Università degli Studi di Milano

DIPARTIMENTO DI INFORMATICA
Scuola di Dottorato in Informatica – XXV ciclo

PhD thesis

Extending Linked Open Data resources exploiting Wikipedia as source of information

Student:
Alessio Palmero Aprosio
Matricola R08605

Advisor:
Prof. Ernesto Damiani

Co-Advisors:
Alberto Lavelli
Claudio Giuliano

Year 2011–2012

Abstract

DBpedia is a project aiming to represent Wikipedia content in RDF triples. It plays a central role in the Semantic Web, due to the large and growing number of resources linked to it. Currently, the information contained in DBpedia is mainly collected from Wikipedia infoboxes, a set of attribute-value pairs that represent a summary of the Wikipedia page. The extraction procedure requires to manually map Wikipedia infoboxes into the DBpedia ontology.

Thanks to crowdsourcing, a large number of infoboxes in the English Wikipedia has been mapped to the corresponding classes in DBpedia. Subsequently, the same procedure has been applied to other languages to create the localized versions of DBpedia. However, *(i)* the number of accomplished mappings is still small and limited to most frequent infoboxes, as the task is done manually by the DBpedia community, *(ii)* mappings need maintenance due to the constant and quick changes of Wikipedia articles, and *(iii)* infoboxes are manually compiled by the Wikipedia contributors, therefore in more than 50% of the Wikipedia articles the infobox is missing. As a demonstration of these issues, only 2.35M Wikipedia pages are classified in the DBpedia ontology (using a class different from the top-level `owl:Thing`), although the English Wikipedia contains almost 4M pages. This shows a clear problem of coverage, and this issue is even worse in other languages (like French and Spanish).

The objective of this thesis is to define a methodology to increase the coverage of DBpedia in different languages, using various techniques to reach two different goals: automatic mapping and DBpedia dataset completion. A key aspect of our research is multi-linguality in Wikipedia: we bootstrap the available information through cross-language links, starting from the available mappings in some pivot languages, and then extending the existing DBpedia datasets (or create new ones from scratch) comparing the classifications in different languages. When the DBpedia classification is missing, we train a supervised classifier using DBpedia as training. We also use the Distant Supervision paradigm to extract the missing properties directly from the Wikipedia articles.

We evaluated our system using a manually annotated test set and some existing DBpedia mappings excluded from the training. The results demonstrate the suitability of the approach in extending the DBpedia resource. Finally, the resulting resources are made available through a SPARQL endpoint and as a downloadable package.

Acknowledgments

This thesis would not have been possible without the help and the guidance of some valuable people, who gave me help and assistance in many ways.

First of all, I would express my gratitude to my advisors Alberto Lavelli and Claudio Giuliano, who supported me with patience and enthusiasm. I thank Bernardo Magnini, head of the Human Language Technology unit at Fondazione Bruno Kessler, who accepted me despite my particular academic situation. I also wish to thank all the members of the HLT group for being always present both as inspiring colleagues and precious friends.

I am also thankful to Prof. Silvio Ghilardi and Prof. Ernesto Damiani, from University of Milan, for their availability and collaboration.

I would like to thank Volha Bryl, Philipp Cimiano and Alessandro Moschitti for agreeing to be my thesis referees and for their valuable feedback.

I gratefully thank Elena Cabrio, Julien Cojan and Fabien Gandon from INRIA (Sophia Antipolis) for letting me be a part of their research work.

Outside research, I thank all friends and flatmates I met during these years, for adding precious moments to my everyday life in Trento.

Finally, I thank my parents and family for their constant and loving support throughout all my studies.

Contents

List of Figures

List of Tables

1
Introduction

1.1 The context

Until 2001, data on the web was a collection of unstructured or semi-structured documents located on different servers, and interconnected by means of hyperlinks. Some structured datasets were available for download, but they employed different schemas, and substantial effort was needed to use them jointly. In 2001, Tim Berners-Lee formalized these limitations of the WWW in the perspective of automatic processing: until then, the web was intended mostly for humans or for very specialized agents [BLHL$^+$01]. In addition to the classic "Web of documents", he imagined a technology stack to support a "Web of data", the sort of data you find in databases. The term "Semantic Web" (SW) refers to that vision of the Web of linked data.

In 2006, another document by Berners-Lee [BL06] noted that the SW is also about linking data to each other. He published a set of principles to be observed by the data contributors to the SW (see Section 2.2). In his presentation he stated that Linked Data is "the Semantic Web done right".[1] According to the official document published by W3C "*Linked Data lies at the heart of what Semantic Web is all about: large scale integration of, and reasoning on, data on the Web.*"[2]

Pretty soon, scientific communities and public bodies followed the ideas expressed by Berners-Lee turning the WWW from a collection of interlinked documents to a repository where both documents and data are stored and linked to each other. The first project in such direction started in 2006, at Freie Universität in Berlin. Chris Bizer, an academic researcher, noticed that in Wikipedia documents "there are little squares, little boxes", called *infoboxes*. "And in most information boxes, there's data. So he wrote a program to take the data, extract it from Wikipedia,

[1] http://www.w3.org/2008/Talks/0617-lod-tbl/
[2] http://www.w3.org/standards/semanticweb/data

and put it into a blob of linked data on the web" [BL09]. DBpedia[3] – this is the name of the project – is still compiled by a vibrant community of volunteers from all over the world and it is commonly considered the hub of the growing LOD community.

