded the Chord Taxonomy, in order to describe the harmony of the musical samples included in the composition.

[6] http://creativecommons.org

[7] http://www.freesound.org

As shown in Figure 3.14 the annotation tool has been integrated into this sequencer. A user can use it to give the semantic description of an audio file content. By clicking on the Annotation button the user has the opportunity to choose an argument: every argument corresponds to an existing ontology. The current implementation proposes the Sound Producing Events ontology recommended for natural sounds and the taxonomy on Chords, useful to describe the harmony of a music track. Adding new ontologies is really simple for a developer, due to the fact that every ontology is an OWL file located over the Web. After this choice, the GUI is enriched by some new buttons and text. At this point the user can select a portion of an audio sample from the sequencer composition and choose to annotate it. Then, from the annotation tool panel that appears on the screen, he or she can listen to portions of the samples and select concepts that better describe the content (Figure 3.15). In this way we improve the granularity of the annotation, enriching the semantic description of an audio resource.

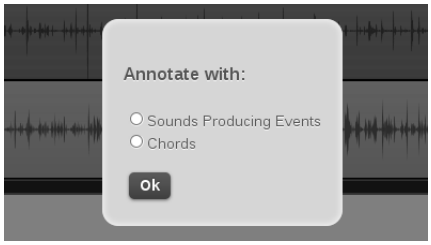

Figure 3.15. In our use case, we propose two ontologies. The first is useful in annotating natural sounds, while the second is useful in describing the harmony aspect of a music track.

The technologies used permit easy integration of the annotation tool on every website. What is needed is to include the Javascript libraries into the site code. Furthermore, the possibility to plug in any ontology available on the web makes the tool a possibly useful instrument for web sites that want to include annotation functionality.

Chapter 4

Knowledge extraction from unstructured data

The Web hosts millions of unstructured data such as scientific or medical papers, news articles as well as forum and archived mailing list threads or (micro-)blog posts. This information has usually a rich semantic structure which is clear for the author but that remains mostly hidden to computing machinery. Natural Language Processing (NLP) and information extractors aim to bring such a structure from those free texts. They provide algorithms for analyzing part-of-speech (POS) terms which occur in a sentence and identify Named Entity (NE) such as name of people or organizations, locations, time references, quantities, etc. classifying them according to predefined schema, increasing discoverability (e.g. through faceted search), reusability and the utility of information. The named entity extraction is a mature task in the NLP field that has yielded numerous services gaining popularity in the Semantic Web community for extracting knowledge from Web documents. These services are generally organized as pipelines, using dedicated APIs and different taxonomies for extracting, classifying and disambiguating named entities. The integration of one of these extractors in a particular application requires to implement an appropriate driver. Furthermore, the results of these services are not comparable due to different formats. This prevents the comparison of the performance of these services as well as their possible combination. We address this problem by proposing NERD, a framework which unifies 10 popular named entity extractors publicly available on the Web, and the NERD ontology which provides a rich set of axioms aligning the taxonomies of these tools. To highlight strengths and weaknesses of these extractors we propose two set of evaluations: a qualitative experiment, where we performed two human evaluation campaigns and a quantitative experiment where we performed extraction experiment and we grouped them according to the NERD ontology.

4.1 The NERD framework

NERD is a web framework plugged on top of the following NE extractors: AlchemyAPI[1], DBpedia Spotlight[2], Evri[3], Extractiv[4], Lupedia[5], OpenCalais[6], Saplo[7], Wikimeta[8], Yahoo! Content Analysis (YCA)[9] and Zemanta[10]. Its architecture follows the REST principles [32] and includes an HTML front-end for humans and an API for computers to exchange content in JSON and according to NLP Interchange Format (NIF) specification[11]. Both interfaces are powered by the NERD REST engine. Figure 4.1 shows the workflow of an interaction among clients (humans or computers), the NERD REST engine and various NLP tools which are used by NERD for extracting NEs, classification types and list of pointers to real world objects as they could be defined in the Web of Data.

4.1.1 The NERD Data Model

We propose the following data model that encapsulates the common properties for representing NERD extraction results. It is composed of a list of entities for which a label, a type and a URI is provided, together with the mapped type in the NERD taxonomy, the position of the named entity, the confidence and relevance scores as they are provided by the NER tools. The example below shows this data model (for the sake of brevity, we use the JSON syntax):

```
entities: [{
      entity:"Tim Berners-Lee",
      type:"Person",
      uri:"http://dbpedia.org/resource/Tim_berners_lee",
      nerdType:"http://nerd.eurecom.fr/ontology#Person",
      startChar:"30",
```

[1] http://www.alchemyapi.com

[2] http://dbpedia.org/spotlight

[3] http://www.evri.com/developer/index.html

[4] http://extractiv.com

[5] http://lupedia.ontotext.com

[6] http://www.opencalais.com

[7] http://saplo.com

[8] http://www.wikimeta.com

[9] http://developer.yahoo.com/search/content/V2/contentAnalysis.html

[10] http://www.zemanta.com

[11] http://nlp2rdf.org/nif-1-0

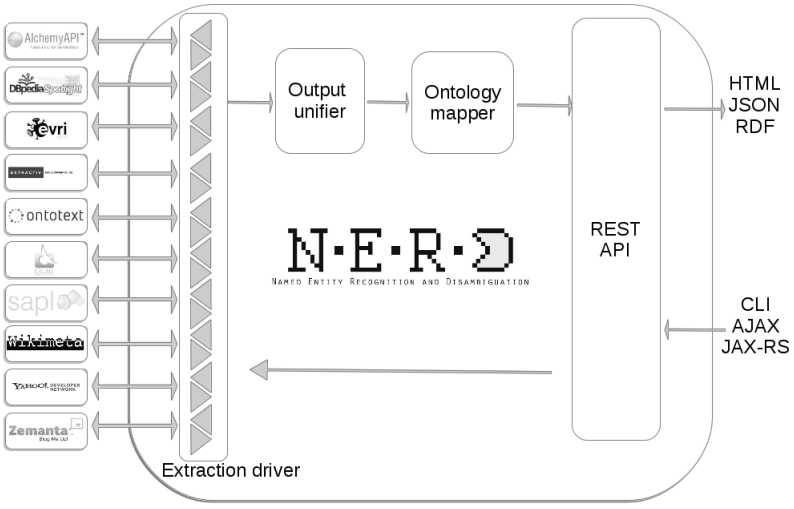

Figure 4.1. A user interacts with NERD through a REST API. The engine drives the extraction to the NLP tool. The NERD REST engine retrieves the output, unifies it and maps the annotations to the NERD ontology. Finally, the output result is sent back to the client using the format reported in the initial request.

```
 endChar: "45",
 confidence: "1",
 relevance: "0.5"
}]
```

4.1.2 The NERD REST API

The REST engine runs on Jersey[12] and Grizzly[13] technologies. Their extensible framework allow to develop several components and NERD is composed of 7 modules namely authentication, scraping, extraction, ontology mapping, store, statistics and web. The authentication takes as input a FOAF profile of a user and links the evaluations with the user who performs them (we are freezing an OpenID implementation and it will replace soon the simple authentication system working right now). The scraping module takes as input the URI of an article and extracts all its raw text. Extraction is the module designed to invoke the external service APIs

[12] http://jersey.java.net

[13] http://grizzly.java.net

and collect the results. Each service provides its own taxonomy of named entity types it can recognize. We therefore designed the NERD ontology which provides a set of mappings between these various classifications. The ontology mapping is the module in charge to map the classification type retrieved to our ontology. The store module saves all evaluations according to the schema model we defined in the NERD database. The statistic module enables to extract data patterns form the user interactions stored in the database and to compute statistical scores such as the Fleiss Kappa score and precision/recall measures. Finally, the web module manages the client requests, the web cache and generates HTML pages.

Plugged on the top of this engine, there is an API interface[14]. It is developed following the REST principles and it has been implemented to enable programmatic access to the NERD framework. It follows the following URI scheme (the base URI is `http://nerd.eurecom.fr/api`):

`/document` : GET, POST, PUT methods enable to fetch, submit or modify a document parsed by the NERD framework;

`/user` : GET, POST methods enable to insert a new user to the NERD framework and to fetch account details;

`/annotation/{extractor}` : POST method drives the annotation of a document. The parametric URI allows to pilot the extractors supported by NERD;

`/extraction` : GET method allows to fetch the output described in section 4.1.1;

`/evaluation` : GET method allows to retrieve a statistic interpretation of the extractor behaviors.

4.1.3 The NERD Ontology

Although these tools share the same goal, they use different algorithms and different dictionaries which makes hard their comparison. We have developed the NERD ontology, a set of mappings established manually between the taxonomies of NE types. Concepts included in the NERD ontology are collected from different schema types: ontology (for DBpedia Spotlight, Lupedia, and Zemanta), lightweight taxonomy (for AlchemyAPI, Evri, and Yahoo!) or simple flat type lists (for Extractiv, OpenCalais, Saplo, and Wikimeta). The NERD ontology tries to merge the linguistic community needs and the logician community ones: we developed a core set of axioms based on the Quaero schema [36] and we mapped similar concepts described in the other scheme. The selection of these concepts has been done considering the

[14] `http://nerd.eurecom.fr/api/application.wadl`

greatest common denominator among them. The concepts that do not appear in the NERD namespace are sub-classes of parents that end-up in the NERD ontology. This ontology is available at http://nerd.eurecom.fr/ontology. To summarize, a concept is included in the NERD ontology as soon as there are at least two extractors that use it. The NERD ontology becomes a reference ontology for comparing the classification task of NE extractors. We show an example mapping among those extractors below: the `City` type is considered as being equivalent to `alchemy:City`, `dbpedia-owl:City`, `extractiv:CITY`, `opencalais:City`, `evri:City` while being more specific than `wikimeta:LOC` and `zemanta:location`.

```
nerd:City a rdfs:Class ;
  rdfs:subClassOf wikimeta:LOC ;
  rdfs:subClassOf zemanta:location ;
  owl:equivalentClass alchemy:City ;
  owl:equivalentClass dbpedia-owl:City ;
  owl:equivalentClass evri:City ;
  owl:equivalentClass extractiv:CITY ;
  owl:equivalentClass opencalais:City .
```

4.1.4 The NERD User Interface

The user interface[15] is developed in HTML/Javascript. Its goal is to provide a portal where researchers can find information about the NERD project, the NERD ontology, and common statistics of the supported extractors. Moreover, it provides a personalized space where a user can create a developer or a simple user account. For the former account type, a developer can navigate through a dashboard, see his profile details, browse some personal usage statistics and get a programmatic access to the NERD API via a NERD key. The simple user account enables to annotate any web documents via its URI. The raw text is first extracted from the web source and a user can select a particular extractor. After the extraction step, the user can judge the correctness of each field of the tuple *(NE, type, URI, relevant)*. This is an important process which gives to NERD human feedbacks with the main purpose of evaluating the quality of the extraction results collected by those tools [82]. At the end of the evaluation, the user sends the results, through asynchronous calls, to the REST API engine in order to store them. This set of evaluations is further used to compute statistics about precision measures for each tool, with the goal to highlight strengths and weaknesses and to compare them [83]. The comparison aggregates all the evaluations performed and, finally, the user is free to select one or

[15] http://nerd.eurecom.fr

more evaluations to see the metrics that are computed for each service in real time.

4.2 Factual Comparison of Named Entity extractors

The NE recognition and disambiguation tools vary in terms of response granularity and technology used. As granularity, we define the way how the extraction algorithm works: One Entity per Name (OEN), where the algorithm tokenizes the document in a list of exclusive sentences, recognizing the dot as a terminator character, and for each sentence, detects named entities; and One Entity per Document (OED), where the algorithm considers the bag of words from the entire document and then detects named entities. The result set differs from the approaches used in terms of duplicates for the same output record *(NE, type, URI)*. In this paper, we take into account the OED approach, removing duplicates in the case that the service provides them.

Figure 4.2 provides an extensive comparison that take into account the technology used: algorithms used to extract NE, supported languages, ontology used to classify the NE, dataset for looking up the real world entities and all the technical issues related to the online computation such as the maximum content request size and the response format. We also report whether a tool provides the position where an NE is found in the text or not. We distinguish four cases: *char offset* considering the text as a sequence of characters, it reports the char index where the NE starts and the length (number of chars) of the NE; *range of chars* considering the text as a sequence of characters, it reports the start index and the end index where the NE appears; *word offset* the text is tokenized considering any punctuation, it reports the word number after the NE is located (this counting does not take into account the punctuation); *POS offset* the text is tokenized considering any punctuation, it reports the number of part-of-a-speech after the NE is located.

We performed an experimental evaluation to estimate the max content chunk supported by each API, creating a simple application that is able to send to each extractor a text of 1KB initially. In case that the answer was correct (HTTP status 20x), we performed one more test increasing of 1 KB the content chunk. We iterated this operation until we received the answer "text too long". Figure 4.2 summarizes the factual comparison of the services involved in this study. The * means the value has been estimated experimentally (as the content chunk), + means a list of other sources, generally identifiable as any source available within the Web, finally *N/A* means not available.

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	Lupedia	OpenCalais	Saplo	Wikimeta	YCA	Zemanta
Granularity	OED	OEN	OED	OEN	OEN	OED	OED	OEN	OEN	OED
Language support	English French German Italian Portuguese Russian Spanish Swedish	English German (partial) Portuguese (partial) Spanish (partial)	English Italian	English	English French Italian	English French Spanish	English Swedish	English French	English	English
Restriction on academic use (calls/day)	30000	unlimited	3000	1000	unlimited	50000	1333	unlimited	5000	10000
Sample clients	C/C++ C# Java Perl PHP-5 Python Ruby	Java Javascript PHP	Action Script Java PHP	Java	N/A	Java	Java Javascript PHP Python	Java Perl	Javascript PHP	C# Java Javascript Perl PHP Python Ruby
API interface	CLI JAX-RS SOAP	AJAX CLI JAX-RS SOAP	AJAX JAX-RS	AJAX CLI JAX-RS	CLI JAX-RS	AJAX CLI JAX-RS SOAP	AJAX CLI JAX-RS	CLI JAX-RS	JAX-RS CLI	AJAX CLI JAX-RS
Content chunk	150KB*	452KB*	8KB*	32KB*	20KB*	8KB*	26KB*	80KB*	7769KB*	970KB*
Response format	JSON Microformats XML RDF	HTML+uF(rel-tag) JSON RDF XHTML+RDFa XML	GPB HTML JSON RDF	HTML JSON RDF XML	HTML JSON RDFa XML	JSON Microformats N3 Simple Format	JSON	JSON XML	JSON XML	XML JSON WNJSON RDF XML
Entity type number	324	320	300*	34	319	95	5	7	13	81
Entity position	N/A	char offset	N/A	word offset	range of chars	char offset	N/A	POS offset	range of chars	N/A
Classification ontologies	Alchemy	DBpedia FreeBase Schema.org	Evri	DBpedia	DBpedia	OpenCalais	Saplo	ESTER (partial)	Yahoo	FreeBase
Deferencable vocabularies	DBpedia Freebase US Census GeoNames UMBEL OpenCyc YAGO MusicBrainz CIA Factbook CrunchBase	DBpedia	Evri	DBpedia	DBpedia LinkedMDB	OpenCalais	N/A	DBpedia Geonames CIAFactbook Wikicompanies others+	Wikipedia	Wikipedia IMDB MusicBrainz Amazon YouTube TechCrunch MusicBrainz Twitter MyBlogLog Facebook others+

Figure 4.2. Factual information about 10 extractors under analysis.

4.3 Experiment preliminaries

We conduct two different experiments: *i)* a qualitative experiment composed of two different human evaluation campaigns. In each, human assessors had rate the output pattern given by NERD for the same set of English documents, with the goal to create new benchmarks and to compare them with the benchmark proposed in [64]. Each participant received first a training session consisting in explaining the various functionalities of the tools and the purpose of evaluating the accuracy of the NE recognition, typing and disambiguation. Moreover, during the assessment step, they were encouraged to look at the NERD help page[16]. *ii)* a quantitative experiment, where we extracted NEs from three different datasets: a dataset composed of transcripts of five *TED* talks, a dataset composed of 1000 news articles from *The New York Times* and a dataset composed of the 217 abstracts of the papers published at *WWW 2011* conference. The aim of this experiment is to focus on how these extractors perform in different scenarios.

4.4 Qualitative Experiments

Each experiment produced a benchmark, respectively: *WEKEX2011* and *ISWC2011*. Both of them are publicly available. In this section we focus on the agreement score and on the precision of the extraction tasks.

4.4.1 WEKEX2011 benchmark

This experiment consisted of asking 4 participants to evaluate the output pattern given by the analysis of the same 10 news articles, each article being rated 5 times (1 for each NE extractor), yielding a total number of analysis of 200. News articles have been selected from the following categories: world, business, sport, science, health from two sources: BBC and The New York Times. The average word number per article is 981.

Some of the extractors (e.g. DBpedia Spotlight and Extractiv) provide NE duplicates because they compute the NE extraction task for each statement of the text. Instead, the others run the extraction task on the whole text, removing intrinsically the duplicates. In order not to bias the statistics, we first removed all duplicates. The final number of unique entities detected was 4641 with an average number of entity per article equal to 23.2.

[16] http://nerd.eurecom.fr/help

Agreement

We compute the Fleiss's Kappa score [33] in order to assess the agreement among the four raters. We interpret this score using the normalization proposed by Sim *et al.*'s classification [86]. Table 4.1 shows the average agreement for each extractor used in the experiment according to all analysis. Low agreement level is obtained for

	NE	Type	URI	relevant
AlchemyAPI	**0.2788**	0.2934	0.8106	0.2013
DBpedia Spotlight	0.0166	0.5252	0.2007	0.1290
Extractiv	0.0200	**0.5396**	0.8036	**0.2027**
OpenCalais	0.0689	0.3927	**1.0**	0.1170
Zemanta	0.0538	0.1013	0.3124	0.0906

Table 4.1. The average agreement for the Fleiss's kappa score computed for each extractor and per involved fields $(NE, Type, URI, relevant)$.

the NE detection and its relevance for all extractors. Instead, an overall agreement is reached for AlchemyAPI, Extractiv and OpenCalais when users evaluated the Type and URI field. DBpedia Spotlight presents substantial agreement among all raters for the type field, instead low agreement for other fields due, essentially, to the heterogeneous results provided by the extractor (i.e. entity list includes named entities and often topic concepts affecting the overall evaluation). Instead, Zemanta shows an interesting agreement when URI field is evaluated. Table 4.2 details the agreement score grouped by the source.