Nowadays (2013), there exist more than 300 large-scale knowledge bases (KB) in the LOD cloud. Besides DBpedia, other relevant examples include FreeBase,[4] owned by Google and compiled using crowdsourcing, YAGO,[5] created using various techniques that range from crowd sourcing to handcrafted rules, and Wikidata[6], a semi-structured database intended to provide a common source of certain data types which can be used by Wikimedia projects such as Wikipedia.

On the one hand, these KBs are collected using data from Wikipedia; on the other hand, a growing number of institutions around the world has started to release their information using open formats and following the LOD principles defined by Berners-Lee (see Chapter 2). For example, governments such as the United States[7], Italy[8], and many more have started to put public data on the web available through APIs or SPARQL endpoints.

In October 2012, Europeana[9] (Europe's digital library, connecting over 2,000 providers and containing 23 million items in its dataset) transformed a large subset of its data into linked data [HI11].

While more and more semantic data is published on the Web, the question of how Internet users can access this huge amount of knowledge becomes of crucial importance. The availability of generalist and domain-specific free data, and this particular need to access it, have led to the rise and expansion of the so called *computational journalism*. Using and merging open information, journalists in the web era are able to tell stories that were unimaginable some years ago [Fam11]. The Guardian, one of the most widespread newspapers in UK and the third online newspaper for page views in the world, has a blog section in its website dedicated to data journalism.[10]

Apart from journalistic purposes, LOD can be useful for everyday research in various fields, even for end-users. For this reason, Natural Language Processing (NLP) interfaces have received wide attention, as users can obtain complex information from data on the web in an intuitive fashion and, at least in principle, in their own language. Question Answering (QA) is the scientific discipline that tries to automatically answer questions expressed using natural language. The availability of a huge amount of structured data brings new blood to this theme, and in the last

[3]http://dbpedia.org/About
[4]http://www.freebase.com/
[5]http://www.mpi-inf.mpg.de/yago-naga/yago/
[6]http://www.wikidata.org/
[7]http://www.data.gov/
[8]http://www.dati.gov.it/
[9]http://pro.europeana.eu/linked-open-data
[10]http://www.theguardian.com/media/data-journalism

year several systems using different approaches have been proposed to the scientific community (see Section 3.5).

In this context, the availability, precision and completeness of the LOD resources are critical points for research studies and development of tools that make use of them.

We investigate such direction, trying to identify and tackle the issues in DBpedia, one of the most important and generalist LOD resources in the Web.

In addition, amount of Web users speaking languages different from English has grown in recent years. As a consequence, the Internet is turning more and more into a multilingual platform in which agents from different languages and cultural backgrounds collaborate, consuming and producing information at a scale without precedent [BCCH13]. This is why in our work we always consider DBpedia as a multi-lingual resource and we concentrate our effort on more than 30 languages.

1.2 DBpedia

The main reference for our work is the DBpedia project [ABK+07]. Started in 2006, it aims at building a large-scale knowledge base semi-automatically extracted from Wikipedia. Due to the large and constantly increasing number of links from and to other data sources, DBpedia continues to gain popularity and today it plays a central role in the development of the Web of Data.

In release 3.8, the English version of the DBpedia knowledge base describes around 4 million objects, out of which 2.35 million are classified in a consistent ontology.

The process of building DBpedia consists of different steps.

The ontology. First of all, DBpedia releases an ontology that includes a taxonomy (classes) enriched by relationships (properties) between pairs of concepts or between concepts and scalar values. Examples of classes are `Person`, `Place`, `Event`; examples of properties are `height`, `birthDate`, `spouse`. Version 3.8 of the ontology contains more than 350 classes and 1,700 properties.

The mappings. Wikipedia infobox names and properties (see Section 2.5.1) are manually mapped to classes and properties in the DBpedia ontology. These mappings are hand-generated by the DBpedia communities around the world. For example, there is a mapping between *Infobox Australian road* in Wikipedia and the class `Road` in DBpedia. The attributes of *Infobox Australian road*, such as *direction*, *exits*, etc. are mapped to the corresponding DBpedia properties (`routeDirection` and `routeJunction`, respectively). Since this work currently needs manual effort and infoboxes are distributed according the Zipf's law [SHB+12], DBpedia facilitates the operation by releasing the list of available Wikipedia infoboxes sorted by frequency:[11] most frequent items can be mapped first and a small number of mappings would result in a big number of articles added to the resource.

Extraction framework. The DBpedia community releases an open-source tool that, given a Wikipedia dump and the list of mappings as input, automatically adds every page in Wikipedia that contains an infobox mapped to a specific class to such class. For example, the Wikipedia article `Hume Highway` contains the *Infobox Australian road*. The framework applies the mapping between *Infobox Australian road* and the DBpedia class `Road`, by assigning `Hume Highway` to such class.

SPARQL endpoint. In addition, the resulting resource is available as a SPARQL endpoint[12] that can be used for complex queries over Wikipedia data.

Version 3.8 of DBpedia is available in 111 languages, 22 of which are generated from mappings of classes and properties in the original language.[13]

The most recent release (3.9) has been made available on 17th September 2013, and contains some improvements (see Section 2.5.2). Since our work has been done between August 2012 and August 2013, all the results in this thesis refer to version 3.8 of DBpedia, the last available in that period of time.

1.3 The problem

During our research, we took in consideration the most critical part in the process of building DBpedia, that is the hand-created mappings between infoboxes and the DBpedia ontology. We deal with this problem from two different points of view: first, the classification of a Wikipedia article in the DBpedia ontology when the information is not adequate to apply mappings (for example, when the infobox is missing); second, the expansion of DBpedia mappings to other languages.

[11]See, for example, such a list relative to the English Wikipedia: http://mappings.dbpedia.org/server/statistics/en/

[12]http://dbpedia.org/sparql

[13]http://wiki.dbpedia.org/Downloads38

1.3.1 Coverage expansion

At the time of starting the work reported in this thesis, the last available version of DBpedia, 3.8, covers around 4M entities in its English chapter, the same number of the articles included in the English Wikipedia. However, this apparently good result is due to the fact that, when a user-provided mapping is not available, an article in Wikipedia is by default mapped to the top-level class `owl:Thing`. In fact, only 2.35M pages are mapped to classes different from `owl:Thing`. From here on, when we speak about coverage we will refer to these pages only.

There is a lot of variability in the names used for infoboxes and infobox attributes. Thus, it often happens that two or more infoboxes might be mapped to the same class, but none of them is included in DBpedia because their individual frequency is too small. Moreover, the DBpedia ontology often has classes that do not have a corresponding Wikipedia infobox. For example, the class `Actor` does not have a generic infobox in the English Wikipedia. However, the English Wikipedia provides some very specific infoboxes mapped to subclasses of `Actor`, such as *Chinese-language_singer_and_actor*. In this way, Bruce Lee is present in the database as an `Actor`, while other very famous actors like Clint Eastwood and Brad Pitt are not, clearly an undesirable result. Finally, some articles do not have an infobox, even if Wikipedia provides one for the purpose. This may happen because the user who writes that article does not know how to specify it, or simply does not know that infoboxes exist.

Concerning properties, in the previous section we said that DBpedia uses the infobox attributes to populate them. Similarly to the case of classes, it may happen that in a Wikipedia article some infobox attributes are missing, for example because the user forgot to populate them. There are projects aiming to extract properties from some structured parts of the page different from infoboxes. For example, YAGO exploits categories [SKW07]. However, such approach is feasible only for a small number of attributes (for example the Wikipedia page `Barack Obama` is included in the *1961 births* category, from which it can be inferred that Obama's birth year is 1961). Therefore, the only place where we can find the values for the whole DBpedia set of properties (over 1,500 in version 3.8) when the corresponding infobox attribute is missing in the article, that can be exploited using NLP tools.

1.3.2 Automatic mapping

Until 2011, localized DBpedia datasets included data from non-English Wikipedia pages only if there existed an equivalent English page. However, since there are many pages in the non-English Wikipedia editions without an equivalent English page, relying only on English Wikipedia pages had the negative effect that DBpedia did not contain data for these entities. More recently, other contributors around the world have joined the project to create localized and interconnected versions of the resource. The goal is to populate the same ontology used by the English project, taking articles from editions of Wikipedia in different languages. These new localised versions of DBpedia required the same effort already used for the English one: infoboxes and attributes in the localized Wikipedia are manually mapped to the corresponding classes and properties in the DBpedia ontology. Several research groups around the world started to join the project and take charge of this manual task. As the mappings have to follow syntactic constraints due to the MediaWiki language, requested to compile DBpedia mappings, a community user spends some minutes for each mapping. If a corresponding class/property does not exist yet in the ontology, the job can take longer. As each language may have hundreds of infoboxes, this effort can be heavy for the community.

In addition, the granularity of Wikipedia infoboxes is not consistent across language editions, and often there is not a clear one-to-one mapping between those infoboxes [RLN13]. For example, the 10 most frequent infoboxes in the English Wikipedia cover around 900K articles (21% of the total), in French the same amount of infoboxes cover around 200K pages (14%), and in Italian the 10 most frequent infoboxes even reach 550K pages (55%): this difference is due to the existence – in the Italian Wikipedia – of two infoboxes (`Bio` and `Divisione amministrativa`), used for biographies and locations, which do not have an analogous in English and French.

1.4 The solution

In this thesis, we want to deal with the current limitations of DBpedia outlined above, by extending the coverage of DBpedia over Wikipedia and releasing the resulting resource.

A key aspect of our approaches is the multi-linguality of Wikipedia, with which we can automatically extend the DBpedia resource and further bootstrap its coverage.