		AlchemyAPI	Extractiv	OpenCalais	Spotlight	Zemanta
BBC	NE	**0.45**	0.01	0.03	-0.04	0.09
	Type	0.39	0.47	0.25	**0.55**	0.13
	URI	0.85	0.84	**1**	0.18	0.28
	rel	**0.24**	0.19	0.09	0.13	0.16
NYTimes	NE	**0.11**	0.03	**0.11**	0.08	0.02
	Type	0.19	**0.61**	0.54	0.5	0.08
	URI	0.77	0.77	**1**	0.22	0.34
	rel	0.16	**0.21**	0.14	0.13	0.02

Table 4.2. Fleiss's Kappa score computed for each extractor and per involved fields (NE,Type,URI,relevant) grouped by source.

Table 4.3 presents the average agreement according to the categories involved in the experiment. Scores are similar for all categories, showing how this experiment reached a good level of agreement for the type and URI evaluation task.

		AlchemyAPI	Extractiv	OpenCalais	Spotlight	Zemanta
business	NE	**0.01**	-0.05	-0.08	-0.04	-0.14
	Type	0.61	0.53	0.28	**0.78**	-0.03
	URI	0.73	**0.83**	1	0.1	0.15
	rel	0.08	**0.14**	0.07	0	-0.07
health	NE	**0.59**	0.07	0.08	-0.06	0.03
	Type	-0.05	**0.78**	0.27	0.25	0.2
	URI	0.8	0.92	**1**	0.28	0.07
	rel	**0.59**	0.16	0.16	-0.05	0.14
science	NE	0.0	-0.1	-0.01	-0.08	**0.14**
	Type	0.03	0.44	0.01	**0.48**	0.47
	URI	0.95	0.74	**1**	0.02	**1**
	rel	-0.08	0.03	**0.09**	-0.03	-0.09
sport	NE	**0.57**	0.1	0.32	0.37	0.49
	Type	0.5	0.37	**0.64**	0.38	-0.31
	URI	0.84	0.77	**1**	0.5	0.16
	rel	0.06	0.33	0.24	**0.61**	0.47
world	NE	**0.22**	0.08	0.04	-0.11	0.04
	Type	0.38	0.58	**0.76**	0.74	0.18
	URI	0.74	0.76	**1**	0.09	0.19
	rel	**0.35**	**0.35**	0.02	0.11	0

Table 4.3. Fleiss's Kappa score computed for each extractor and per involved fields (NE,Type,URI,relevant) grouped by article category.

Precision

The precision value, p, is computed with the average of the precision for each field of the output triple $o = (NE, type, URI)$. The relevant score, is computed considering the user rating of each pair $(NE, type)$. According to Table 4.4, AlchemyAPI has the best overall performances both in terms of precision and relevant score.

	p_T	*relevant score*
AlchemyAPI	**0.7054**	**0.9005**
DBpedia Spotlight	0.4915	0.5525
Extractiv	0.611	0.6805
OpenCalais	0.5396	0.8224
Zemanta	0.6463	0.8800

Table 4.4. Aggregate result comparisons considering the average of the precision and recall for all submitted runs in the controlled experiment.

In Table 4.5, we focus on the detailed precision value of each output o_i. The results are a bit more contrasted. AlchemyAPI, although preserving good performance in NE extraction and accurate typing, has a clear weakness to link the NE to a web resource. URI disambiguation is better performed by Zemanta and DBpedia Spotlight. Moreover, Zemanta has a good reliability to recognize NE in contrast to DBpedia Spotlight. However, both lack the rich type classification. For what concerns DBpedia Spotlight, this result contrasts with the large ontology used to classify the extracted NEs. OpenCalais and Extractiv demonstrate good results in the type identification task.

	p_{NE}	p_{type}	p_{URI}
AlchemyAPI	**0.9440**	**0.8938**	0.2783
DBPedia Spotlight	0.5995	0.0922	0.7828
Extractiv	0.7713	0.6768	0.3849
OpenCalais	0.8687	0.75	0.0
Zemanta	0.9031	0.1403	**0.8954**

Table 4.5. Precision results of NE extraction, type classification and URI selection on all NE extractors evaluated in the controlled experiment.

Configuration parameters affect the general behaviour of these NE extractors. We investigate whether an extractor shows better results or not for a particular genre of text such as news articles. We group all submitted runs according to both authorities: BBC and the New York Times. The results shown that AlchemyAPI has, again, the best performances in terms of NE extraction and type classification (Table 4.6), but its contribution to select URIs is very low. Previous results showed a good performance of DBpedia in URI disambiguation, but this result is affected by drops with The New York Times articles while showing slight increase for the BBC news articles. Zemanta keeps good performance of URI disambiguation for both authorities.

	BBC			NY Times		
	p_{NE}	p_{type}	p_{URI}	p_{NE}	p_{type}	p_{URI}
AlchemyAPI	**0.9676**	**0.9380**	0.4013	**0.9235**	**0.8702**	0.2069
Dbpedia Spotlight	0.5955	0.1252	0.7677	0.6197	0.0635	0.0635
Extractiv	0.7674	0.7169	0.3633	0.7891	0.6520	0.3882
OpenCalais	0.9056	0.8755	0.0	0.8340	0.6851	0.0
Zemanta	0.9075	0.1413	**0.8938**	0.8950	0.1450	**0.8950**

Table 4.6. Comparison of NE extraction, type and URI precision among all NE extractors according to the source authority in the controlled experiment.

4.4.2 ISWC2011 benchmark

This experiment consists in asking 10 participants to evaluate the output pattern given by the analysis of different parts of two news articles, each article being rated 7 times (1 for each NE extractor), yielding a total number of analysis of 140. News articles have been selected from the dataset published in [64]. The average word number per article is 190. Some of these extractors, such as DBpedia Spotlight, and Extractiv provide NE duplicates because they compute the NE extraction task according to the OEN approach. Using the NERD framework, we obtained a list of NEs without duplicates. In this way, the extraction results are not affected by evaluation ambiguities. The final number of unique entities detected is 1780 with an average number of entity per article equal to 12.71.

The NERD framework provides a unified extraction output composed of named entity, type, URI and a relevance field (which subjectively indicates if the triple NE, type and URI is "important" for the text). Each human rater assessed the output evaluating these criteria with a Boolean value: true if the detected entity was indeed present in the article; true if the assigned type of the named entity is correct in the context of the article; true if the URI provided is an accurate disambiguation of the named entity detected. Furthermore, the users were asked to judge subjectively the relevance score. In the case where the participant did not assess a result, it was considered false. In the case where no type or no disambiguation URI was provided by the tool, it was counted as false. Finally, some of these extractors provide configuration parameters, such as confidence score or support. To unify all behaviors, we used those tools using their default configurations.

Agreement

In order to assess the agreement among the 10 human raters, we compute the Fleiss's kappa score [33]. Table 4.7 shows the average agreement for each extractor used in the experiment according to all analysis. Results show an overall low agreement about the NE evaluation with respect to the other fields, mainly due to the different interpretations of the NE definition and the low precision of NE detection by the extractors. In this scenario, OpenCalais is an exception and has obtained a considerable agreement, mainly because its NE detection algorithm has a high precision rate (see Table 4.10). It is worth noting that Zemanta has the best agreement score on classification task and OpenCalais the best agreement score for the URI disambiguation. Finally, raters reached a perfect agreement on judging the relevance score for the triple NE, type and URI provided by Evri.

In contrast with the WEKEX2011 [83] benchmark, the average agreement score is increased for each single task of field evaluation. However, the human evaluation process still remains fuzzy especially for the NE evaluation. This is due to the slight

	NE	Type	URI	relevant
AlchemyAPI	0.4853	0.8833	0.7838	0.8677
DBpedia Spotlight	0.2337	0.6011	**0.847**	0.2287
Evri	0.4927	0.8208	0.4866	**1**
Extractiv	0.6168	0.8146	0.7519	0.6108
OpenCalais	**0.8646**	0.4805	0.4222	0.8611
Zemanta	0.7445	**0.9599**	0.2291	0.686

Table 4.7. The average agreement for the Fleiss's kappa score computed for each extractor and per involved fields $(NE, Type, URI, relevant)$ between two news articles.

difference that exists between the strict NE definition and the more general idea of keyword useful to index a text. Table 4.8 shows the agreement about the NE evaluation task on two different evaluation benchmarks (this one named ISWC2011 and the WEKEX2011 one) according to the interpretation of the Fleiss's kappa score using the normalization proposed by Sim *et al.*'s classification [86].

	ISWC2011	WEKEX2011
AlchemyAPI	moderate	slight
DBpedia Spotlight	fair	poor
Evri	moderate	-
Extractiv	substantial	slight
OpenCalais	almost	fair
Zemanta	substantial	slight

Table 4.8. Agreement computed according to the Fleiss's kappa score on the NE evaluation grouped by the evaluation scenario.

Precision

The precision value p_T is computed with the average precision for each field of the output triple *(NE, type, URI)*. According to Table 4.9, Evri has the best overall performances in terms of average precision among all fields involved in the evaluation, while AlchemyAPI has the best performance to find out important extraction triple (the set of NE, type and URI) according to the text evaluated.

In Table 4.10, we focus on the detailed precision value of each field such as p_{NE}, p_{type} and p_{URI}. This result view confirms the overall high level performances of Evri, showing how raters considered precise the NE extraction, typing and disambiguation tasks of this tool. However, disambiguated resources are provided just in

XML format, making complex the evaluation process for a human being[17]. AlchemyAPI results are a bit more contrasted. Although preserving good performances in NE extraction and accurate typing, it has a clear weakness to link the NE to a web resource. OpenCalais shows an interesting result on typing entity and NE detection, but as well as AlchemyAPI, it suffers from a low precision rate for the disambiguation task. Although this seems weird (the entire dataset used by OpenCalais to disambiguate resources is part of the LOD cloud), the motivation behind is simple: OpenCalais often links entities with not-final real word objects but sort of disambiguating pages. This is the case presented during the evaluation of the article http://nyti.ms/dC4KPE when the entity `Los Angeles` is disambiguated with a URI which contains a list of disambiguation URIs[18]. Instead, Zemanta has a good reliability to recognize NE and disambiguated URIs in contrast to DBpedia Spotlight. However, both lack the rich type classification. For what concerns DBpedia Spotlight, these results show that this tool has bad performances when it works using default configuration, i.e. without tuning the configuration parameters. Extractiv has steady performance, without being excellent in none of the evaluation field.

	p_T	*relevant score*
AlchemyAPI	0.7305	**0.933**
DBpedia Spotlight	0.2526	0.3956
Evri	**0.9412**	0.8706
Extractiv	0.6859	0.7061
OpenCalais	0.6905	0.8857
Zemanta	0.6782	0.825

Table 4.9. Aggregate result comparisons considering the average precision and recall for all submitted runs in the controlled experiment.

Although we cannot compare these results with what is proposed in the WEKEX2011 benchmark, since different datasets have been used and more tools are considered in this experiment, we observe that the general trend is similar. The main difference is the interesting performances offered by Evri which outperformed all the extractors evaluated so far. We can however perform a partial comparison with the work proposed in [64] regarding the disambiguation process since we purposely re-used the same dataset. Taking into account the three common extractors (AlchemyAPI,

[17] E.g. http://api.evri.com/concept/artificial-intelligence-0x5007de describes the named entity "Artificial Intelligence".

[18] E.g. http://d.opencalais.com/genericHasher-1/874eaab9-7b66-36e3-9650-8de7a5001cf9.htm shows a list of disambiguation URIs.

	p_{NE}	p_{type}	p_{URI}
AlchemyAPI	0.9833	0.9167	0.2917
DBpedia Spotlight	0.3067	0	0.4511
Evri	**0.9941**	0.9588	**0.8706**
Extractiv	0.8156	0.6606	0.5818
OpenCalais	0.8714	**0.9786**	0.2214
Zemanta	0.82	0.39	0.825

Table 4.10. Precision results of NE extraction, type classification and URI selection on all NE extractors evaluated in the controlled experiment.

DBpedia Spotlight, OpenCalais) the only difference we have is about the value of the p_{URI} of AlchemyAPI. The reason is due to a different approach to evaluate the provided URI list. Indeed, this list may have one or more URIs and usually it may contain wiki links, LOD URIs, web resources (such as http://www.italia.it). Mendes *et al.* explored this list to look up the wiki link. Instead, in this work, we considered the first URI provided. This has affected the final human evaluation task, because a web resource URI is sometimes less descriptive than a wiki link.

4.5 Quantitative Experiment

We performed a quantitative experiment using three different datasets: a dataset composed of transcripts of five *TED* talks (different category of talks), a dataset composed of 1000 news articles from *The New York Times* (collected from 09/10/2011 to 12/10/2011), and a dataset composed of the 217 abstracts of the papers published at *WWW 2011* conference. The aim of these evaluations is to assess how these extractors perform in different scenarios, such as news articles, user generated content and scientific papers. The total number of document is 1222, with an average word number per document equal to 549. Each document was evaluated using 6 extractors supported by the NERD framework[19]. The final number of entities detected is 177,823 and the average of unique entity number per document is 20.03. Table 4.11 shows statistics about grouped view according to the source documents.

We define the following variables: the number n_d of evaluated documents, the number n_w of words, the total number n_e of entities, the total number n_c of categories and n_u URIs. Moreover, we compute the following measures: word detection rate $r(w, d)$, i.e. the number of words per document, entity detection rate $r(e, d)$, i.e.

[19]At the time this evaluation has been conducted Lupedia, Saplo, Wikimeta and YCA were not part of the NERD framework.

the number of entities per document, the number of entities per word $r(e, w)$, the number of categories per entity $r(c, e)$ (this measure has been computed removing not relevant labels such as "null" or "LINKED_OTHER") and the number of URIs per entity $r(u, e)$.

	WWW2011	TED	NYTimes
n_d	217	5	1,000
n_w	38,062	13,381	62,0567
r_w	175.4	2,676.2	620.567
n_e	12,266	1,441	164,116
r_e	56.53	288.2	164.1

Table 4.11. Statistics about the three dataset used in the quantitative experiment, grouped according to the source where documents were collected.

4.5.1 User Generated Content

In this experiment, we focus on the extractions performed by all tools for 5 *TED* talk transcripts. The goal is to find out NE extraction patterns for user generated content, such as speech transcripts of videos. First, we propose general statistics about the extraction task and then, we focus on the classification, showing statistics grouped according to the NERD ontology. DBpedia Spotlight classifies each resource according three different schema (see Figure 4.2). For this experiment, we consider only the results which belong to the DBpedia ontology.

The total number of documents is 5, with an overall number of total words equal to 13,381. The word detection rate per document $r(w, d)$ is equal to 2,676.2 with an overall number of entities equal to 1,441, and the $r(e, d)$ is 288.2. Table 4.12 shows the statistics about the computation results for all extractors. DBpedia Spotlight is the extractor which provides the highest number of NE and disambiguated URIs. These values show the ability from this extractor to locate NE and to exploit the large cloud of LOD resources. In parallel, it is crucial noting that it is not able to classify these resources, although it uses a deep classification schema. All the extractors show high ability for the classification task, except Zemanta as shown by the $r(c, e)$. Contrarily, Zemanta shows strong ability to disambiguate NE via URI identification, as shown by $r(u, e)$. It is worth noting OpenCalais and Evri have almoust the same performances of Zemanta.

The last part of this experiment consists in aligning all the classification types provided by these extractors, while performing the analysis of *TED* talk transcripts, using the NERD ontology. For the sake of brevity, we report all the grouping results according to 6 main concepts: Person, Organization, Country, City, Time and

Number. Table 4.13 shows the comparison results. AlchemyAPI classifies a higher number of Person, Country and City than all the others. In addition, OpenCalais obtains good performances to classify all the concepts except Time and Number. It is worth noting that Extractiv is the only extractor able to locate and classify Number and Time. In this grouped view, we consider all the results classified with the 6 main classes and we do not take into account all potentially inferred relationships. This is why the Evri results contrast with what is showed in the Table 4.12. Indeed, Evri provides a precise classification about Person such as Journalist, Physicist, Technologist but it does not describe the same resource as a sub-classes of the Person axiom.

4.5.2 Scientific Documents

In this experiment, we focus on the extraction performed by all tools for the 217 abstract papers published at the *WWW 2011* conference, with the aim to seek NE extraction patterns for scientific contributions. The total number of words is 13,381, while the word detection rate per document $r(w, d)$ is equal to 175.40 and the total number of recognized entities is 12,266 with the $r(e, d)$ equal to 56.53. Table 4.14 shows the statistics of the computation results for all extractors. DBpedia Spotlight keeps a high rate of NEs extracted but shows some weaknesses to disambiguate NEs with LOD resources. $r(u, e)$ is equal to 0.2871, lesser than is performance in the previous experiment (see section 4.5.1). OpenCalais, instead, has the best $r(u, e)$ and it has a considerable ability to classify NEs. Evri performed in a similar way as shown by the $r(c, e)$.

The last part of this experiment consists in aligning all the classification types retrieved by these extractors using the NERD ontology, aligning 6 main concepts: Person, Organization, Country, City, Time and Number. Table 4.15 shows the comparison results. AlchemyAPI still preserves the best result to classify named entities as Person. Instead, differently to what happened in the previous experiment, Evri outperforms AlchemyAPI while classifying named entities as Organization. It is important to note that Evri shows an high number of NEs classified using the class Person in this scenario, but does not explore deeply the Person inference (as shown in the user generated content experiment). OpenCalais has the best performance to classify NEs according to the City class, while Extractiv shows reliability to recognize Country and, especially, to classify Number.