First, we focus on the problem of automatically mapping infoboxes and infobox attributes to classes and properties into the DBpedia ontology both for extending the coverage of the existing localized versions (e.g., Italian, Spanish) and for building from scratch versions for languages not yet covered (e.g., Swedish, Norwegian, Ukranian). This task is currently performed using crowdsourcing and there are no published attempts to perform it automatically. Related work has exclusively focused on developing automatic approaches to the discovery of mappings between different Wikipedia editions; these results can be used to automatize the mapping process, though this solution is highly prone to changes in Wikipedia, a noticeable drawback considering how fast edits are made. We propose an instance-based approach, that exploits the redundancy of Wikipedia in different editions (languages). In the particular case of property mappings, we assume that an infobox attribute and an ontology property are equivalent if their instantiated values are similar. Specifically, the mapping is cast as a binary classification task in which instances are infobox attribute/ontology property pairs extracted from versions of Wikipedia and DBpedia in different languages and cross-language links are used to represent the instances in a unified space. Attributes and properties are compared using their values taking into account their types (i.e., date, integer, object, etc.). For Wikipedia infobox attributes, the type is calculated; for DBpedia properties, the type is given by the ontology. We show that this approach is robust with respect to rapid changes in Wikipedia.

Second, we extend the coverage on classes and properties when the information is missing (for example, when there is not any infobox in the Wikipedia page). For classes, we train a supervised kernel-based classifier, while for properties, we use the distant supervision paradigm to extract the missing information directly from the Wikipedia article, using a Relation Extraction tool. In both cases, we use the information already present in DBpedia as training data.

Finally, we release the resulting resources as DBpedia mappings or RDF triples. The datasets are licensed under the terms of the Creative Commons Attribution-ShareAlike License[14], and are available for download on the project website http://www.airpedia.org/.

1.5 Interacting with the Semantic Web

To enhance user's interactions with the web of data, query interfaces providing a flexible mapping between natural language expressions, and concepts and relations in structured knowledge bases are becoming particularly relevant.

In the last part of this thesis, we present a Question Answering (QA) system, called QAKiS, that allows end users to submit a query to an RDF triple store in English and obtain the answer in the same language, hiding the complexity of the non intuitive formal query languages (see Section 2.3.3) involved in the resolution process.

The project, a joint work with INRIA (Sophia Antipolis), addresses the problem of question interpretation as a relation-based match, where fragments of the user question are matched to binary relations of the triple store, using relational textual patterns automatically collected. In the current implementation, the relational patterns are automatically extracted from Wikipedia, while DBpedia is the RDF data set to be queried using a natural language interface.

A demo of the system is available at http://qakis.org/.

[14]http://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License

1.6 Contributions

In the following we provide the list of contributions of the thesis. It should be noted that part of this work have already been peer-reviewed by the scientific community.

1.6.1 DBpedia expansion

The following table summarizes the different steps of the effort on DBpedia expansion, with references to the papers that have been already published.

	Automatic mapping for new languages	Instance-based classification
Classes	①	②
Properties	③	④

① Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Automatic Mapping of Wikipedia Templates for Fast Deployment of Localised DBpedia Datasets. In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*, 2013 *(Acceptance rate 39%)*

② Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Towards an Automatic Creation of Localized Versions of DBpedia. In *Proceedings of the 12th International Semantic Web Conference*, 2013 *(Acceptance rate 21%)*

③ Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Automatic expansion of DBpedia exploiting Wikipedia cross-language information. In *Proceedings of the 10th Extended Semantic Web Conference*, 2013 *(Acceptance rate 26%)* *Best paper nominee*

④ Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Extending the Coverage of DBpedia Properties using Distant Supervision over Wikipedia. In *Proceedings of the 1st Workshop on NLP & DBpedia (ISWC)*, 2013

1.6.2 Question Answering

The following publications refer to QAKiS.

5. Elena Cabrio, Alessio Palmero Aprosio, Julien Cojan, Bernardo Magnini, Fabien Gandon, and Alberto Lavelli. QAKiS at QALD-2. In *Proceedings of the Interacting with Linked Data (ILD) Workshop at ESWC 2012*, Heraklion, Greece, 2012

6. Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. QAKiS: an open domain QA system based on relational patterns. In Birte Glimm and David Huynh, editors, *International Semantic Web Conference (Posters & Demos)*, volume 914 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012

7. Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, and Fabien Gandon. Natural language interaction with the web of data by mining its textual side. *Intelligenza Artificiale*, 6(2):121–133, 2012

1.7 Structure of the thesis

The rest of this document is structured as follows.

Chapter 2 introduces notations and concepts related to Linked Open Data, with the description of the major resources used in the following chapters.

Chapter 3 presents the most relevant work on the topics and techniques used in our work.

Chapter 4 describes some pre-processing steps on Wikipedia and DBpedia dumps, that filter unnecessary information and transform raw data into a useful format.

Chapters 5 and 6 address the problem of automatic mapping generation to create from scratch new chapters of DBpedia and to expand the existing ones. These chapters correspond to the papers (1)-(2) listed above.

Chapter 7 describes a machine learning method used to classify pages in the DBpedia ontology (paper (3) above).

Chapter 8 details the approach used to deal with the extraction of property values from the article text of a Wikipedia page (paper (4) above).

Chapter 9 describes Airpedia, the resource obtained by the previous steps. The datasets extracted using the method described in Chapters 5 and 7 are made available for download from the project website, in an open format.

Chapter 10 refers to the question answering system QAKiS, that has led to the publications (5)-(7) listed above.

2 Linked Open Data

Linked Data is a method under which structured data is published on the World Wide Web so that it can be interlinked and become more useful. Similarly to the web documents written in HTML, it is built upon the standard HTTP protocol, but rather than using it to serve web pages, it extends that protocol to provide information in machine-readable way. Similarly to the standard web hyperlinks, this paradigm enables data from different sources to be connected and queried (hence the name). Thus, Linked Data can be viewed as a global raw data space, a web of data, organized similarly to the web of documents, but in contrast to it and its display-oriented purpose, since the HTML language, actually used to build web pages, is itself entirely aimed for reading. Substantial part of the Linked Data is available for free, and constitutes the Linked Open Data (LOD).

2.1 Origins

In 1989, Tim Berners-Lee wrote a proposal[1] for an information management system and sent it to his boss, Mike Sendall. “Vague, but exciting”, was the comment that Sendall wrote on the top of the paper, allowing Berners-Lee to continue his work. The following year, it was 1990, a more formal proposal was published, outlining the principal concept and the main ideas behind the object that now we call World Wide Web (WWW), and that in 2013 is used by almost 3 billion people every day.

Despite the benefits provided by the World Wide Web (WWW), in the first period the Internet was a mere collection of documents whose purpose was readability. The HTML language, expressly created for the web, is display-oriented, confining raw data to tables, CSV dumps

[1]http://info.cern.ch/Proposal.html

or XML repositories (in the luckiest cases). Even hypertext links, traditionally used to create relationships between documents, are not sufficient to enable the computer to "understand" the information behind them. For instance, if we search an ambiguous word through a search engine (e.g., Google), we will get a jumble of results involving all the possible meanings of the given term. In fact, search engines only consider the term as it is: as long as a given page (or the text used to link to it) contains the term, it will be included in the results.

Some services, like Amazon, provided APIs that sent structured data encoded in a micro-format in a response to a structured query. However, formats of documents or API queries and responses varied from provider to provider. Another problem was the different semantics of the structured sources, e.g. two fields named "Address" in two different databases do not necessarily contain the same data. For example, one database could contain websites, while another listed human readable mailing addresses. Format heterogeneity and absence of semantics made the task of automatic accessing and processing structured data challenging as each data source had to be processed separately, with its data format and semantics taken into account. It is up to the user to understand and interpret the content of the page and select the required information.

In 2001 again Tim Berners-Lee formalized the limitations of the WWW in the perspective of automatic processing [BLHL^{+}01]. He coined the expression "Semantic Web" (SW) and proposed the idea of extending the Internet with machine readable data.

In 2006 [BL06], another document by Tim Berners-Lee noted that SW is also about linking data to each other. He proposed an idea of extending the WWW with machine-readable data, following some of the principles that has led the web to reach its popularity: decentralized structure by independent data providers, standard formats, and so on. As an example, the founder of the web provided a use-case of an automatic agent, scheduling a visit to a doctor for the mother of a hypothetical user. In order to deliver a solution, the agent would need to retrieve, understand and analyze the schedules of the doctor and the user, to take into account the distance from the user's home to the doctor's office. This can be achieved by using standard formats for knowledge representation, unambiguous identifiers for things, and ontologies describing the world in a common language. The format suggested for this purpose was RDF (Resource Description Format): a way of describing knowledge in triples, consisting of subject, predicate and object.

In the subsequent years, the scientific community accepted the idea and created a number of new resources [SHBL06]. At the same time a set of organizations, including the World Wide Web Consortium (W3C), devised the Web Ontology Language (OWL) to model knowledge, the Resource Description Format (RDF) specification to describe simple objects in triples, and various reasoners and rule exchange formats to support the inference. Some databases built for that purpose, called *triplestores*, permit storage, indexing, and retrieval of RDF triples. The query

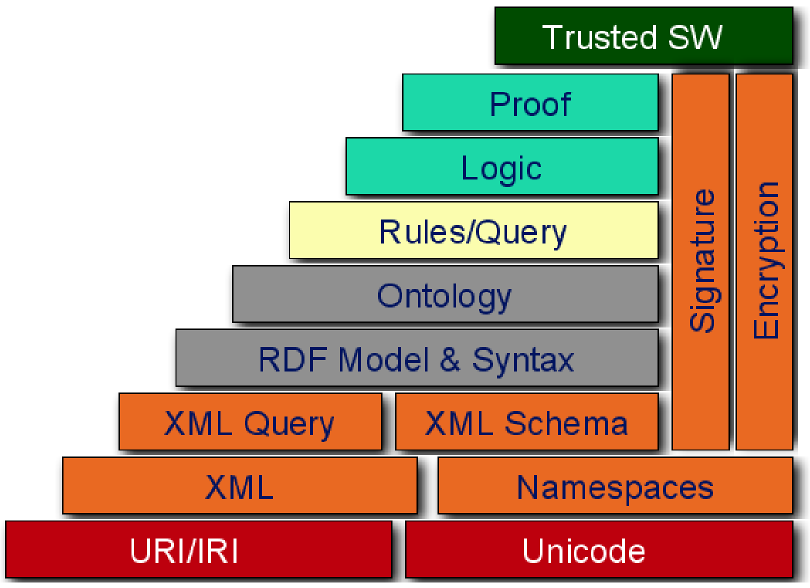


Figure 2.1: The Semantic Web stack

language designed to interact with RDF triplestores is SPARQL, designed and standardized by the W3C in 2008.