4.5.3 News Articles

For this experiment, we collected 1000 news articles of *The New York Times* from 09/10/2011 to 12/10/2011 and we performed the extraction for the tools involved in this comparison. The goal is to explore the NE extraction patterns with this

dataset and to assess commonalities and differences with the previous experiments. The total number of words is 620,567, while the word detection rate per document $r(w, d)$ is equal to 620.57 and the total number of recognized entities is 164,12 with the $r(e, d)$ equal to 164.17. Table 4.16 shows the statistics of the computation results for all extractors.

Extractiv is the tool which provides the highest number of NEs. This score is considerably greater than what does the same extractor in the other test scenarios (see section 4.5.1 and section 4.5.2), and it does not depend from the number of words per document, as reported by $r(e, w)$. In contrast, DBpedia Spotlight shows a $r(e, w)$ which is strongly affected by the number of words: indeed, the $r(e, w)$ is 0.048 lower than the same score in the previous experiment. Although the highest number of URIs detailed is provided by OpenCalais, the URI detection rate per entity is greater for Zemanta, with a score equal to 0.577. Alchemy, Evri, and OpenCalais confirm their reliability to classify NEs and its detection score value $r(c, e)$ is sensibly greater than all the others.

Finally, we propose the alignment of the 6 main types recognized by all extractors using the NERD ontology. Table 4.17 shows the comparison results. Differently to what has been detailed previously, DBpedia Spotlight recognizes few classes, although this number is not comparable with what performed by the other extractors. Zemanta and DBpedia Spotlight increase classification performances with respect to the experiments detailed in the two previous test cases, obtaining a number of recognized Person which is lower than one magnitude order. AlchemyAPI preserves strong ability to recognize Person, but still shows great performance to recognize City and significant scores for Organization and Country. OpenCalais shows meaningful results to recognize the class Person and especially a strong ability to classify NEs with the label Organization. Extractiv holds the best score for classifying Country and it is the only extractor able to seek the classes Time and Number.

4.6 Discussion

The experiment results proposed in this work highlight strengths and weaknesses of the extractors under analysis for particular tasks, scenario and settings. The qualitative experiment elects Evri as the tool with the best average precision when it extracts named entities from news articles, while AlchemyAPI preserves an important relevance score. This metric shows the reliability of an extractor to locate NEs which are important (relevant) for the text. This result follows what we proposed in our previous work and, still elects AlchemyAPI as one of the best NE extractor tool. Although the precision score of each single field is affected by the precision of extraction for the NE (type and URI are assigned as a consequence of the NE extraction), it shows the ability to locate a proper value in the classification schema and

a meaningful resource which may describe the extracted NE. These metrics confirm the average result, electing Evri as most precise to recognize and to disambiguate a named entity. At the same time, it elects OpenCalais as the most precise tool for classifying resources. OpenCalais, exploiting a reduced classification schema, is better than other tools to disambiguate the NEs extracted. When using the complete DBpedia ontology, DBpedia Spotlight may potentially give a much more precise evaluation of the entity type. Up to now, indeed, when the type is associated, DBpedia Spotlight gives a very deep class hierarchy which helps a computer machinery to better structure the text. The type generation still remains a tricky point for most of them especially for DBpedia Spotlight, Zemanta!. It is worth noting that Evri, OpenCalais and DBpedia Spotlight provide a disambiguation mechanism which links each information unit with web resources in its authority domain and provides occasionally same as links to other LOD dataset. AlchemyAPI, Zemanta, Extractiv are able to exploit the LOD cloud and at the same time may locate real world objects within the Web.

The quantitative experiment proposes an evaluation of these extractors in different scenarios, such as news articles, user generated content and scientific papers with the goal to seek common extraction patterns. In all of these three scenarios, DBpedia Spotlight shows an almost steady entity detection rate per word from 0.097 to 0.048. According to these results, we may state DBpedia Spotlight is not affected by the number of words. Also, OpenCalais shows a word independence NE extraction rate, ranging from 0.012 and 0.038. All the others vary their named entity extraction rate per document according to the number of words. Despite a large number of NE detection, DBpedia Spotlight lacks precision, highlighting how the algorithm suffers when it works using the default configuration. OpenCalais confirms good result in terms of URI detection rate per entity, but, according to what we detailed above, its disambiguation algorithm is not often able to locate final real world object, but instead uses resources that are actually a list of possible disambiguation URIs. Evri, instead, guarantees a good type detection rate per entity and URI detection rate per entity. However, all the URIs detected point to resources described only in XML. Moreover, an important weakness is represented by the classification schema used. Despite these shortcomings, it seems very accurate, although the specifications are not yet released and make tough the alignment process to the NERD ontology. All these extractors do not show particular differences when they work with different kind of text, expect for Extractiv, which is definitely the only one able to locate Number and Time references. Zemanta shows an important URI detection rate per named entity in all the test cases.

	n_e	n_c	n_u	$r(e,d)$	$r(e,w)$	$r(c,e)$	$r(u,e)$
AlchemyAPI	141	**141**	71	28.2	0.01	**1**	0.504
DBpedia Spotlight	**810**	0	**624**	**162**	**0.06**	0	0.77
Evri	120	120	113	24	0.009	**1**	0.942
Extractiv	60	53	22	12	0.004	0.883	0.367
OpenCalais	163	136	157	32.6	0.012	0.834	0.963
Zemanta	50	17	49	10	0.003	0.34	**0.98**

Table 4.12. Statistics about computation results for the sources coming from *TED* talks of all extractors used in the comparison.

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	OpenCalais	Zemanta
Person	42	-	10	6	27	4
Organization	15	-	-	-	20	1
Country	16	-	11	1	16	3
City	14	-	3	3	7	-
Time	-	-	-	1	-	-
Number	-	-	-	5	-	-

Table 4.13. Number of axioms aligned for all the extractors involved in the comparison according to the NERD ontology for the sources coming from *TED* talks.

	n_e	n_c	n_u	$r(e,d)$	$r(e,w)$	$r(c,e)$	$r(u,e)$
AlchemyAPI	323	171	39	1.488	0.008	0.529	0.121
DBpedia Spotlight	**3,699**	0	**1,062**	**17.04**	**0.097**	0	0.287
Evri	282	282	167	1.299	0.007	**1**	0.592
Extractiv	1,345	725	415	6.198	0.035	0.539	0.309
OpenCalais	1,158	**1,158**	713	5.337	0.03	**1**	**0.616**
Zemanta	1,748	97	757	8.055	0.046	0.055	0.433

Table 4.14. Statistics about extraction results for the 217 abstract papers published at the *WWW 2011* conference of all extractors used in the comparison.

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	OpenCalais	Zemanta
Person	17	-	12	6	6	1
Organization	20	-	24	-	5	-
Country	9	-	8	14	7	6
City	4	-	3	8	9	-
Time	-	-	-	-	-	-
Number	-	-	-	184	-	-

Table 4.15. Number of axioms aligned for all the extractors involved in the comparison according to the NERD ontology for the sources coming from the *WWW 2011* conference.

	n_e	n_c	n_u	$r(e,d)$	$r(e,w)$	$r(c,e)$	$r(u,e)$
AlchemyAPI	17,433	17,443	3,833	17.44	0.028	**1**	0.22
DBpedia Spotlight	30,333	20	8,702	30.33	0.048	0.001	0.287
Evri	16,944	16,944	8,594	16.94	0.027	**1**	0.507
Extractiv	**47,455**	**41,393**	8,177	**47.45**	**0.076**	0.872	0.172
OpenCalais	23,625	23,625	**12,525**	23.62	0.038	**1**	0.53
Zemanta	9,474	4621	5,467	9.474	0.015	0.488	**0.577**

Table 4.16. Statistics about extraction results for the 1000 news articles published by *The New York Times* from 09/10/2011 to 12/10/2011 of all extractors used in the comparison.

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	OpenCalais	Zemanta
Person	6,246	14	2,698	5,648	5,615	1,069
Organization	2,479	-	900	81	2,538	180
Country	1,727	2	1,382	2,676	1,707	720
City	2,133	-	845	2,046	1,863	-
Time	-	-	-	123	1	-
Number	-	-	-	3,940	-	-

Table 4.17. Number of axioms aligned for all the extractors involved in the comparison according to the NERD ontology for the sources collected from the *The New York Times* from 09/10/2011 to 12/10/2011.

Chapter 5

Use cases of Linked Data applications for managing educational data

Key to realize the Linked Data vision is to structure textual documents by means of NLP techniques. Such a techniques have been proposed by the NLP community and used by the Linked Data community not only for extracting named entities but, especially, to classify them according to fine grained ontologies and to disambiguate them though URIs which point to real word objects. The Web becomes an enormous look-up space where computing machinery find additional and meaningful information. Well-formed repositories, constantly upgraded, play an increasingly important role in enhancing data aggregation from public archives, enabling machines to enrich the content with external data, just following the URIs extracted. Data becomes the new oil. Existing structured datasets require to be lifted in the new LOD cloud. Moreover, the role of social networks has changed the way how group of people can agree on topics and discuss about social things. Leveraging on the idea of the Web as knowledge space where data repositories are sources which can enrich the content of documents or comments, in this chapter we propose three different Linked Data applications. Basically, our process uses the Linked Data idea and, therefore, addresses the exploitation of the Web as a platform for data integration. First we propose a Linked Data approach to augment the automatic classification task in a Systematic Literature Review, then we go further proposing an application to lift educational data stored in SCORM data silos to the Web of Data and, finally, we propose a scientific conference venue enhancer plug on the top of several data live

collectors, such as Twitter[1], G+[2], Flickr[3], SlideShare[4].

5.1 Improving SLR preselection process through Linked Data enrichment

A SLR (Systematic Review) is a mean to identify, evaluate and interpret all available interesting research to a particular topic area or phenomenon of interest [52]. SLR has to be performed according to a pre-defined protocol describing how primary sources are selected and categorized, reducing as much as possible subjectivity bias. It is composed by five steps: (i) identification of research, (ii) selection of studies, (iii) study quality assessment, (iv) data extraction and monitoring progress, (v) data synthesis. The first step defines the search space, i.e. the set inside which researchers may select papers. Then, every research document fallen out is not treated in the selection process. The second step represents an attempt to identify and analyze all possible useful studies to answer the research questions among the papers which are contained in the search space. The selection workload depends on to the dimension of the search space, consequently a large one determines a great deal of work to be done manually. Moreover, being an operation performed manually, the human opinion might influence the outcome. Our approach focuses on a filter strategy resorting on Semantic Web and text mining techniques to reduce the number of papers that researchers performing a SLR should read. We use a text classifier to filter potentially interesting documents within the search space. The classifier produces a reduced set that shall contain a higher interesting document percentage than the initial set. Afterwards, that reduced set is examined manually by researchers [91]. In this way, we reduce the workload required to all researchers, limiting human error rate. This phenomenon usually occurs when a set is sparse and searching on that may require more fatigue than in a clean set, where the noise is smaller.

5.1.1 Study Selection Process

The first step in the approach presented by [52] is the identification of research. The aim of this phase is to identify a subset of articles, W (the working area gathered from the universe of all scientific papers), in the domain of interest applying the

[1] http://twitter.com

[2] https://plus.google.com

[3] http://www.flickr.com

[4] http://www.slideshare.net

defined search strategy. For instance, W could be composed by all papers published by a given set of journals or by all papers that a digital library provided as result of the search with keywords. The following step is the selection process which operates on W to obtain the primary sources to consider in the review. This process is performed by researchers and it is divided in two sub-steps: the former operates a selection based on reading titles and abstracts (*first selection*), the latter is the decision based on the full text human analysis (*second selection*). We define C (*candidate studies*) the set of studies that successfully passed the first selection and are eligible to be processed by researchers in the second one. This second sub-step, in fact, has the goal to split C in I (*included studies*) and E (*excluded studies*) where those sets are:

- I is the set of studies $\in C$ that successfully passed the second manual selection and will contribute to the systematic review. The following relation holds: $I \subseteq C$.
- E is the set of studies $\in C$ that didn't pass the second manual selection and will *not* contribute to the systematic review and synthesis. Hence, $E \subseteq C$ and $E \cap I = \oslash$.

In Figure 5.1 is summarized the whole process.

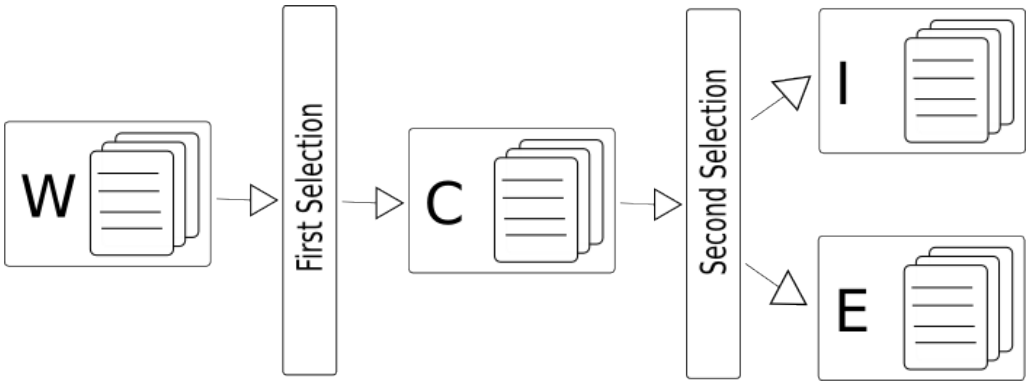

Figure 5.1. Study selection process in systematic reviews (according to the Kitchenham guidelines) represented through sets, selections and their relationships.

5.1.2 Enriched Study Selection Process

Our idea is to create W' set of most interesting papers, according to the domain of interest, gathered from a large set of unread papers W. To obtain it, we use the existing technologies in the field of Semantic Web and text mining techniques, in the context of the Linked Data approach. The process we describe here is a supervised

iterative process built on the top of the following assumption: $W \neq \oslash$ (as a result of the applied search strategy) and $I \neq \oslash$ at the begin (some relevant papers are already known when the systematic review starts).

I_0 construction

The initial set of sources contained in I is named I_0 and it is composed of primary sources already classified as relevant for the systematic review: this is the first step of our process and it is needed to start the iterative part of the algorithm. I_0 can be built in two different ways. The first way is to ask researchers to use their previous knowledge indicating the most well known and fundamental papers in the field of interest. This strategy considers that often systematic reviews are undertaken by researchers experts in the field. The second way is to explore a portion of the search space using the basic process, e.g. searching on digital libraries or selecting the issues of (a) given journal(s). This portion is marked as I_0 and the enriched process is used to explore the remaining search space.

Model building

The second step of our approach consists in computing automatically a model M from the I_0, pool of interesting papers chosen to initialize the model. The idea is to build a bag of words model starting from the primary studies in I_0. The bag of words model is a representation of the text as an unordered and weighted collection of words, holding their combined appearance frequency, disregarding grammar and word order. For each primary study, we will consider only abstract and introduction. According to [26] terms that appear at the start or at the end of the document are more significant. We excluded the title and the conclusions using just the abstract and introduction. The rationale is to validate our approach also for situations when less information is available. Finally, we perform stop words elimination and stemming process, using the Porter algorithm [74]. The model so built is used to train the Naive Bayes classifier, which will compute the weight for each words according to the TF-IDF normalized approach [50].

Linked Data enrichment of papers

As described above, the main actor of this process is DBpedia, a Resource Description Framework (RDF) repository where information stored in Wikipedia is represented as structured data. We define w_i a paper $\in W$: each w_i is processed to get a set of named entities N which summarizes w_i. Named entities are basically information units univocally defined, those units are normally described by a set of properties. Formally, a named entity is a phrase that contains name of people,

organizations, locations, times and quantities and it refers to exactly one or multiple identical, real or abstract text concept instances as introduced by [69]. This operation is done using a NLP extractors, which calculates contextualized entities using NLP algorithm and results are disambiguated using the Web of Data [94]. After that, we link each $n_i \in N$ to the correspondent DBpedia resource (when it is available). Then, from this resource we collect all words contained in the description field (abstract property) and the we add those text information to the bag of words natively taken by the paper w_i. We call it enriching process and the resulting paper is named $w+_i$. Finally, it is compared with the trained model, M, by means of the Naive Bayes classifier, which is described below.

The use of NLP in the SLR is not novelty, but the way in which we perform this approach is. Indeed, [25] exploited NLP in order to automatically index relevant studies in a SLR in order to prioritize the work to analyze the papers. In our approach, instead, we use it to overcome the limitation introduced by the analyzing of a reduced set of words linking other related information already existing in the LOD cloud.

Naive Bayes model classification

We used a Multinomial Naive Bayes (MNB) classifier [77] and we implement the TF-IDF weight normalization. According to [50] this implementation outperforms the CNB used by [63]. So, we use the classifier to compare $w+_i$ with the model learnt and we determine whether the conditional probability that $w+_i$ belongs to I (from which M derives) is significant. We assume that all papers that do not belong to I, belong to E adopting the Boolean algebra. The comparison is done for each $w+_i \in W$: papers with $P[w+_i \in I] \geq threshold$ are moved to W' and they will be manually analyzed by researchers. Contrariwise, all papers whose $P[w+_i \in I] < threshold$ remain in W.

Iteration

As described in the previous section, papers with $P[w+_i \in I] \geq threshold$ are moved to W' to be manually processed, whilst the remaining ones still stay in W. Likely some of the papers moved in W' will pass the following manual selections and will go to I, while the others will go to E. Whether I is modified, M becomes obsolete then it is necessary to re-build it and repeat the classification step for all papers $w+_i \in W$. Again, if $P[w+_i \in I] \geq threshold$, $w+_i$ is moved to W' to be manually analyzed. If any $w+_i$ goes to W', i.e. $W' = \oslash$ after a classification, iteration stops. Papers that remain in W after the last iteration are finally discarded and not considered by researchers, the exclusion of those papers represent the reduction in workload for the human researchers. As a result of the iteration process, at each iteration

Algorithm 1 Enriched selection process algorithm

```
Define I_0
Init I with I_0
repeat
    Train classifier with I
    Extract model M
    for all w_i in W do
        Enrich w_i obtaining w+_i
        Compare w+_i with model M:
        if P[w+_i in I] ≥ threshold then
            move w_i to W'
        end if
    end for
    for all w'_i ∈ W' do
        Manually read title and abstract (w'_i ∈ I ) ? move w'_i to C : discard w'_i
    end for
    for all c_i ∈ C do
        Manually read full paper (c_i ∈ I ) ? move c_i to I : move c_i to E
    end for
until C ≠ ∅
Discard ∀ w_i ∈ W
```

the model will be progressively tailored to the domain of interest, permitting to refine the selection process. We provide in Algorithm 1 the synopsis of the whole study selection process proposed in this paper and in Figure 5.2 its complementary graphical representation. Comparing this picture with Figure 5.1, that represents the selection process provided by guidelines [52], we observe that the original process is not changed, but we add a selection of primary studies that recommends papers similar to the model at each iteration. We also reported in Figure 5.2 the steps of the new process described in this section: the use of a model of bag of words (b) derived from I_0 or I (a), the enrichment of papers through Linked Data (c) and the comparison with the model M by means of the Naive Bayes classifier (d). For the sake of simplicity, we do not represent the iteration.