Figure 2.1[2] shows how these components are integrated into the Semantic Web (SW) architecture.

2.2 Linked Data principles

Tim Berners-Lee introduced for the first time the concept of the Linked Data (LD) in [BL06]. In his definition, LD is the web of RDF descriptions of objects ("things") interconnected by hyperlinks, and publishers must observe principles that can be summarized as follows [BL09]:

(i) Use HTTP URIs as names for things.

(ii) When someone looks up a URI, provide useful information in standard format.

(iii) Include links to other URIs, so that one can discover more things.

The first principle means that each "thing" of each dataset must have a unique name, called a Unique Resource Identifier (URI), in the world. The protocol suggested is HTTP, already widespread and understood by humans and machines.

[2]Picture taken from [S+05]

The second principle says that the URI has to be dereferenceable, that is users must have a representation of the resource by typing it in a standard web browser. In the context of traditional HTML web pages, this is the normal and obvious behavior, but there are contexts (such as XML schemas) where URIs are used as identifiers, but their addresses do not point to any description of the schema document. In addition, the information should be encoded in the Resource Description Framework (RDF) formalism (see Section 2.3.1).

Finally, the last principle means that the data providers should link their URIs to URIs in other datasets.

2.3 Linked Data in practice

In order to use Linked Data, one can try different approaches, depending on the size of the needed dataset or the required speed in processing data. For instance, if we want to do some reasoning on an entire dataset, we can download it locally; otherwise, if we only need a subset of knowledge, there are various techniques to query the resource on-the-fly without storing terabytes of data. Approaches in Chapters 5 to 8 use the first technique; the QA system QAKiS (see Chapter 10) uses the second.

In the next subsections, we will first briefly describe the RDF data format; then, we will summarize the key aspects of SPARQL, a query language designed for RDF triplestores; finally we will describe the major tools used to manage big amount of RDF data.

2.3.1 Resource Description Framework

Resource Description Framework (RDF)[3] is a standard model for encoding, exchange and use of structured metadata. In RDF, data are modeled using a paradigm consisting of *subject*, *predicate* and *object*. These expressions are also known as *triples*. Each triple describes a directed relationship between a subject and an object. The parts of the triple that are linked to a thing may be identifiers, i.e. URIs, also from other resources. Objects may also be literals, e.g. integers, strings, dates, and so on.

Table 2.1 shows a set of RDF statements extracted from DBpedia, regarding Barack Obama. Labels `dbpedia:` and `dbpedia-owl:` in the table are shortcuts to `http://dbpedia.org/resource/` and `http://dbpedia.org/ontology/` respectively (see the definition of `PREFIX` in Section 2.3.3). Note that a single property (eg. dbpedia-owl:almaMater) can take multiple values for the same subject. There are many formats used to write RDF statements (see below). For example, the first row of Table 2.1 in N-Triples would result in:

[3] `http://www.w3.org/RDF/`

Subject	Predicate	Object
dbpedia:Barack_Obama	dbpedia-owl:birthDate	1961-08-04
dbpedia:Barack_Obama	dbpedia-owl:spouse	dbpedia:Michelle_Obama
dbpedia:Barack_Obama	dbpedia-owl:almaMater	dbpedia:Columbia_University
dbpedia:Barack_Obama	dbpedia-owl:almaMater	dbpedia:Harvard_Law_School
dbpedia:Barack_Obama	dbpedia-owl:almaMater	dbpedia:Occidental_College
dbpedia:Barack_Obama	owl:sameAs	`http://www.freebase.com/m/02mjmr`
...	...	...

Table 2.1: RDF statements examples

```
<http://dbpedia.org/resource/Barack_Obama>
<http://dbpedia.org/ontology/birthDate>
"1961-08-04"^^<http://www.w3.org/2001/XMLSchema#date> .
```

where there are no URL prefixes, and `http://www.w3.org/2001/XMLSchema#date` refers to the data type used to represent dates. The final dot `.` is mandatory and marks the end of the triple.

The URIs used in the triples can be dereferenced to retrieve the alternative description of the involved entities. One of the most important predicates is owl:sameAs. Typically, different LOD datasets have different URIs (and correspondingly different descriptions) for the same thing. For example, Freebase (a LOD crowd-generated resource owned by Google, see Section 3.1.2) URI for Barack Obama is `http://www.freebase.com/m/02mjmr`. In the last line of Table 2.1, we can see how the owl:sameAs link connects the DBpedia URI to the Freebase URI. By dereferencing the two URIs, we can find two alternative descriptions related to the US President. In the LOD cloud, owl:sameAs links are the "glue" that connects the different resources to each other. By using them, we can navigate between datasets published by different providers and obtain different information about the same entity.

Alternatively, a collection of RDF statements can be seen as a multi-graph, where predicates are directed labeled edges which connect subject and object nodes.

There currently exists a number of standard RDF serializations that can be processed by the main Semantic Web processing engines. They include RDF/XML[4] serialization, Turtle,[5] N-Triples.[6]

[4]`http://www.w3.org/TR/rdf-syntax-grammar/`
[5]`http://www.w3.org/TeamSubmission/turtle/`
[6]`http://www.w3.org/TR/rdf-testcases/#ntriples`

2.3.2 Resource Description Framework in Attributes

The RDF standard just described is useful to share information in a structured way, but it is a standalone format: an RDF graph is serialised into a document that exists on its own. A web developer, who has already built pages in (X)HTML, may need to embed the salient information contained in the page in such a way that a machine can read them. There are a number of proposal how to join structured and unstructured data in the same document. Some years ago, W3C started to investigate a more "natural" way of using RDF and (X)HTML together. The result is RDFa, RDF in attributes.

In general, RDFa uses standard attributes into (X)HTML tags (such as <div> or <span>) to assign a description to specific information included between the starting part and the ending part of the tag itself. The document needs an initial xmlns declaration where the vocabulary name space is defined.

As a well-known example, the British Broadcasting Corporation, one of the largest broadcaster in the world, has started to adopt RDFa in its "programmes" portal. They first create a Programmes Ontology (PO),[7] and then they linked all the information in the programmes website to that ontology.

Each episode of each program has its own web page, with information such as airing scheduling, cast, description, and so on. For example, the episode of the BBC famous program "Panorama" entitled "Dying for a Bargain" aired on BBC One on 23rd September 2013 and on BBC News on 26th September 2013. On the episode webpage,[8] such information is shown in the "Broadcast" section, and is easily readable by a human. But how can a machine get this information? Usually, this goal is reached by parsing the (X)HTML source, effort that needs a human examination. If the code contains the RDFa labels, the machines is helped to understand the most important information in the page, without analyzing the page template used to show the information in an appealing way.

Back to the example, this is the portion of code concerning the airing dates:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-GB"
    xmlns:po="http://purl.org/ontology/po/">
...
<ul about="/programmes/b03bvmyf#programme">
    <li rel="po:broadcast" resource="/programmes/p01gsb21#broadcast">
    <p>
    <a href="/bbcone" title="BBC One">
```

[7] See http://www.bbc.co.uk/ontologies/programmes/2009-09-07.shtml for more information.
[8] http://www.bbc.co.uk/programmes/b03bvmyf

```
        <span class="title">
            BBC One
        </span>
    </a>
    <span datatype="xsd:date" property="timeline:start">
        Mon 23 Sep 2013
    </span>
    <span datatype="xsd:time" property="timeline:start">
        20:30
    </span>
    </p>
    </li>
    <li rel="po:broadcast" resource="/programmes/p01gx997#broadcast">
    <p>
    <a href="/bbcnews" title="BBC News Channel">
    <span class="title">
             BBC News Channel
    </span>
    </a>
    <span datatype="xsd:date" property="timeline:start">
             Thu 26 Sep 2013
    </span>
    <span datatype="xsd:time" property="timeline:start">
        04:30
    </span>
    </p>
    </li>
</ul>
...
</html>
```

- The example starts with the declaration of the used namespace (a list of classes/properties and their relations). In this case, `http://purl.org/ontology/po/` refers to the PO.

- The `property` attribute denotes that the information between the tags is the value of that property. In the example, start date and time of the program.

- The `datatype` attribute gives an additional information respect to the type of the stored data (integer, date, string, and so on).

2.3.3 SPARQL query language

SPARQL (SPARQL Protocol and RDF Query Language)[9] is a query language designed to retrieve and manipulate data stored in RDF format. It became a standard by the W3C in 2008, and it is one of the key technologies in the Semantic Web.

The following is an example of a query against DBpedia. It will search for all people born in Italy, and list their URIs, names and dates of birth.