5.1.3 Goal definition

We define the goal of the case study following the GQM template [5]. The goal structure is summarized in Table 1.

5.1.4 Research questions

We derive from the goal different research questions. Firstly, we are interested in investigating whether our selection process allows reducing the manual work needed for the selection of sources while preserving completeness, and if yes, in quantifying the work saved (RQ1). Subsequently, we assess the contribution of the Linked Data mechanism. We compare results achieved with the Naive Bayes classifier in the case of analysis of enriched papers (obtained with the Linked Data enrichment) and in

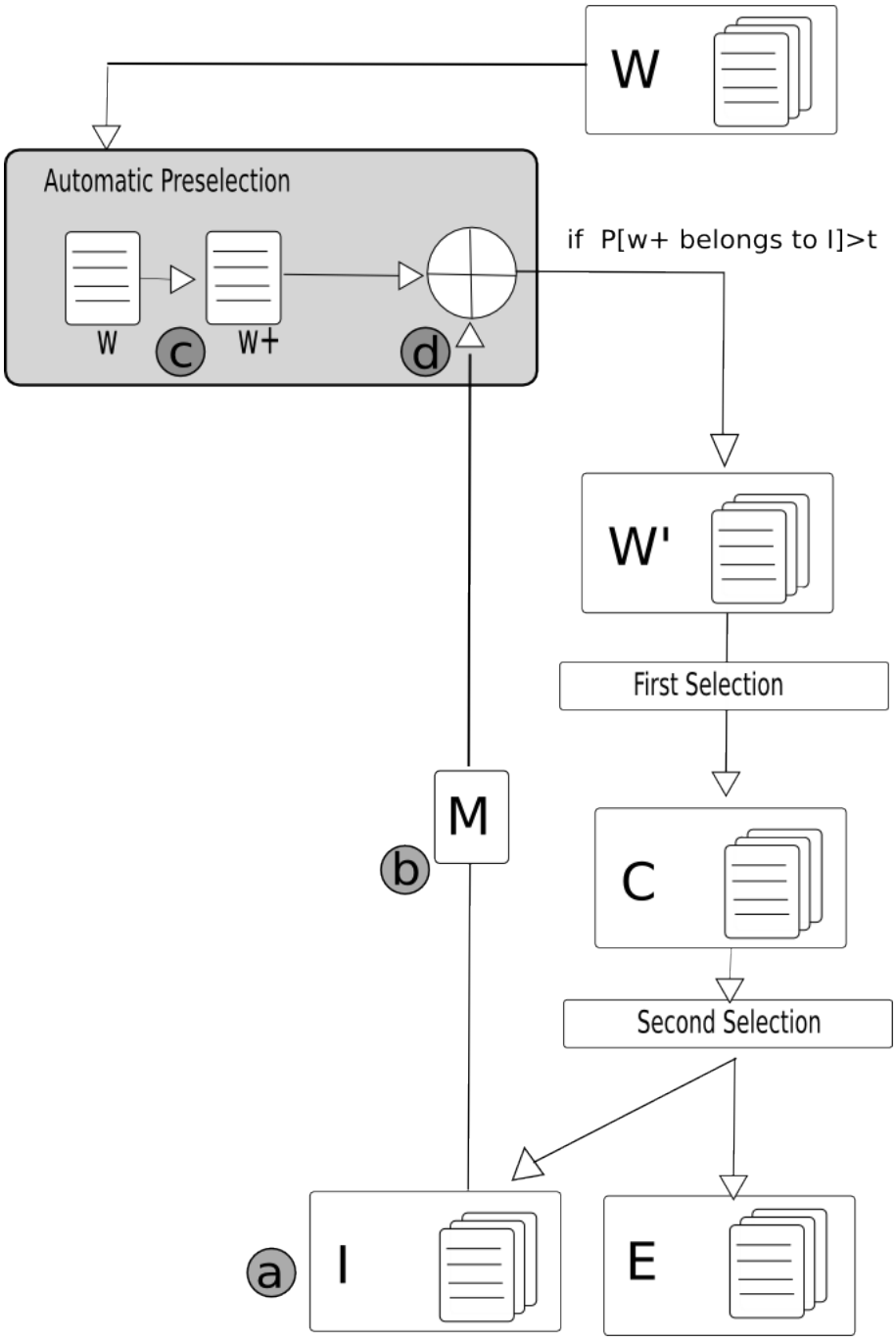

Figure 5.2. The enriched study selection process and its principal steps: model extraction (b) after I is built (a), enrichment of papers through Linked Data (c) and comparison with the model through the Naive Bayes classifier (d).

Evaluate	a primary studies selection mechanism based on the Naive Bayes classifier and the Linked Data enrichment,
for the purpose	of evaluation,
with respect to	the reduction of the amount of manual work without loosing relevant primary studies,
from the viewpoint	of researchers that undertake SLRs,
in the context of	universities and content providers.

Table 5.1. Case Study Goal Definition

the case of examination of original papers (not-enriched ones) (RQ2). Summarizing, we obtain two research questions:

1. Does the automatic selection process based on the Naive Bayes classifier and Linked Data enrichment (enriched process) reduce the amount of manual work of a SLR with respect the original process defined by guidelines?

2. Does the automatic selection process based on Naive Bayes classifier and Linked Data enrichment (enriched process) reduce the amount of manual work of the alternative version of the process with only Naive Bayes classifier (not-enriched process)?

With RQ2 we aim to validate the idea behind the use of enriched papers as test samples instead to use original papers as test samples.

5.1.5 Subject selection

We select as a reference a Systematic Review on Software Cost Estimation done by [48]. The authors firstly selected a list of interested journals, then they examined the title and the abstract of all papers appeared in the issues of these journals in order to select which papers download. Finally they carefully read downloaded primary studies to find which were interesting for the review. Our idea is to simulate a portion of their manual selection and check if our automatic selection could reduce the human workload without loosing any interesting paper. Our strategy is the following: we select from [48] the journal containing the highest number of relevant papers, i.e. IEEE Transactions on Software Engineering (IEEE TSE) with 51 relevant papers. Then, we consider all papers of all issues of TSE from 1977 to April 2004, the timeframe Jorgensen's work with the exclusion of the first volume of TSE, which is not present on IEEEXplore. As a consequence we obtain 2215 papers. This set is called W and represent the result of applying a search criterion, that consists in

considering all the papers of a specific journal. Excluding the first volume we have 50 relevant papers contained in W, out of the 51 from TSE included in the original systematic review. W is our dataset, the starting point to evaluate the selection capabilities of our process.

Typos from the original SLR

According to the IEEE TSE[5] citation, we identify a typos from this SLR for the paper indexed by the id number 240 in [48]. The error concerns the year reported, which should be 2001 instead of 2000.

5.1.6 Variables selection

The main outcome under measurement is the manual work, consisting of reading primary studies either entirely or only title and abstract, to select the interesting ones for the subject of the SLR. We measure the manual work as number of papers that are read assuming the number as a proxy for the actual time that would be spent reading the articles. The minimum manual work ideally required is the total number of interesting papers. However, this minimum could reasonably never be reached in SLR. Indeed, the relation $I \subset W$ holds, where I is the set of relevant papers and W is the set containing papers defined by the search criterion. This choice is motivated by the fact that the SLR selected as subject of the case study does not report neither the time spent for papers selection nor which papers were read entirely and which partially (only title and abstract).

As a consequence, we define the following two metrics:

mw is the manual work. More specifically mw_O is the manual work performed in the original SLR, i.e. manually selecting and reading all papers, mw_{NE} is the manual work obtained applying the selection based on the Naive Bayes classifier using original papers (not-enriched process), mw_E is the manual work obtained applying the selection based on the Naive Bayes classifier using enriched papers (enriched process).

t the treatment applied. Three levels are possible: manual, not-enriched, enriched.

Finally we define α, that is the probability of obtaining a test statistic at least as extreme as the one that was actually observed if we assume true the null hypothesis.

[5] http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32

5.1.7 Hypothesis formulation

The last step of the design is the hypothesis formulation. We formulate a pair of null and alternative hypothesis for each of the five research questions.

1. $H1_0 : mw_O \leq mw_E$, recall= 0.95
   $H1_A : mw_O > mw_E$, recall= 0.95

2. $H2_0 : mw_{NE} \leq mw_E$, recall=0.95
   $H2_A : mw_{NE} > mw_E$, recall=0.95

5.1.8 Operation

We developed a Java Tool, Semantic Systematic Review[6], that implements Algorithm 1. We created the bag of words model for each paper and we initialize I_0. After the initialization is concluded, our tool is able to perform automatically the remaining steps. First of all, when a source (or candidate) has to be classified, we extract from it some important concept, defined as entities. The extraction of entities is obtained invoking the OpenCalais web service and selecting the social tags retrieved. The enrichment of the model is performed linking entities found with OpenCalais to DBpedia resources. In such a way, for each resource we have related named entities (with a valid URI) which we use to gather all words defined into the property *dbpedia-owl:abstract* and expressed in English. Finally, these words are added to the bag of words of the corresponding paper and the resulting paper is named enriched paper. The classification of the enriched paper with the model I is done with the Naive Bayes classifier. At the end, our tool simulates the manual selection of relevant papers done by [48] in their SLR, so that we can compute all the metrics introduced above.

We decided to assess the validity of our process with different size of I_0 ranging between 1 and 5. In order to limit the bias introduced by a particular configuration of selected papers, we built 30 different I_0 sets per each dimension choosing them randomly among 50 relevant papers. We used each generated I_0 to kick-off the two variants of the process: enriched and not-enriched; moreover we replicated the experiment varying the classification threshold between 0 and 1 with steps of 0.01. The classifier threshold represents the posterior probability for a sample to belong to the I (interesting set). Summarizing, we executed the complete algorithm 30,300

[6]http://sourceforge.net/projects/semreview/

times: 5 (number of I_0 sizes x 30 (number of I_0 sets for each size) x 2 (variants of the algorithm) x 101 (thresholds).

A preliminary step consisted to define the best classifier threshold (named t in Figure 5.2) which maximizes the recall for the two variants. According to [26], we decided to aim a recall of 95%. Although this recall value is a strong constraint, we adopted it for limiting as much as possible the elimination of interesting papers. In Table 2 is reported the distribution of the maximum classifier threshold which permits to obtain the target recall using the different I_0 sets. We chose the maximum threshold because is the one which minimize the workload while still satisfy the requirement of a recall equal or greater to 95%. We select the median values to set the classifier, that means 0.22 for the enriched process and 0.17 for the not enriched one.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
not-enriched	0.11700	0.1700	0.1700	0.1729	0.1775	0.1900
enriched	0.2100	0.2100	0.2200	0.2201	0.2200	0.2600

Table 5.2. Analysis of the best classifier threshold for both enriched and not-enriched process across different I_0 sets. The first and last column show the minimum and maximum values, second and fifth columns respectively first and third quartile of the distribution, then mid columns show median and the mean of it.

5.1.9 Analysis Methodology

The goal of data analysis is to apply proper statistical tests to reject the null hypotheses we formulated. Since the values are not normally distributed (according to Shapiro test), we adopt non parametric tests, in particular we select the Mann-Whitney test [45] that compares the medians of the vectors of workloads to answer RQ1 and RQ2. To do that, we considered all papers extracted from the dataset except those papers used to build the I_0.

5.1.10 Results

RQ1

Results from the Mann-Whitney test are shown in Table 3. Table reports the I_0 size (column 1), the manual work in the original SLR process (column 2), the manual work obtained with our enriched process (column 3), the estimated percentage of manual work to be performed with our enriched approach with respect to the total work required using the common approach (column 4) and the $p - value$ obtained

from Mann-Whitney test. Given the values obtained we can reject the null hypothesis We can see that the workload reduction increases as the size of I_0. That is probably due to the fact that a better model is available since the first iteration and so the number of unrelated papers shown at that time is smaller when I_0 is larger.

	Workload		Manual workload vs enriched workload	
$\|I_0\|$	mw_O	mw_E	median	$p - value$
1	2214	1897.567	**85%**	< 0.001
2	2213	1864.367	**84%**	< 0.001
3	2212	1863.833	**84%**	< 0.001
4	2211	1843.133	**83%**	< 0.001
5	2210	1829.1	**82%**	< 0.001

Table 5.3. For each I_0 configuration, we first compare the workload required to a human being in the original SLR and the workload mean if our process is performed. To verify the goodness of our process, we build the Mann-Whitney test and we reject the hypothesis $mw_O \leq mw_E$ with a recall = 0.95.

RQ2

As performed before, we used the Mann-Whitney test to reject the null hypothesis by which we state that $mw_{NE} \leq mw_E$. Table 4 reports the I_0 size (column 1), the estimated difference of manual workload between the two processes, and the $p - value$ of Mann-Whitney test. While we can observe that than enriched process requires less workload for every size of I_0 we can affirm it with $p < 0.05$ just when the size of I_0 is 5.

$\|I_0\|$	workload median pairwise difference	$p - value$
1	26.67	0.192
2	66.00	0.073
3	40.83	0.090
4	33.00	0.083
5	**49.99**	0.009

Table 5.4. For each I_0 configuration, we performed the Mann-Whitney test, evaluating median pairwise difference and $p - value$ to estimate the minimum workload using both process: enriched and not-enriched. As for RQ1, the minimum recall is 0.95.

Figure 3 summarizes RQ1 and RQ2. It shows how workload ranges respect to the number of papers used for training different *ic*. On the y-axis is reported the

workload needed for a human being after both processes: enriched and not-enriched. We can observe the substantial workload reduction which we obtained by means of the enriched process.

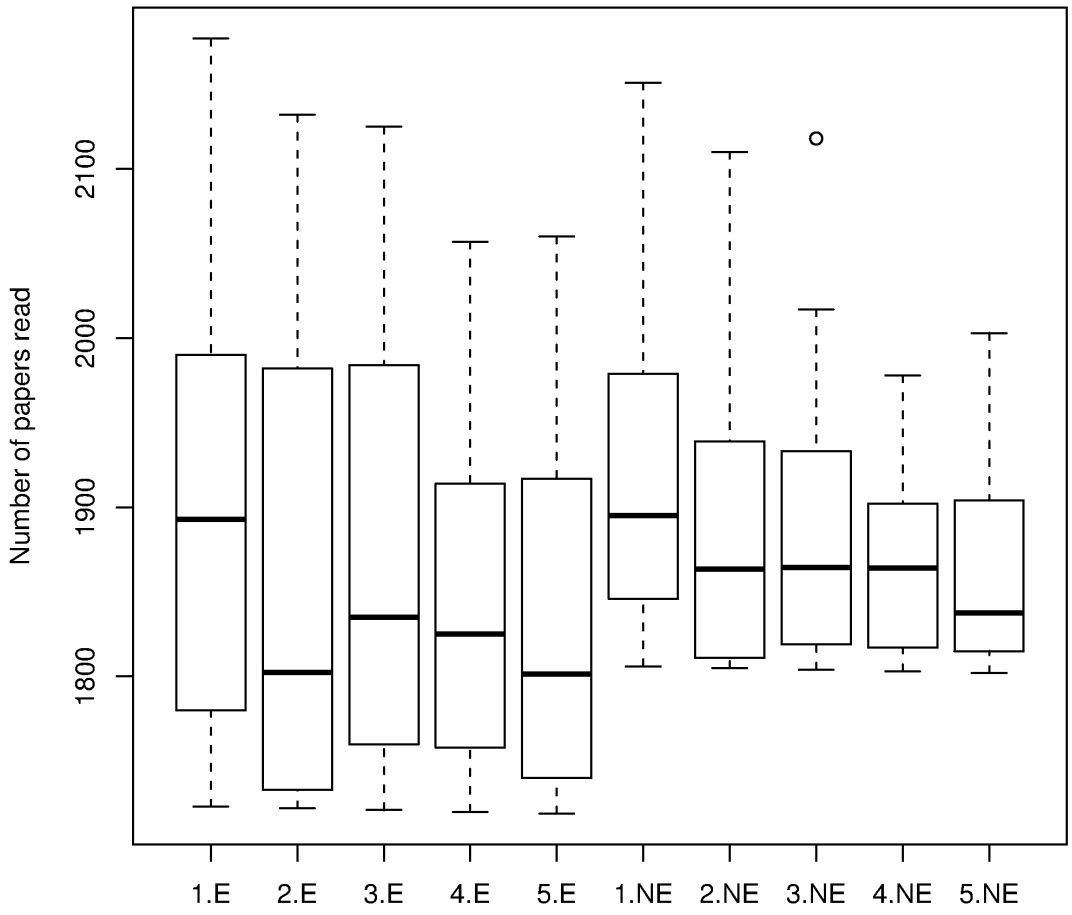

Figure 5.3. Number of papers to read for different I_0 sizes and treatments applied: E (with enrichment) and NE (without).

5.2 Driving SCORM data towards the Web of Data

SCORM (Sharable Content Object Reference Model) is a reference model for the creation of web-based e-learning material with the objective to ensure interoperability, reusability, and accessibility among SCORM data repositories. The SCORM content packaging section specifies how the course material should be packaged, usually in a ZIP file, and described. The course is defined as a collection of SCOs (Sharable Content Object) that can be associated to a lesson or a part of it. A SCO must be a web-deliverable learning unit, that is usually designed as a HTML page with CSS and Javascript so that it can be launched in a web browser. Usually SCORM packages are managed by a LMS (Learning Management System) that, among other important functions such as administration, tracking and reporting, provides also a mean to display this information to the users. In order to correctly deliver the contents, the LMS parses the package metadata to understand the course structure, known as the "activity tree", and to know how to launch each SCO. Since most of times the material takes the form of slides, "traditional" LMS use a fixed HTML frameset structure, that includes a list of the available lessons, a set of navigation buttons, and a panel where the slides are shown. This approach shows limitations:

- The SCO designer can completely define the content and presentation of each single SCO, but it can not control at all the interface used to present and navigate it (that is part of the LMS functions);

- The SCO designer's work area is limited by the dimension of the HTML frame defined by the LMS, thus it is difficult to integrate different media and materials, i.e., a video of the teacher, slideshows, comments;

- SCOs are included in the LMS with the graphical style defined by their own SCO designer, style that can be inhomogeneous if an LMS integrates different SCO sources. If content and presentation where independent, it would be instead easy to redefine each SCO in order to present an homogeneous presentation;

- A fixed interface and the impossibility to separate presentation from content make hard to adapt the interface to different displays, i.e., smartphones, tablets, netbooks, etc.

5.2.1 Zenaminer: a SCORM player for the Linked Data cloud

To overcome these limitations, we propose Zenaminer [68], a SCORM player able to export SCORM packages through a RESTful web API, that is a web service implemented using HTTP and the REST principles. The architecture where a SCORM package is not shared as a single object (the ZIP packet), but as a collection of smaller objects, the SCOs, each of them accessible independently from the others. It is the implementation of the new paradigm of the Web of Data, SCOs are publicly available as read/write raw data that can be retrieved or updated using a REST interface as described in the next sections. The key factors that enable Zenaminer to provide such a functionality are two:

- the knowledge of the SCORM content packaging standard, to import the ZIP files;
- the knowledge of the content internals, i.e. how the slide information is defined, to separate the content from the presentation.

Zenaminer, for the creation of HTML documents, suggests the introduction of a "light formalism" proposed by the W3C with the name Slidy [75]. Slidy defines simple keywords for the class attribute of HTML tags that can be used to identify HTML elements as slides, titles, sidebar, incremental lists, etc. each of them with a particular presentation style defined by CSS or behavior implemented in Javascript. In short, SCORM contents are available as resources of the web service given a specific URI. The SCO designer can freely define the interface as he likes it, then contents will be dynamically retrieved from Zenaminer using AJAX (Asynchronous Javascript and XML) calls. Web API gives access to each single slide of the course as well as to Table of contents for the navigation. In addition Zenaminer extends the SCORM metadata to include also references to additional information such as the video recording of a lesson, synchronization information between video and slides, etc. Taking into account the collaborative learning, Zenaminer enables students to enrich the SCORM resources with comments on the course, the lesson, or the slide. Comments can be contents themselves that extend and improve the original material provided by the teacher. Furthermore, unstructured comments are automatically annotated with DBpedia Spotlight linking them to the Linked Open Data cloud.

5.2.2 Raw content presentation

Educational data have been presented through Slidy, a formalism which uses HTML. The choice to exploit HTML becomes important in our approach because it opens to all devices able to connect and visualize Web pages and, more important, it

intrinsically performs the separation of contents from the presentation. According to that, we define data for the description of contents and data needed to present those contents: this is obtained with the use of the Cascading Style Sheet (CSS) file. By means of it, a view maker can define models to render specific items or group of them. These models are named classes and are referred to an HTML page. The interaction between HTML page and CSS is performed by means of selectors, which are able to point to specific items or class of an HTML page. In this context, Javascript selects an item or a set of them and can easily change the behaviour of the view, previously defined. In addition, it works as a tool to enrich dynamically the presentation, e.g. making table of contents, suggesting the value of an acronyms, changing font or window dimensions. Then, it performs an important role to dynamically access structured data without any layout details, or raw data, which are coming from different archives spread within the Web cloud. The technology that allows to gather data from external sources is the Asynchronous Javascript and XML (AJAX). Our work exploits the Separation of Concerns (SoC) principle, introduced above, in the context of a LMS. The difference between data and view allows to navigate through resources, making customizable views which better respond to the need of who provide contents (author) and who use, reuse and redistribute them (user). Data is presented without layout details, raw, but it is rich of semantics: data is stored with the information about inference to others and are exposed to users by means of REST APIs. Our approach exploits the MVC (Model-View-Controller) paradigm, in which the view is the set of presentation rules for the data, the Model is the amount of data raw available for a generic topic, while Controller is the set of methods which are needed to create the communication channel from the Model and the View. Our data Model is deployed in the system which it holds SCORM packages, while the View is created on all user machines whenever a user require it. The Controller, instead, is located on the user machine where the view is created and on the LMS where SCORMs are available. To provide the maximum interoperability and reusability, contents indexed and managed by our LMS are slides, lectures and courses. All of these are Web resources available by URIs time-invariant. This feature is important because we need to performs durability and reusability to each Web resource. Slide contents are described by means of HTML Slidy formalism; each slide is composed of title, pictures, vertical and horizontal scroll bars, interactive items (buttons, check box, etc.). Lectures are organized as follows: slides and video fragments. A fragment is a part of video which is related to a particular lecture. To map lecture and fragments, we introduced a synchronization file. A video is available and accessible by means of a URI with the above requirements. Course, lecture and slide exploit the SoC principle, separating the view from the data model. The video, instead, holds the information about its presentation and it is shared without changing.

5.2.3 Architecture

The architecture is structured in four building blocks that are summarized in Figure 5.4:

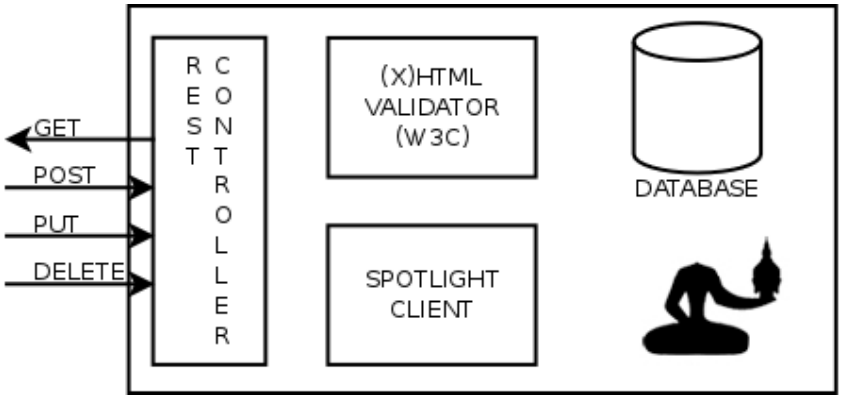

Figure 5.4. Zenaminer architecture. The REST controller is the interface between Zenaminer and the Web; the W3C validator is used to check that imported (X)HTML files are well-formed; the Spotlight client provides automatic annotation; the database stores imported SCORMs together with comments and annotations.

1. The REST controller is the interface between Zenaminer and the Web. Its features are grouped in three sets (as shown in Figure 5.5): user management and authentication (login), get and post of comments (comment) and management of SCORM packages (scorms, item, outline, sync and slide). Each of these features is implemented using the REST architecture, thus resources are available through URIs in the Web. Access to REST calls that provide access to (GET) or modify (POST, PUT, DELETE) existing contents can be limited using the existing facility for user management and authentication.

2. A local validator is used to verify that (X)HTML files are valid and well-formed before they are parsed. When a new SCORM package is imported each HTML file that contains slides is validated using the remote interface offered by the official W3C validator[7].

3. When a comment (an enrichment) is received it is passed to the Spotlight client. The Spotlight client sends a request to DBpedia Spotlight service and receives an annotated version of the comment.

[7] http://validator.w3.org

4. The SQL Database is used to store imported SCORM packages together with comments and annotations.

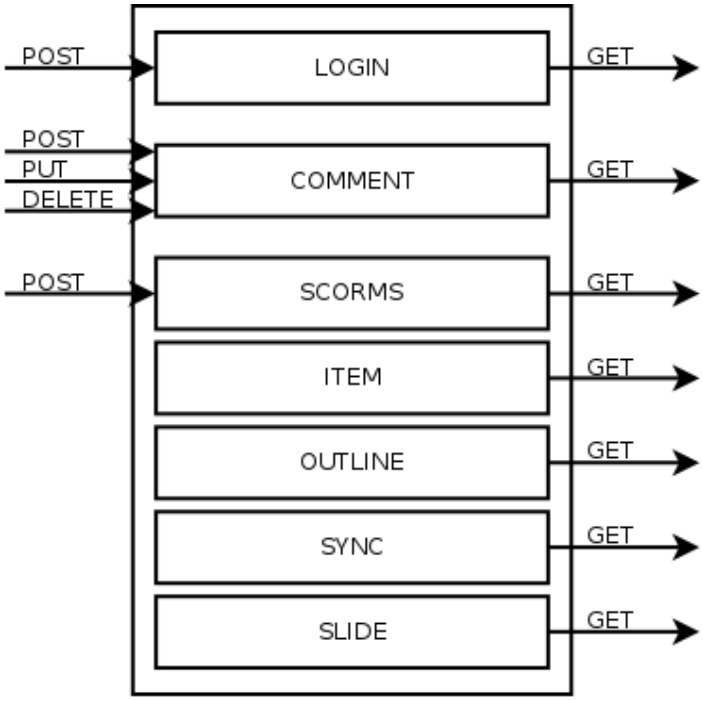

Figure 5.5. Detail of the REST controller, its three main features are: user management, comment management and SCORM management.

Figure 5.6 shows the workflow for the upload of a SCORM package into Zenaminer, it is performed in the four steps below:

1. the client sends a POST request to /scorm page, including in the request body the SCORM package;

2. the REST controller receives the package, unpacks and analyzes the files containing slides and sends them to the W3C validator;

3. the W3C validator validates individual files and reports the result to the REST controller;

4. if all files have been validated correctly the package is imported into Zenaminer. Otherwise if files are not validated Zenaminer sends to the client an error message containing errors generated by the W3C validator.

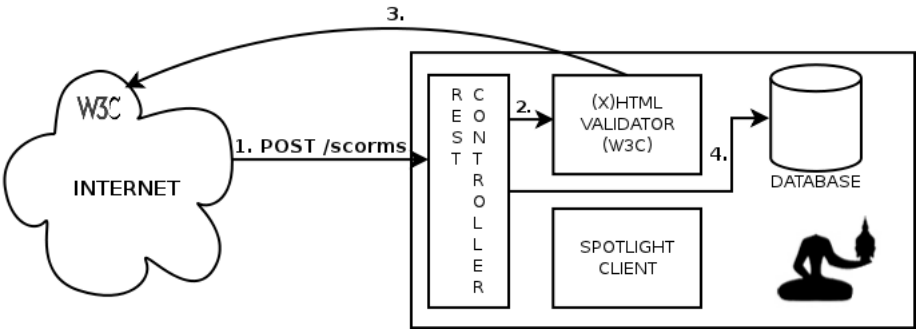

Figure 5.6. Workflow describing the upload of a SCORM package in Zenaminer: 1. a SCORM package is received from a client; 2. (X)HTML files are forwarded to the validator; 3. (X)HTML files are validated by sending requests to the offical W3C validator; 4. if all documents are valid, the SCORM package is stored to the database.

Zenaminer gives the possibility to its users to enrich the content of a given content (e.g. a slide) by adding a textual comment to it. Such comments are named enrichments because they are meant to add more information to the content of a SCO, for example an enrichment to a slide could be a proof to a theorem, a correction to its content, an in-depth explanation to a concept, etc.. In order to avoid that content of an enrichment is buried in the database we added automatic annotation in order to enable machines to access to the content, enhancing search search and the possibility to display complementary information.

Figure 5.7 shows the actions performed by Zenaminer when a comment is received: first the client sends a POST request to /comment page, the message body contains the text of the comment and then the enrichment and its annotations are stored into the database. A SCORM package includes a XML file named "manifest", it describes a course and is used to list all lectures in it, all resources (files) associated to each lecture and some sequencing rules used to define how the user accesses to SCOs. Figure 5.8 shows in a tree structure how the manifest file is represented into the relational database. In addition, a SCORM package lists several SCOs, each SCO includes an HTML file containing slides, a CSS file to describe the style, a Javascript file to describe the behavior. Additional files can be included in a SCO: videos, subtitles or files describing synchronization between a video and the slides. Files linked into the manifest are then packaged together into a ZIP file. Currently Zenaminer does not support sequencing rules, thus management of such rules is up to the SCO designer.

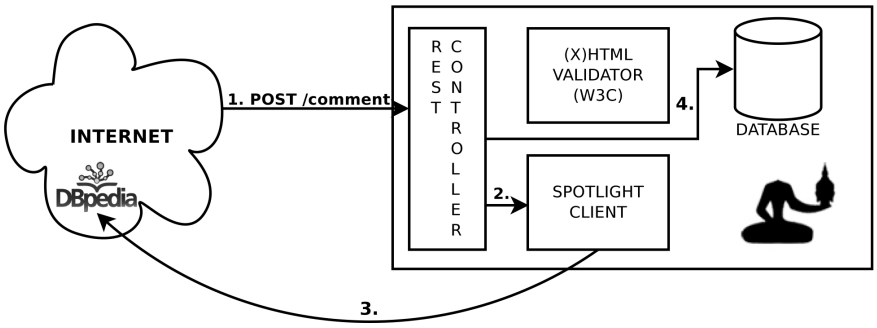

Figure 5.7. Workflow describing the reception of a comment in Zenaminer: 1. a comment is received from a client; 2. the content of the comment is forwarded to the Spotlight client; 3. the Spotlight client contacts DBpedia Spotlight to obtain an annotated version of the comment; 4. the comment together with annotations are stored to the database.

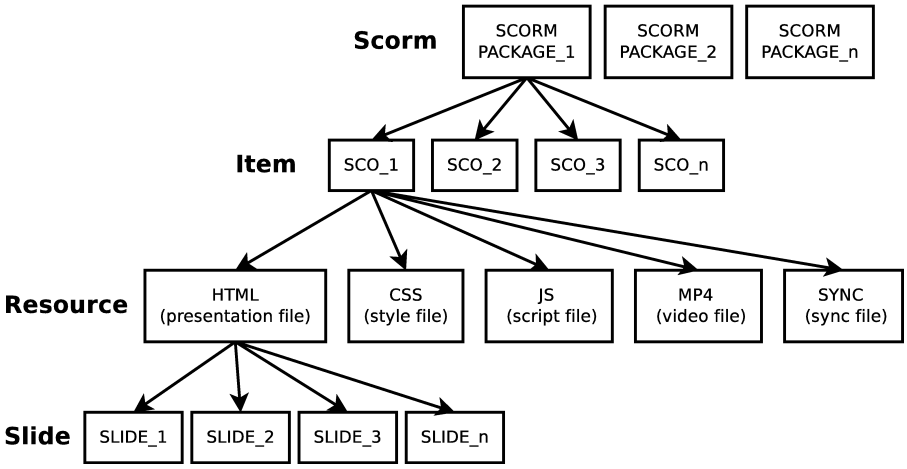

Figure 5.8. A tree representation of a SCORM package in Zenaminer. A SCORM package is a set of SCOs, each SCO represents a lesson in a course, each SCO contains several files like HTML, CSS, JS or videos. HTML files containing a presentation are parsed and slides are extracted.

5.2.4 Test scenario

The validation phase of Zenaminer was conducted during the course "Multimedia Environments" (Academic Year 2010/2011) for the Master of Science in Cinema and Media Engineering at the Politecnico di Torino. Zenaminer was thought as a service in order to design e-learning projects (SCORM packages) for the course "Multimedia Environments". The objective for the students was to create a SCORM package defining both content and presentation. Students were divided in 20 teams, each

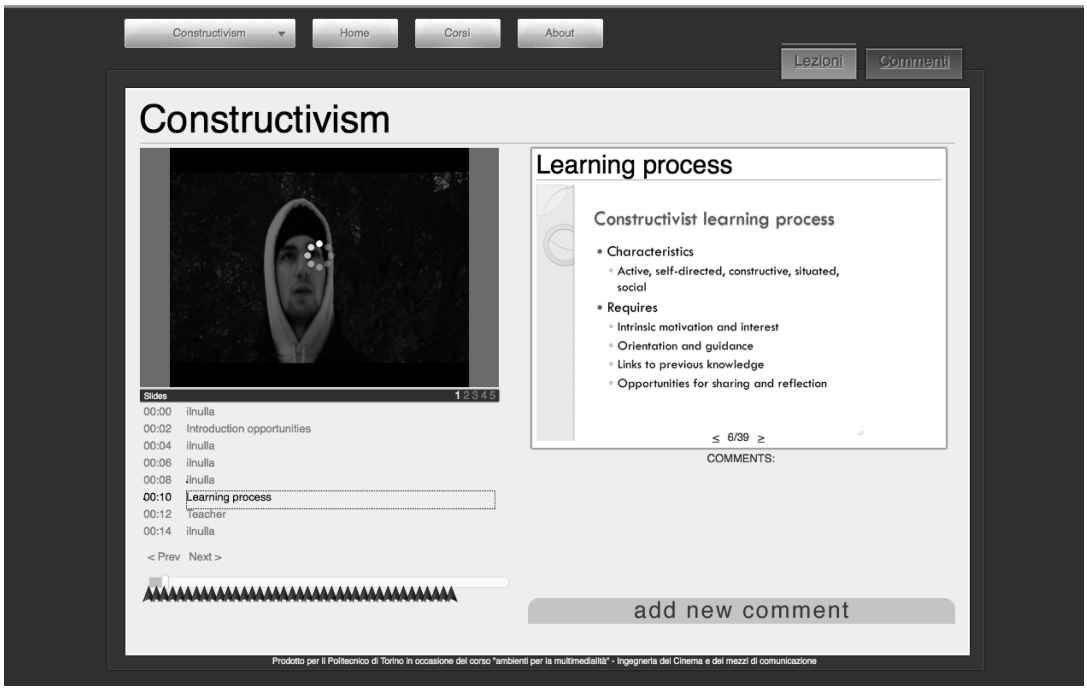

Figure 5.9. One of the user interfaces designed by students for the course Multimedia Environments.

team defined a personal learning environment building different interfaces (using the SoC concept). Students acted as SCO designers, the projects were the use case for Zenaminer and we verified the potentiality of the separation of content from presentation. Figure 5.9 and 5.10 show two different views of the same raw data done by two different groups of students. Such interfaces are able to show same contents in two different ways. The controller of each interface collects the list of SCORM packages stored into Zenaminer. Depending on the SCORM package selected by the user, the controller gathers the entire set of lectures for that package and displays the related table of contents. The user is, then, free to navigate the lectures (SCOs) following the content list, when a lecture is selected the controller requests all files associated to that lecture and the view is updated accordingly. For example, in both figures the selected SCO had a set of slides and a video associated to it. Thanks to the sync file, the view is able to synchronize the video with the slides.

Zenaminer has been developed with the framework Pylons[8]. The source code[9] is

[8] http://pylonshq.com

[9] http://sourceforge.net/projects/zenaminer

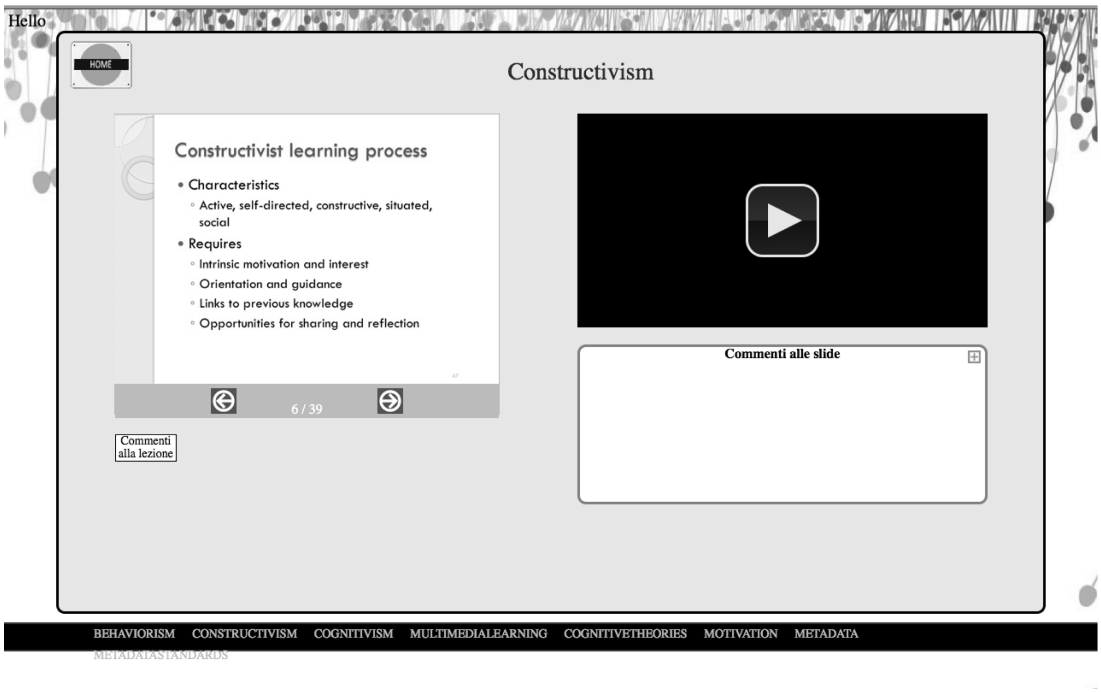

Figure 5.10. Another example of user interfaces designed by students for the course Multimedia Environments.

released under the GNU GPLv3. In order to better balance and distribute the load we decided to use Apache HTTP Server[10] as interface between the requests towards the Web Service. As relational database we used PostgresSQL[11]; it stores data extracted from the SCORM by the web-service. Raw data stored in the database are available through REST calls and serialized into JSON format. The W3C Markup Validation Service is used to validate the HTML of the lectures that must be conform to W3C standards. Finally we used Spotlight to automatically annotate the enrichments of learners.

5.3 Conference enhancer for the Linked Data age

Scientific conferences trigger an ever-growing amount of activities on social media. But in contrast to events like concerts or sport matches, a conference is highly structured, consisting generally in workshops and tutorials, parallel sessions composed of

[10] http://httpd.apache.org

[11] http://www.postgresql.org

talks, keynotes, panels, posters and demos that all have planned schedules, topics, and allocated rooms. The Semantic Web community is used to model this structured data using RDF and to publish it following the Linked Data principles using a so-called Semantic Web dog food server[12] [67]. The social media activities that is shared around the conference consists in slides, photos and videos posted by authors and participants but also status messages published on social networks such as Twitter, Google Plus or Facebook. The problem is that these activities are unstructured data, spread over multiple platforms that are just weakly associated to a conference event as a whole as opposed to its fine grained sub-events. Overall, the physical participants or the ones who try to follow the event online are force to monitor multiple channels to full benefit from a scientific conference. We propose a semantic web application called Confomaton which collects social media activities, reconciles the data and attempts to align it to the various sub-events that compose a conference. Ultimately, Confomaton presents this information as a live visual summary of the conference, enabling a user to re-live the event afterwards and catch up with what (s)he could have missed. This concept, which has been recently awarded at the International Semantic Web Conference 2011, poses outstanding challenges regarding data aggregation and curation, classification and alignment with sub-events, and user interface. We describe these problems and present possible solutions.

5.3.1 Motivation

Building a rich Web-based environment to explore social media content during live events presents a number of challenges. Popular social media services host a substantial amount of photos, posts, and videos illustrating public events. Aggregating these heterogeneous sources of information by consuming Linked Data requires advanced techniques in data collection and reconciliation. Moreover, a public event such as a scientific conference may involve a large number of participants with accounts on different social networks sites. They contribute to generate an evolving space of shared media which needs to be aggregated in realtime and associated with appropriate sub-events.

Exploring this intrinsic connection between events and media shared on the web has been the focus of several studies [59, 92]. They propose different techniques in the area of media classification, data interlinking and event detection, trying to leverage the wealth of user generated knowledge. However, most of these studies have mainly targeted a specific social service such as Twitter or Flickr, without any guarantee that they can be valid for others services. We believe that exploiting the diversification of user generated content from different social services inside one

[12] http://data.semanticweb.org/

application is a challenging task. In this work, we aim at creating a rich environment to enable users navigating events as well as their various representative media such as photos, slides and tweets. A typical usage is to gather data about a scientific conference and investigate the added value of collecting scientific-related media. A non trivial task in such application is to connect structured data with extremely noisy content, especially in the case of a major conference.

In this work, we consider the International Semantic Web Conference (ISWC 2011) which took place in Bonn (Germany). Broadly speaking, considering all co-authors, people who have participated in the reviewing process, people who have attended physically the conference or tried to follow it on social media, we estimate that it has attracted more than 1500 participants. The conference organizers publish a lot of structured data regarding the conference including the list of accepted papers, their authors and institutions, the detailed program composed of sub-events with the exact timetable and the location (rooms) of the talks. This data is modeled using the SWC ontology[13] following the Linked Data principles in the Semantic Web Dog Food server. The SWC vocabulary is designed to describe academic events, and uses classes and properties from other ontologies such as FOAF (for people) and SWRC (BibTeX elements for the papers). The main conference of type `swc:ConferenceEvent` is related to set of sub-events (`WorkshopEvent`, `TutorialEvent`, `SessionEvent`, `TalkEvent`) via the property `swc:isSuperEventOf`.

Table 5.5 shows some statistics about the data provided by the Dog Food server regarding the ISWC 2011 conference. We first notice that the data is incomplete. The conference has hosted 16 workshops in total, but the 75 papers are only associated to 8 of them while the 8 others did not have any papers according to the corpus. Furthermore, we find 133 papers which are not connected to any of the events via the predicate `swc:hasRelatedDocument`. Finally, some useful information are also missing such as the keynote speakers and the `Semantic Web Death Match` (panel) event. This lack of knowledge is also a motivation for our work: can we collect and analyze social media activities in order to complete the factual description of this event?

We collected social media data in real time during the six days of the conference using the main tags advertised by the organizers (#iswc2011, #cold2011, #derive2011, etc.). Table 5.6 shows some statistics about the different media services used by the attendees along with the number of items from a number of distinct users. As expected, Twitter is by far the most used service: we have been able to collect 3390 tweets from 519 different users. A significant proportion of tweets contains hyperlinks that we have further analyzed. Hence, we extracted 384 different websites indexed by so-called shorteners (shortened URL such as Bit.ly) found in

[13] http://data.semanticweb.org/ns/swc/ontology

Main Event	Sub-event	Number of events	Papers	Authors
Conference Event	Workshop Event	16	75	185
	Tutorial Event	7	7	20
	Session Event	1	66	202
	Talk Event	93	93	275
	-	-	133	385
Total (distinct)	4	117	292	735

Table 5.5. Metadata provided by the Dog Food Server for the ISWC 2011 conference.

1464 tweets (43% of tweets). These links represent a rich source of media as they pointed to various web resources categories such as blogs, slides, photos, publications and projects. For example, 25% of these links pointed to PDF documents which are generally one of the conference papers but could also be related papers relevant for the followers of the conference. We also analyze these links to extract the various media services used by Twitter.

Media Service	Items	Users
Twitter	3390 tweets	519
pic.twitter	12 photos	6
yfrog	10 photos	9
Twitpic	10 photos	6
Flickr	47 photos	6
GooglePlus	30 posts	26
Slideshare	25 slides	20

Table 5.6. Media services used during ISWC 2011 conference

A deeper analysis of the tweets reveals insights about the highlights of the conference, or at least, the parts that drew the most attention. Table 5.7 shows the top-ten hashtags found in tweets. The most used tags are related to the main conference, some popular workshops and some concepts widely used by this community. The second most tweeted event is the *Semantic Web Death Match 2011* associated with the hashtag `#deathmatch`. Although this event was scheduled in the main conference program, its description is not available in the Dog Food corpus. We argue that applying an event detection approach [24] on tweets may lead to discover new sub-events to enrich this corpus.

We also analyze the number of tweets per day as depicted in Figure 5.11. We observe a higher peak on *Wednesday* with 1122 tweets from 248 users (almost 47% of all twitterers in ISWC2011). The reason for this peak is two-fold: the *Semantic*

Hashtag	Frequency	Associated Item
#iswc2011	2916	ISWC 2011 main Conference
#linkeddata	215	Linked Data Concept
#deathmatch	213	SemWeb Death Match 2011 Event
#sdow2011	172	SDOW 2011 Worksop
#cold2011	170	COLD 2011 Workshop
#lisc2011	96	LISC 2011 Workshop
#keynote	81	Keynote Event
#semweb	76	Semantic Web Concept
#semanticweb	75	Semantic Web Concept
#derive2011	70	DERIVE 2011 Workshop

Table 5.7. Top-ten hashtags used by the tweeterers during ISWC 2011

Web Death Match 2011 event during which 144 tweets were posted, and the keynote talk given by *Frank van Harmelen*, his name appearing in at least 136 tweets during this day (almost 12% of Wednesday tweets), and referenced 363 times in all the tweets. Both events (the panel and the keynote) are not described in the Dog Food corpus while social media activity clearly shows that they represent an highlight of the conference.

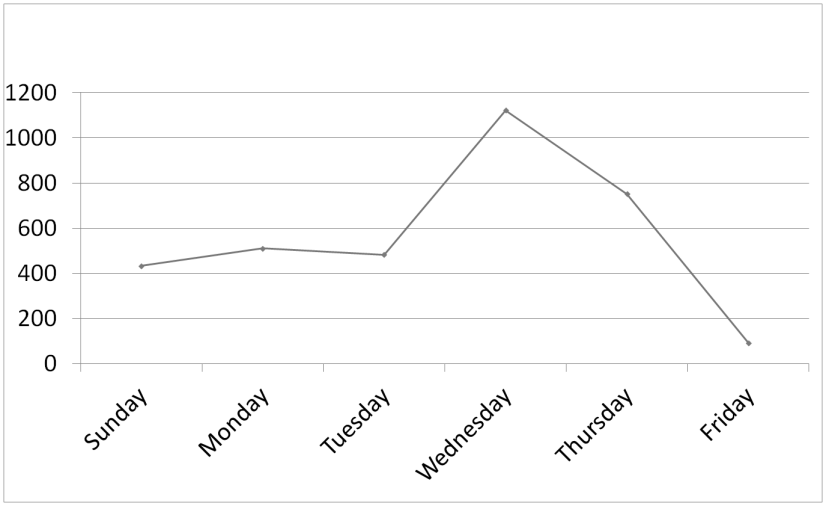

Figure 5.11. Number of tweets per day during ISWC 2011

5.3.2 Confomaton

The name *Confomaton* is a word play on the French term *Photomaton* (English photo booth) and *conference*. Just like a Photomaton illustrates the scene inside of the booth, the Confomaton illustrates an event such as a conference enriched with social media. Confomaton is a semantic web application that produces and consumes linked data. Its architecture is composed of three main components (Figure 5.12): (i) a Media Collector which is in charge of collecting social media content and represent it in RDF using various vocabularies; (ii) an Event Media reconciliation module playing the role of associating social media with sub-events; (iii) a User Interface powered by an instance of the Linked Data API as a logical layer connecting all the data in the triple store with the front-end visualizations.

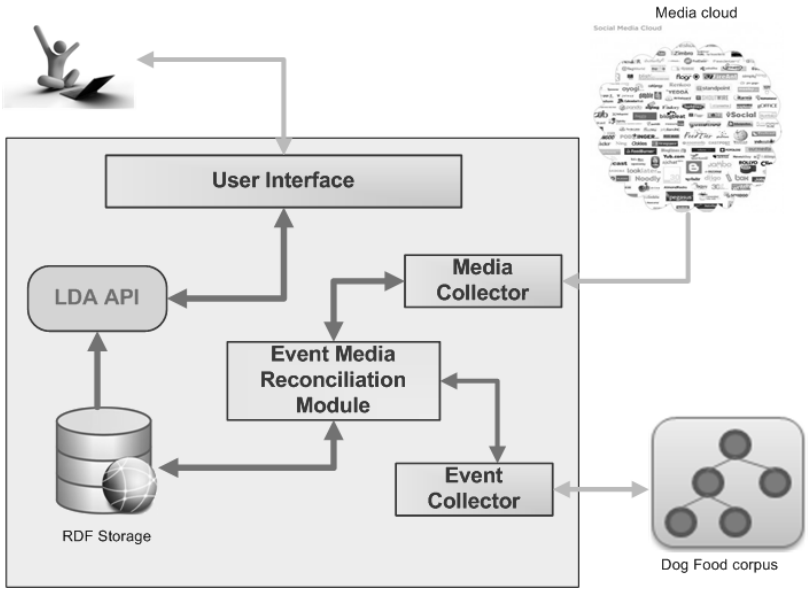

Figure 5.12. Confomaton general architecture.

Unify media platforms

We used a web service media unifier[14] with the purpose of searching various social networks and media platforms for event-related media items such as photos, videos, and slides. It currently supports the social networks Google+, MySpace, Facebook, and Twitter, and the media platforms Instagram, YouTube, Flickr, MobyPicture, img.ly, and TwitPic. In the following, we simply refer to media platforms and social

[14] https://github.com/tomayac/media-server

networks as *media providers*. The approach being agnostic of media providers and it offers a common alignment schema for all media providers, which allows us to treat each media provider data the same way. The resulting set of metadata for a media item can be seen below:

- Media URI, the deep link to the media item, e.g., http://farm7.staticflickr.com/6059/6290784192_567346ba6a_o.jpg
- Type, the type of the media item, e.g., "photo"
- Story URI, the URI of the micropost or story where the media item appeared, e.g. http://www.flickr.com/photos/96628098@N00/6290784192/
- Message, the concrete micropost or description text in raw format, e.g., "Laura. #lumixg20f17, #iswc2011, #internationalsemanticwebconference, #bonn, #germany"
- Clean, the concrete cleaned micropost or description text with some characters (e.g. hash sign) removed, e.g., "Laura. lumixg20f17, iswc2011, internationalsemanticwebconference, bonn, germany"
- User, the URI of the author of the micropost, e.g., http://www.flickr.com/photos/96628098@N00/
- Published, the timestamp of when the micropost was authored, or the media item was uploaded, e.g., `2011-10-27T12:24:41Z`

In order to retrieve data from media providers, we use the particular media provider search Application Programming Interfaces (API) where they are available, and fall back to screen scraping the media provider's website if not. In some cases, we initially use the search API, but then have to fall back to screen scraping in order to get more details on the results, such as the **Media URI** which is not exposed by all APIs. While APIs are generally stable, screen scraping is a very brittle and time-consuming process, which is why we see it more as a necessary evil than as a future proof direction. From all media providers, Twitter plays a special role, as it can serve as a host for other media providers. For example, it is very common for tweets to contain links to media items hosted on external media providers such as TwitPic. Other media providers treat media items as first class objects, i.e., have dedicated object keys in their API results for media items, which is not in all cases true for Twitter. We handle this by searching for a list of URIs of known media providers in combination with the actual search term. To illustrate this, when searching for media items for the search term "Eiffel tower" on Twitter, we would actually search for "eiffel tower AND (twitpic.com OR flic.kr)" in the background, whereas on all other media providers, the search term "eiffel tower" is sufficient.

Data Modeling

We have developed an Event Collector which takes as input the Dog Food corpus described using the SWC ontology and converts all events into the LODE ontology[15], a minimal model that encapsulates the most useful properties for describing events. We use the Room ontology[16] for materializing the various rooms contained in the conference center. An explicit relationship between an event and its representative media (photo, slide, tweet, etc.) is realized through the `lode:illustrate` property. For describing those media, we re-use two popular vocabularies: the W3C Ontology for Media Resources[17] for photos and videos, and SIOC[18] for tweets, status, posts and slides. The example below shows how a tweet is represented in Confomaton.

```
<http://data.linkedevents.org/tweet/af557cef-5d5b-49c6-a4c3-bc9c41ce1555>
 a sioc:Post;
 dcterms:created "2011-10-23T13:34:03+00:00";
 sioc:content    "@smeh Good luck for your presentation at #ssn2011...";
 sioc:hasCreator <http://www.twitter.com/BadmotorF>;
 lode:illustrate <http://data.semanticweb.org/workshop/ssn/2011>;
 gc:hashtag      "#ssn2011";
 owl:sameAs      <http://twitter.com/BadmotorF/status/128071685235671040>.
```

Figure 5.13 depicts how all these vocabularies are used together. The ISWC 2011 conference is illustrated by a photo shared on Flickr, has for sub-event the EvoDyn 2011 workshop in which one of the tweets posted mentioned the recognized named entity Nathasha Noy who is also a general chair of the conference. All the data is available in the Confomaton graph in a public SPARQL endpoint available at http://semantics.eurecom.fr/sparql.

Event Media Reconciliation Module

The Event Media reconciliation module aims to align the incoming stream of social media with their appropriate events and to interlink some descriptions with general knowledge available in the LOD cloud (e.g. people and institutions descriptions). Attaching social media to fine grained event is a challenging problem. We tackle it by pre-processing the data with two successive filters in order to reduce the noise: the former relies on keyword search applied to some fields such as title and tag, while the latter filters data based on temporal clues.

[15] http://linkedevents.org/ontology/

[16] http://vocab.deri.ie/rooms

[17] http://www.w3.org/TR/mediaont-10/

[18] http://rdfs.org/sioc/spec/

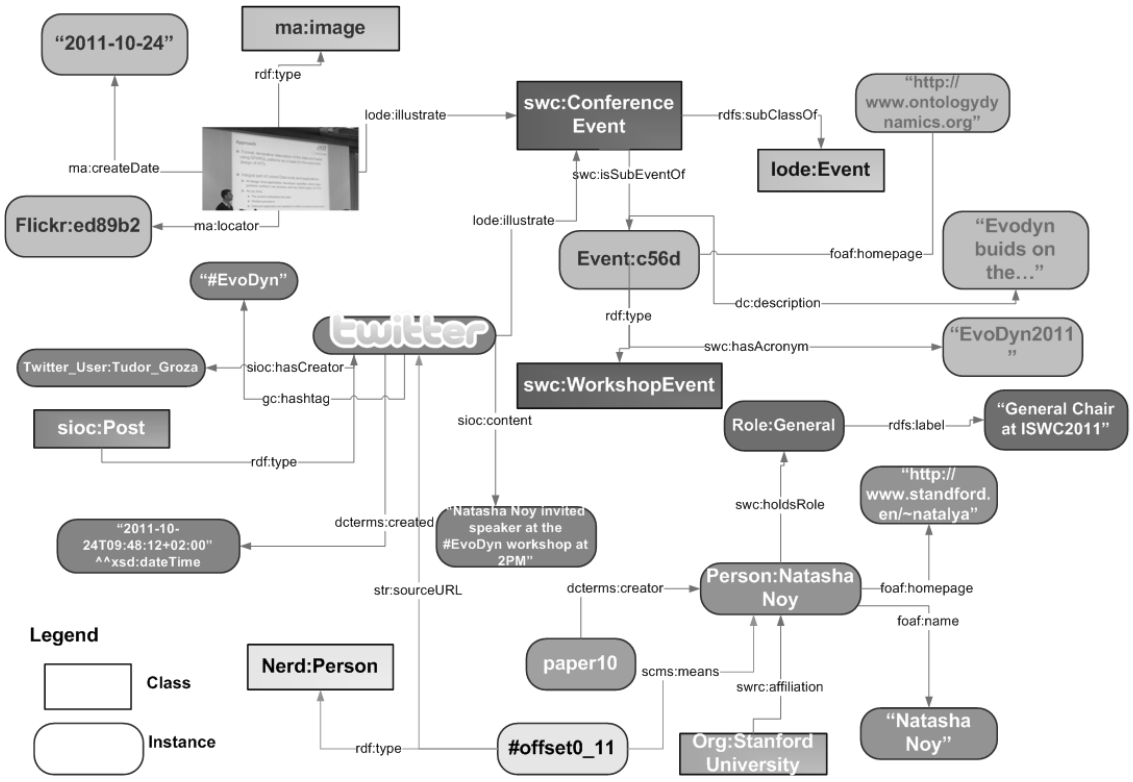

Figure 5.13. Example of data modeled in Confomaton re-using multiple vocabularies

The reconciliation is then ensured through a pre-configured map between a set of keywords and their associated events. This map enables us to associate media with the macro-events which people explicitly refer to in their posts or photos. For example, we connect each media containing the tag *"iswc2011"* with the general ISWC 2011 conference. However, this method is absolutely not convenient to associate media with sub-events. For instance, in the ISWC 2011 conference, there are 99 micro-events of type `TalkEvent` which could be the presentation of a paper, a keynote speech or any other kind of talk. Social media services users usually do not specify a particular tag for such events. We hence advocate the need for more advanced classifiers to associate media with sub-events. These classifiers can exploit a variety of parameters such as geographic metadata and named entities extracted from media content, which will be the focus of our future work.

User Interface

The User Interface (UI) of Confomaton is built around four perspectives characterizing an event reflected in the forms of tabs: (i) *"Where does the event take*

place?", (ii) *"What is the event about?"*, (iii) *"When does the event take place?"*, and finally (iv) *"Who are the attendees of the event?"*. In addition, the user interface offers full text search for these four dimensions. The system is available at http://semantics.eurecom.fr/confomaton. The Confomaton user interface is powered by the Linked Data API[19] which provides a configurable way to access RDF data using simple RESTful URIs that are translated into queries to our SPARQL endpoint. More precisely, we use the Elda[20] implementation developed by Epimorphics. Elda comes with some pre-built samples and documentation which allow to build specification to leverage the connection between the back-end (data in the triple store) and the front-end (visualizations for the user). The API layer helps to associate URIs with processing logic that extract data from the SPARQL endpoint using one or more SPARQL queries and then serialize the results using the format requested by the client. A URI is used to identify a single resource whose properties are to be retrieved or to identify a set of resources, either through structure of the URI or through query parameters. On the left side of the main view, the user can select the main conference event or one of the sub-events as provided by the Dog Food metadata corpus. On the center, the default view is a map centered on where the event took place (e.g. Bonn, Germany) and the user is also encouraged to explore potential other type of events (concerts, exhibitions, sports, etc.) happening nearby, this data being provided by EventMedia [92]. The *What* tab is media-centered and allows to quickly see what illustrates a selected event (tweets, photos, slides). Zooming in an event triggers a popup window that contains the title and timetable of the event, the precise room location and a slideshow gallery of all the medias collected for this event. For the *When* tab, a timeline is provided in order to filter events according to a day time period. Finally, the *Who* tab aims at showing all the participants of the conference. This is intrinsically bound to a social component, aiming not only to present relevant information about a participant (his affiliation, homepage, or role at the conference) but also the relationships between the participants between themselves and with the events.

5.3.3 Media content analysis

As we have seen in the section 5.3.1, the Dog Food metadata corpus provides fine-grained information about scientific conferences. However, we have also shown that this information is not yet complete. One of our motivation is to analyze the media content shared on social networks in order to enrich the conference program. In this section, we present the results of an analysis conducted on all the tweets collected

[19] http://code.google.com/p/linked-data-api/wiki/Specification

[20] http://code.google.com/p/elda

during the conference. We performed several NE (Named Entity) extractions using the NERD API [82], a service that aggregates the results of numerous named entity extractors such as AlchemyAPI[21], Extractiv[22], OpenCalais[23], Wikimeta[24], and Zemanta[25]. It is worth noting that tweets often contain informal terms or abbreviations which can limit the performance of these tools.

We define the following metrics: word detection rate $r(w,t)$, i.e. the number of words per tweet, entity detection rate $r(e,t)$, i.e. the number of entities per tweet, the number of entities per word $r(e,w)$, the number of categories per tweet $r(c,t)$ and the number of URIs per tweet $r(u,t)$. The word detection rate per tweet $r(w,t)$ is equal to 12.5 with an overall number of entities equal to 23761, and the $r(e,t)$ is 1.154. Table 5.8 shows the statistics for each extractor (all the values are approximated at the third digit). Wikimeta recognizes the highest number of NEs in this dataset, respectively $r(e,t)$ equal to 2.009 and 1.844. A common behavior for all the extractors, except for Zemanta, is an accurate NE classification. The entity disambiguation, instead, is a task which produces heterogeneous results: OpenCalais has the highest volume of disambiguated URIs, with a revealing performance of the $r(u,t)$ equal to 0.338. It is crucial noting that Wikimeta has a strong ability to locate NEs and at the same time to classify and disambiguate them. Finally, we can observe that the magnitude order of the $r(e,w)$ is the same for all the extractors involved in this analysis.

	n_e	n_c	n_u	$r(e,t)$	$r(e,w)$	$r(c,t)$	$r(u,t)$
AlchemyAPI	1787	1787	97	0.527	0.013	0.527	0.029
Extractiv	1755	1755	36	0.518	0.012	0.518	0.011
OpenCalais	4699	4667	1147	1.386	0.033	1.377	0.338
Wikimeta	6251	6241	583	1.844	0.044	1.841	0.172
Zemanta	2460	580	476	0.726	0.017	0.171	0.140

Table 5.8. Statistics about computation results from all NE extractors on the entire tweets dataset.

All these NE extractors have their own taxonomy. NERD provides an ontology that fully align them[26] enabling further comparison. We report below the grouping

[21] http://www.alchemyapi.com

[22] http://extractiv.com

[23] http://www.opencalais.com

[24] http://www.wikimeta.com

[25] http://www.zemanta.com

[26] http://nerd.eurecom.fr/ontology

results according to 6 main concepts: `Person`, `Organization`, `Country`, `City`, `Time` and `Number` (Table 5.9). Wikimeta classifies a higher number of Person, Organization than all the other tools, but it fails to identify other classes which is mainly due to the small taxonomy it uses. It is worth noting that Extractiv is the only extractor able to extract and accurately classify Time and Number, the former being interesting for recreating the detailed program of an event.

	AlchemyAPI	Extractiv	OpenCalais	Wikimeta	Zemanta
Person	879	71	568	1340	138
Organization	54	-	47	2742	16
Country	13	4	13	-	34
City	47	11	47	-	9
Time	-	5	-	-	-
Number	-	62	-	-	-

Table 5.9. Number of axioms aligned for all the extractors involved in the comparison according to the NERD ontology.

We compute the top 10 ranking of NE frequencies according to the classes Person (Table 5.10) and Organization (Table 5.11). We report in bold the instances which are identified as positive, but that we manually classified as false. To validate this process, we used the disambiguation URI, when available, to check if the NE classified with a class points to a real world object which describes the NE itself. Most of these tools show good performances for classifying Person while Organization has much more false positive and true negative. For example, AlchemyAPI disambiguates the named entity `BTC` with the "British Transport Commission"[27] instead of the "Billion Triple Challenge". Similarly, the string `RT` is classified by Wikimeta as an organization linked to the "Rio Tinto enterprise"[28] while it simply means "Retweet".

5.3.4 Discussion

Confomaton is an ambitious project that shows well the difficulty to use linked data technologies in a real setting. The solution we propose makes use of many services starting from scrapping, aggregating data and their reconciliation. The architecture we proposed makes relatively easy the integration of these existing technologies. However, the more services is handled and the more issues one has to deal with,

[27] http://dbpedia.org/resource/British_Transport_Commission

[28] http://www.wikimeta.com/perl/display.pl?query=RioTinto(entreprise)&search=FR

generally with the API provided by those services (e.g. the number of requests of some APIs, 1500 requests per day for the Twitter API).

Concerning the Linked Data API, at the time we write this paper, it was not possible to handle queries using selectors with *DISTINCT* and *GROUP BY* queries, although there are means to go through this limitation using *UNION*. We also face the problem of the objective criteria to select a particular vocabulary for modeling the data. For instance, how to choose the right model for a hashtag.(e.g., hashtag a `sioc:Topic`?) or the right property for the date of a tweet (e.g., `rdfs:label` for a tweet creation date?) or how to align a tweet post with `opo:OnlinePresence` or just with a `sioc:Topic`? There is a need for providing more guidelines and metrics to help the developers in deciding which vocabulary best fit their need in the design phase of the application.

Dealing with the graphical user interface is not so easy, at least when having in mind what to display, there is sometimes a "gap" with the actual view implementation, due sometimes to time constraints in the prototyping process. Therefore it is very important to be the first access to the data generated behind the scene, but time consuming for the efforts to find the suitable and attractive way to show the data to the user.

AlchemyAPI	#	Exctractiv	#	OpenCalais	#	Wikimeta	#	Zemanta	#
Gerhard Weikum	55	**Confomaton**	8	Gerhard Weikum	55	Gerhard Weikum	49	Alex Pentland	32
Chris Welty	43	Denny Vrandecic	6	Chris Welty	45	Chris Welty	46	Ian Horrocks	13
Alex Pentland	24	Claudia Wagner	6	Alex Pentland	32	**http://t.co**	41	Mark Greaves	10
Stefan Schlobach	20	Chris Welty	5	Sandy Pentland	19	Yahoo	35	Matthew Rowe	4
Sandy Pentland	19	Natasha Noy	5	Csaba Veres	18	Alex Pentland	29	John McCarthy	4
Natasha Noy	19	chris ivan	4	Natasha Noy	18	**Linked**	19	Aditya Kalyanpur	3
Fabian Abel	17	Rudi Studer	3	Stefan Schlobach	18	Sandy Pentland	19	Paul Groth	3
Csaba Veres	16	Chris Bizer	3	Fabian Abel	15	Fabian Abel	18	Ansgar	2
Lalana Kagal	14	**Don t**	3	Ian Horrocks	11	Stefan Schlobach	18	Amit Sheth	2
Marko	13	Tom Gruber	3	Martin Hepp	11	Natasha Noy	17	Saw Rudi Studer	1

Table 5.10. Top 10 ranking of NEs classified as Person for each extractors

AlchemyAPI	#	OpenCalais	#	Wikimeta	#	Zemanta	#
EU	11	European Union	17	RT	982	IMO	6
Open University	8	Open University	8	**RDF**	78	AAAI	5
W3C	8	MIT	7	KB	34	**W3C standards**	4
SPARQL Endpoint Federation	5	Press Association	4	**Of Objects**	33	WWF	1
Falcons	4	New Model Army	2	**OWL2**	32		
OASIS	3	**Party on Garth**	1	Social Semantic Web	27		
New Model Army	2	**Party on Wayne**	1	**http://t.co**	26		
MIT	2	**@anjeve Too**	1	**LOD**	20		
CC	2	**Query Federation**	1	**SPARQL**	19		
BTC	1	Semantic Web Science Association	1	**RT @takechan2000**	18		

Table 5.11. Top 10 ranking of NEs classified as Organization for each extractors

Chapter 6

Conclusions

As the increasing of the volume of interconnected data spread in the Web, becomes crucial to handle efficiently data in terms of data storing and data retrieval. The availability of well known techniques in Data Management and Natural Language Processing communities boost the research efforts in the Semantic Web community towards an efficient landscape of data management.

This dissertation thoroughly investigated on the distribution of RDF triples in a set of nodes which share the same storing and retrieval logic. It proposed a distributed RDF triple storage based on a P2P network. The triple distribution logic among the network of peers is optimized in order to have each node responsible for nearly the same amount of data. Each field of the triple, excluded the literal, is hashed and distributed according to the long distance links that are defined using the estimated network load information. This logic addressed the problem of network failures and frequent node join and leave events with a redundancy mechanism, i.e. a portion of each peer storage is used to store copies of triples managed by the other peers. Finally ensuring that if a link fails or a node abruptly disconnects from the network, no triples are lost and a query is able to return consistent results. The effectiveness of the presented architecture has been analyzed by monitoring the load balancing algorithm and the overhead introduced on the network load in both a static (only join events) and dynamic scenario. Afterwards, two real applications have been built on the top of this assumption to drive human users during the annotation of media (audio and video) contents.

Moreover, this dissertation investigated the crucial challenge of the current Web: how to automatically structure data from the enormous amount of legacy documents spread on it. Many research efforts have been spent to answer to this challenge. This dissertation presented the NERD framework developed following the REST principles and the NERD ontology, a reference ontology to map several NE extractors. Furthermore, a thorough comparison of 10 named entity extractors in particular task, scenario and setting is presented. Two different experiments are proposed:

qualitative experiment and quantitative experiment. For the former, two different benchmarks have been created using human ratings for the bag of entities provided by these extractors when they parsed textual web resources. Each human rating evaluated using a Boolean value the correctness of each field of the tuple *(NE, type, URI, relevant)*. In other words, the evaluation consisted in analyzing the precision of the named entity extraction task, precision of the classification of each NE, precision of the real word object identification within the LOD via URI, and precision to identify relevant NE for the text. The Fleiss's kappa scores obtained show a general agreement for each different evaluation. The final precision among all the extractors was calculated taking into account the human ratings according to the assessment gave about each field. The precision results showed the strengths and weaknesses of seven different extractors. In the quantitative experiment, a large scale analysis of the extractors under analysis has been conducted in particular task, scenario such as scientific papers, news papers, and user generated contents. The goal was to assess the performance variations according to different kind of texts and different text length. Results showed that some extractors are affected by the word cardinality and the type of text, especially for scientific papers. DBpedia Spotlight and OpenCalais are not affected by the word cardinality and Extractiv is the best solution to classify NEs according to "scientific" concepts such as Time and Number. The importance to have a system able to compare them is under investigation from the NIF project. NERD has recently became part of NIF and the NERD ontology is a milestone for creating a reference ontology for this task.

The Web has been characterized by a remarkable variety of data publishing. Linked Data is able to encompass this heterogeneity, linking each other data spread on the cloud. As a step towards this vision, this dissertation presented three Linked Data applications. For decades educational data have been stored in data silos, making though the process of linking them to other web resources. The first Linked Data application presented taking into account this limitation and it drives the SCORM standard to the Web of Data. The selection process in a SLR has been conducted by human beings. This process is time consuming and a considerable number of research efforts have been spent to help this step and to save time to humans. The second Linked Data application presented links data mining fundamentals to the Linked Data principles. First, structuring the data contained in scientific papers by means of NE recognition techniques and then extracting text data from the disambiguated web resources (previously gathered by the NE recognition process). A new bag of words is used for the classification process, which uses the state of the art of the text mining techniques. Usually a scientific conference is a venue where people discuss and meet each other. With the advent of several social media services, such as Twitter, Slideshare, Flickr, a remarkable number of web contents is created and collected. The third Linked Data application detailed in this dissertation went beyond the actual limitations of current web services: the data heterogeneity. Data

aggregation and visualization is the key actor of this field which is supported by a real use case: the ISWC2011 venue.

Currently, significant research efforts have been devoted to make the Web a space where structured information is stored in reliable and always accessible open repositories. In the next years, the increasing of the availability of legacy data (mainly documents written according to the natural language) will boost the importance to extract and to classify meaningful data especially for improving its retrieval. Moreover, the Web is becoming also a space where entities are defined and agreed by large communities. This process, described as crowdsourcing, defines entities which univocally describe real world objects. Hence, the data extracted from the text will be identified by web resources. This dissertation addressed some of these research insights showing significant results. By the way, we are confident that a convergence of Natural Language Processing and Semantic Web research contributions will guide the research to go further the solutions proposed by this dissertation.

Bibliography

[1] E. Alfonseca, S. Manandhar, An Unsupervised Method for General Named Entity Recognition And Automated Concept Discovery, in: 1^{st} International Conference on General WordNet, 2002.

[2] E. P. Andy Seaborne, SPARQL query language for RDF, http://www.w3.org/TR/rdf-sparql-query (January 2008).

[3] M. Asahara, Y. Matsumoto, Japanese Named Entity extraction with redundant morphological analysis, in: International Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL'03), Edmonton, Canada, 2003, pp. 8–15.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A Nucleus for a Web of Open Data, in: 6^{th} International Semantic Web Conference (ISWC'07), Busan, South Korea, 2007, pp. 722–735.

[5] V. Basili, G. Caldiera, D. H. Rombach, The goal question metric approach, in: J. Marciniak (ed.), Encyclopedia of Software Engineering, Wiley, 1994.

[6] M. Bergman, Advantages and myths of rdf (April 2009).
URL http://www.mkbergman.com/?p=483

[7] T. Berners-Lee, Giant global graph (February 2007).
URL http://dig.csail.mit.edu/breadcrumbs/node/215

[8] T. Berners-Lee, Linked Data (Juin 2009).
URL http://www.w3.org/DesignIssues/LinkedData

[9] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax (August 1998).
URL http://www.ietf.org/rfc/rfc2396.txt

[10] A. R. Bharambe, M. Agrawal, S. Seshan, Mercury: supporting scalable multi-attribute range queries, Computer Communication Review 34 (4) (2004) 353–366.

[11] D. Bikel, S. Miller, R. Schwartz, R. Weischedel, Nymble: a high-performance learning name-finder, in: 5^{th} International Conference on Applied Natural Language Processing, Washington, USA, 1997, pp. 194–201.

[12] P. Biron, A. Malhotra, Xml schema part 2: Datatypes second edition - W3C Recommendation (October 2004).

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

[13] C. Bizer, T. Heath, T. Berners-Lee, Linked Data - The Story So Far, International Journal on Semantic Web and Information Systems 5 (3) (2009) 1–22.

[14] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, DBpedia - A crystallization point for the Web of Data, Web Semantics: Science, Services and Agents on the World Wide Web 7 (3) (2009) 154–165.

[15] U. Bojars, B. J. G., V. Peristeras, G. Tummarello, S. Decker, Interlinking the Social Web with Semantics, IEEE Intelligent Systems 23 (3) (2008) 29–40.

[16] A. Bonifati, P. K. Chrysanthis, A. M. Ouksel, K.-U. Sattler, Distributed databases and peer-to-peer databases: past and present, SIGMOD Record 37 (2008) 5–11.

[17] A. Borthwick, J. Sterling, E. Agichtein, R. Grishman, NYU: Description of the MENE Named Entity System as Used in MUC-7, in: 7^{th} Message Understanding Conference (MUC-7), 1998.

[18] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, Extensible Markup Language (XML) 1.0 fifth edition - W3C Recommendation (November 2008).
URL http://www.w3.org/TR/REC-xml

[19] D. Brickley, R. Guha, RDF Vocabulary Description Language 1.0: RDF Schema - W3C Recommendation (February 2004).
URL http://www.w3.org/TR/rdf-schema

[20] J. Broekstra, A. Kampman, F. van Harmelen, Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, John Wiley & Sons, Ltd, 2003, pp. 71–89.

[21] M. Cai, M. Frank, RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network, in: 13^{th} International conference on World Wide Web, 2004, pp. 650–657.

[22] M. Cai, M. Frank, J. Chen, P. Szekely, MAAN: A Multi-Attribute Addressable Network for Grid Information Services, IEEE/ACM International Workshop on Grid Computing 0 (2003) 184.

[23] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson, Jena: implementing the semantic web recommendations, in: 13^{th} International World Wide Web conference on Alternate track papers & posters, ACM, New York, NY, USA, 2004, pp. 74–83.

[24] S. Choudhury, J. G. Breslin, Extracting Semantic Entities and Events from Sports Tweets, in: Making Sense of Microposts (#MSM2011), 2011.

[25] A. M. Cohen, Optimizing feature representation for automated systematic review work prioritization, in: Annual Symposium of the American Medical Informatics Association: AMIA, 2008, pp. 121–125.

[26] A. M. Cohen, W. R. Hersh, K. Peterson, P. Y. Yen, Reducing workload in systematic review preparation using automated citation classification, Journal

of the American Medical Informatics Association : JAMIA 13 (2) (2006) 206–219.

[27] J. R. Cowie, W. G. Lehnert, Information extraction, ACM Communication 39 (1) (1996) 80–91.

[28] R. Cyganiak, A. Jentzsch, Linking Open Data cloud diagram (September 2011). URL http://lod-cloud.net

[29] J. Davies, M. Lytras, A. P. Sheth., Semantic-Web-Based Knowledge Management, IEEE Internet Computing 11 (5) (2007) 14–16.

[30] P. Di Nunzio, F. Di Gregorio, G. Rizzo, A. Servetti, Reliable SPARQL queries with consistent results over P2P-shared RDF storage, International Journal of Web Applications 2 (3) (2010) 151–163.

[31] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext Transfer Protocol - HTTP/1.1 (June 1999). URL http://www.w3.org/Protocols/rfc2616/rfc2616.html

[32] R. T. Fielding, R. N. Taylor, Principled design of the modern web architecture, ACM Transaction Interneternet Technology 2 (2002) 115–150.

[33] J. L. Fleiss, Measuring nominal scale agreement among many raters, Psychological Bulletin 76 (5) (1971) 378–382.

[34] A. Francois, R. Nevatia, J. Hobbs, R. Bolles, J. Smith, VERL: an ontology framework for representing and annotating video events, IEEE MultiMedia 12 (4) (2005) 76–86.

[35] M. J. Freedman, K. Lakshminarayanan, S. Rhea, I. Stoica, Non-transitive connectivity and DHTs, in: 2^{th} Conference on Real, Large Distributed Systems, 2005, pp. 55–60.

[36] O. Galibert, S. Rosset, C. Grouin, P. Zweigenbaum, L. Quintard, Structured and extended named entity evaluation in automatic speech transcriptions, in: Proceedings of 5th International Joint Conference on Natural Language Processing, Chiang Mai, Thailand, 2011, pp. 518–526.

[37] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, D. Suciu, What Can Peer-to-Peer Do for Databases, and Vice Versa?, in: International Workshop on the Web and Databases (WebDB 2001), 2001.

[38] R. Grishman, B. Sundheim, Message Understanding Conference-6: a brief history, in: 16^{th} International Conference on Computational linguistics (COLING'96), Copenhagen, Denmark, 1996, pp. 466–471.

[39] R. Grishman, B. Sundheim, Message Understanding Conference-6: a brief history, in: 16^{th} International Conference on Computational linguistics (COLING'96), Copenhagen, Denmark, 1996, pp. 466–471.

[40] W. O. W. Group, OWL 2 Web Ontology Language Document Overview - W3C Recommendation (October 2009). URL http://www.w3.org/TR/owl2-overview

[41] T. R. Gruber, A translation approach to portable ontology specifications, Knowledge Acquisition 5 (1993) 199–220.

[42] O. Hartig, R. Heese, The SPARQL Query Graph Model for Query Optimization, in: 4^{th} European conference on The Semantic Web: Research and Applications (ESWC'07), 2007, pp. 564–578.

[43] J. A. Hendler, D. L. McGuinness, The DARPA Agent Markup Language, IEEE Intelligent Systems 15 (2000) 67–73.

[44] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, G. Weikum, Robust Disambiguation of Named Entities in Text, in: Conference on Empirical Methods in Natural Language Processing, 2011, pp. 782–792.

[45] M. Hollander, D. A. Wolfe, Nonparametric Statistical Methods, John Wiley and Sons, 1973.

[46] P. Hovareshti, J. S. Baras, Consensus Problems on Small World Graphs: A Structural Study, in: International Conference on Complex Systems, 2006.

[47] H. Ji, R. Grishman, Data selection in semi-supervised learning for name tagging, in: Workshop on Information Extraction Beyond The Document, Sydney, Australia, 2006, pp. 48–55.

[48] M. Jorgensen, M. Shepperd, A systematic review of software development cost estimation studies, Software Engineering, IEEE Transactions on 33 (1) (2007) 33–53.

[49] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, D. Lewin, Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web, in: 20^{th} annual ACM symposium on Theory of computing, 1997, pp. 654–663.

[50] A. Kibriya, E. Frank, B. Pfahringer, G. Holmes, Multinomial Naive Bayes for Text Categorization Revisited, Lecture Notes in Computer Science 3339 (2005) 235–252.

[51] A. Kiryakov, B. Popov, I. Terziev, D. Manov, D. Ognyanoff, Semantic annotation, indexing, and retrieval, Journal of Web Semantics: Science, Services and Agents on the World Wide Web 2 (1) (2004) 49–79.

[52] B. Kitchenham, Procedures for performing systematic reviews, Tech. rep. (2004).

[53] J. Kleinberg, The small-world phenomenon: an algorithm perspective, in: 3^{th} annual ACM symposium on Theory of computing, 2000, pp. 163–170.

[54] T. Klingberg, R. Manfredi, Gnutella Protocol Development (June 2002).
URL http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[55] G. Klyne, J. J. Carroll, Resource Description Framework (RDF): Concepts and Abstract Syntax - W3C Recommendation (February 2004).
URL http://www.w3.org/TR/rdf-concepts

[56] S. Kulkarni, A. Singh, G. Ramakrishnan, S. Chakrabarti, Collective annotation of Wikipedia entities in Web text, in: 15^{th} ACM International Conference on Knowledge Discovery and Data Mining (KDD'09), Paris, France, 2009, pp. 457–466.

[57] A. M. W. Li, Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons, in: 7^{th} International Conference on Natural Language Learning at HLT-NAACL (CONLL'03), Edmonton, Canada, 2003, pp. 188–191.

[58] E. Liarou, S. Idreos, M. Koubarakis, Evaluating conjunctive triple pattern queries over large structured overlay networks, in: 5^{th} International Semantic Web Conference (ISWC), 2006, pp. 399–413.

[59] X. Liu, R. Troncy, B. Huet, Using Social Media to Identify Events, in: 3^{rd} Workshop on Social Media (WSM'11), Scottsdale, Arizona, USA, 2011.

[60] G. S. Manku, M. Bawa, P. Raghavan, V. Inc, Symphony: Distributed hashing in a small world, in: 4^{th} USENIX Symposium on Internet Technologies and Systems, 2003, pp. 127–140.

[61] F. Manola, E. Miller, RDF Primer - W3C (November 2004).
URL http://www.w3c.org/TR/rdf-primer

[62] E. MartÃnez, O. Celma, M. Sordo, B. De Jong, X. Serra, Extending the folksonomies of freesound.org using content-based audio analysis, in: Sound and Music Computing Conference, 2009.

[63] S. Matwin, A. Kouznetsov, D. Inkpen, O. Frunza, P. O'Blenis, A new algorithm for reducing the workload of experts in performing systematic reviews., Journal of the American Medical Informatics Association : JAMIA 17 (4) (2010) 446–453.

[64] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, DBpedia Spotlight: Shedding Light on the Web of Documents, in: 7^{th} International Conference on Semantic Systems (I-Semantics), 2011.

[65] S. Milgram, The small world problem, Psychology Today, 1967.

[66] D. Milne, I. H. Witten, Learning to link with Wikipedia, in: 17^{th} ACM International Conference on Information and Knowledge Management (CIKM'08), Napa Valley, California, USA, 2008, pp. 509–518.

[67] K. Möller, T. Heath, S. Handschuh, J. Domingue, Recipes for Semantic Web dog food - The ESWC and ISWC metadata projects, in: 6^{th} International Semantic Web Conference (ISWC'07), Busan, Korea, 2007, pp. 802–815.

[68] E. Mudu, L. Schiatti, G. Rizzo, A. Servetti, Zenaminer: driving the SCORM standard towards the Web of Data, in: ESWC'11) Workshop on eLearning Approaches for the Linked Data Age (Linked Learning 2011), Heraklion, Greece, 2011.

[69] D. Nadeau, S. Sekine, A survey of named entity recognition and classification, Lingvisticae Investigationes 30 (1) (2007) 3–26.

[70] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, T. Risch, EDUTELLA: a P2P networking infrastructure based on RDF, in: 11^{th} International conference on World Wide Web, 2002, pp. 604–615.

[71] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, A. Löser, Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks, in: 12^{th} International conference on World Wide Web, 2003, pp. 536–543.

[72] T. Neumann, G. Weikum, Scalable join processing on very large rdf graphs, in: 35^{th} International conference on Management of data (SIGMOD), 2009, pp. 627–640.

[73] R. Olfati-Saber, R. Murray, Consensus problems in networks of agents with switching topology and time-delays, IEEE Transactions on Automatic Control 49 (9) (2004) 1520–1533.

[74] M. F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.

[75] D. Raggett, HTML Slidy: Slide Shows in HTML and XHTML (December 2011).
URL http://www.w3.org/Talks/Tools/Slidy2

[76] L. Rau, Extracting company names from text, in: 7^{th} IEEE Conference on Artificial Intelligence Applications, vol. i, 1991, pp. 29–32.

[77] J. D. M. Rennie, L. Shih, J. Teevan, D. R. Karger, Tackling the Poor Assumptions of Naive Bayes Text Classifiers, in: 20^{th} International Conference on Machine Learning, 2003, pp. 616–623.

[78] L. Restagno, V. Akkermans, G. Rizzo, A. Servetti, A Semantic Web Annotation Tool for a Web-Based Audio Sequencer, Lecture Notes in Computer Science 6757 (2011) 289–303.

[79] G. Rizzo, F. Di Gregorio, P. Di Nunzio, A. Servetti, J. C. De Martin, A Peer-to-Peer Architecture For Distributed And Reliable RDF Storage, in: 1^{th} International Conference on Networked Digital Technologies (NDT'09), 2009, pp. 94–99.

[80] G. Rizzo, B. Meirone, P. Di Nunzio, F. Di Gregorio, Distributed Semantic Video Tagging for Peer-to-Peer Authoring System, in: (DEXA'10) Workshop on Web Semantics and Information Processing (WebS '10), 2010, pp. 183–187.

[81] G. Rizzo, F. Morando, J. C. De Martin, Open Data: la piattaforma di dati aperti per il Linked Data, Informatica e Diritto 20 (1-2) (2011) 493–511.

[82] G. Rizzo, R. Troncy, NERD: A Framework for Evaluating Named Entity Recognition Tools in the Web of Data, in: 10^{th} International Semantic Web Conference (ISWC'11), Demo Session, Bonn, Germany, 2011, pp. 1–4.

[83] G. Rizzo, R. Troncy, NERD: Evaluating Named Entity Recognition Tools in the Web of Data, in: Workshop on Web Scale Knowledge Extraction (WEKEX'11), Bonn, Germany, 2011, pp. 1–16.

[84] S. Sekine, NYU: Description of the Japanese NE system used for MET-2, in: 7^{th} Message Understanding Conference (MUC-7), 1998.

[85] S. Sekine, C. Nobata, Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy, in: 4^{th} International Conference on Language Resources and Evaluation (LREC'04), Lisbon, Portugal, 2004.

[86] J. Sim, C. Wright, The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements, Physical Therapy 85 (3) (2005) 257–268.

[87] H. S. Steffen Staab, Semantic Web and Peer-to-Peer, Springer, 2006.

[88] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, D. Reynolds, SPARQL basic graph pattern optimization using selectivity estimation, in: 17^{th} International Conference on World Wide Web (WWW'08), 2008, pp. 595–604.

[89] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transactions on Networking 11 (1) (2003) 17–32.

[90] F. Suchanek, G. Kasneci, G. Weikum, Yago: a Core of Semantic Knowledge, in: 16^{th} International Conference on World Wide Web (WWW'07), 2007, pp. 697–706.

[91] F. Tomassetti, G. Rizzo, A. Vetro', L. Ardito, M. Torchiano, M. Morisio, Linked Data approach for selection process automation in Systematic Reviews, in: 15^{th} Annual Conference on Evaluation and Assessment in Software Engineering (EASE'11), Durham City, United Kingdom, 2011, pp. 31–35.

[92] R. Troncy, B. Malocha, A. Fialho, Linking Events with Media, in: 6^{th} International Conference on Semantic Systems (I-SEMANTICS'10), Graz, Austria, 2010.

[93] D. J. Watts, S. H. Strogatz, Collective dynamics of "small-world" networks, Nature 393 (6684) (1998) 440–442.

[94] Y. Wilks, C. Brewster, Natural Language Processing as a Foundation of the Semantic Web, Found. Trends Web Sci. 1 (2009) 199–327.

[95] L. Xiao, S. Boyd, S. Lai, A space-time diffusion scheme for peer-to-peer least-squares estimation, in: 5^{th} International Conference on Information Processing in Sensor Networks (IPSN'06), 2006, pp. 168–176.

[96] Z. Zha, T. Mei, Z. Wang, X. Hua, Building a comprehensive ontology to refine video concept detection, in: International Workshop on multimedia information retrieval, 2007, pp. 227–236.