```
 1 PREFIX prop: <http://dbpedia.org/property/>
 2 PREFIX owl: <http://dbpedia.org/ontology/>
 3 PREFIX dbp: <http://dbpedia.org/resource/>
 4
 5 SELECT ?person, ?name, ?bdate WHERE {
 6   ?person   a                owl:Person   .
 7   ?person   prop:birthName   ?name        .
 8   ?person   prop:birthPlace  dbp:Italy    .
 9   ?person   prop:birthDate   ?bdate       .
10 }
```

Users familiar with other SQL languages should recognize keywords like `SELECT` and `WHERE` (line 5). The keyword `PREFIX` (lines 1 to 3) specifies a shortcut for the frequently used parts of URIs, to make the rest of the query more readable. Labels starting with "?", such as `?name`, `?date` and `?person` are variables that should be bound to concrete values in a resolved query. The SPARQL keyword `a` (line 6) is a shortcut for the common predicate http://www.w3.org/1999/02/22-rdf-syntax-ns#type, giving the class of a resource.

Resolving a SPARQL query against an RDF triplestore means retrieving subsets of the graph that match the pattern expressed by the query. For example, the DBpedia (see Section 2.5.2) set of results matching the query above is listed in Table 2.2.

Remote RDF graphs can be queried through SPARQL endpoints. A SPARQL endpoint is a service that accepts SPARQL queries as input and returns triples. Such an endpoint must be conformant to the SPARQL protocol as defined in the SPROT specification.[10]

Most of large-scale knowledge triplestores have publicly available SPARQL endpoints. For example, the SPARQL endpoint of DBpedia, one of the core LOD datasets, can be accessed

[9] http://www.w3.org/TR/rdf-sparql-query/
[10] http://semanticweb.org/wiki/SPARQL_endpoint

person	name	date
http://dbpedia.org/resource/Ambra_Angiolini	“Ambra Angiolini”@en	1977-04-22
http://dbpedia.org/resource/Gino_Paoli	“Gino Paoli”@en	1934-09-23
http://dbpedia.org/resource/Raoul_Bova	“Raoul Bova”@en	1971-08-14
http://dbpedia.org/resource/Toni_Servillo	“Toni Servillo”@en	1959-08-09
http://dbpedia.org/resource/Gabry_Ponte	“Gabriele Ponte”@en	1973-04-20
...	...	...

Table 2.2: Example of SPARQL result

at http://dbpedia.org/sparql/. Lately, OpenLink Software[11] made available the *OpenLink Software LOD Cache*[12] mirroring a number of LOD resources, e.g. YAGO, OpenCyc and WordNet (see Section 2.4).

2.3.4 Processing RDF data

There exists a number of tools developed to process the RDF data. Two of the most popular tools are the Java-based Apache Jena Framework[13] and Sesame Framework.[14]

Another set of tools is designed to store the datasets. In theory, data can be stored as RDF files and uploaded directly into the RAM to be processed. However, if the data are large-scale, it is more reasonable to store them in an index (like a *triplestore* defined in Section 2.1), that allows to store and quickly access the large-scale data. They may have their own storage mechanism implementation, e.g. Jena TDB,[15] Virtuoso,[16] AllegroGraph,[17] Sesame,[18] or use a third party storage implementation, e.g. a standard relational database management system.

2.4 The LOD cloud

Linked Data has grown rapidly in the last years. While in its initial history LOD has mainly been the domain of researchers, nowadays possible uses of this concept have become more widely recognized. Contributors to the W3C Linking Open community project[19] are constantly making

[11] http://www.openlinksw.com/
[12] http://lod.openlinksw.com/sparql
[13] http://jena.apache.org/
[14] http://www.openrdf.org/
[15] http://jena.apache.org/documentation/tdb/index.html
[16] http://virtuoso.openlinksw.com/
[17] http://www.franz.com/agraph/allegrograph/
[18] http://www.openrdf.org/
[19] http://linkeddata.org/

Domain	Datasets	Triples	(Out-)Links
Media	32	1 888 203 627	90 595 450
Geographic	41	6 249 316 381	36 823 880
Government	65	15 767 531 020	54 939 115
Publications	96	3 210 008 966	199 663 820
Cross-domain	42	3 062 558 698	76 697 070
Life sciences	42	1 478 739 496	13 133 147
User-generated content	23	1 672 891 245	18 335 104
...			
Total	322	33 286 864 530	473 917 366

Table 2.3: Linked Data resources by domain

their datasets available in RDF format and connecting them to other datasets in compliance with the LOD principles. Current state of LOD is visualized in the so-called Linking Open Data cloud diagram, created and maintained by Chris Bizer, Richard Cyganiak and Anja Jentzsch.[20] Additionally, the Open Knowledge Foundation,[21] a no-profit movement to open up knowledge, catalogs the datasets available under the LOD principles on a website called “the Data Hub”.[22] Using the API exposed by the website, we can group repositories by different domains, e.g. media, government, life sciences. Table 2.3 shows the state of the LOD by 12th September 2013.

In the following subsections we describe the most important vocabularies employed in LOD and the largest cross-domain datasets.

2.5 Resources

In this section, we describe the LOD resources we will use in our work. We also include Wikipedia, although it should not be considered as a Linked Open Data resource, since it does not follow the principles defined by Tim Berners-Lee. Nevertheless, most of the cross-domain resources, like DBpedia, YAGO and Freebase (see Sections 2.5.2 and 3.1) collects data from Wikipedia, therefore we include it in this Chapter.

Some other relevant LOD resources, not used in our research work, are described in Section 3.1.

[20] http://lod-cloud.net/state/#diagram
[21] http://okfn.org/
[22] http://datahub.io/

2.5.1 Wikipedia

Wikipedia is a multilingual collaborative encyclopedia, supported by the Wikimedia Foundation. It is the biggest encyclopedia ever built by human, it is available online and free to consult. As of 2013, it is the sixth most visited website in the world, with 60 million visitors per day.[23] Its crowdsourcing model and the very productive community around it assure that the stored information is very precise and updated.

As a result, beside becoming a reference knowledge source for people in their everyday life, in the last years Wikipedia has been more and more exploited for various Information Extraction tasks. Researchers started to use such resource to substitute or integrate conventional lexical semantic resources such as WordNet [Fel98] or linguistically annotated corpora in different NLP tasks (e.g. word sense disambiguation, semantic role labeling, information retrieval, text categorization, question answering).

Wikipedia is divided into articles, like a traditional encyclopedia, each of which corresponds to a particular topic and is identified by a unique title. In case of ambiguity, a characteristic of the page is added after the title enclosed in brackets, to assure the uniqueness of the page title. For example, the page titled "Titanic" refers to the ship, while "Titanic (1997 film)" is related to the famous movie. In addition, Wikipedia community provides a particular kind of pages, called *disambiguation pages*, to host the list of all pages related to a certain lemma. In our example, there exists the page "Titanic (disambiguation)" for this purpose.

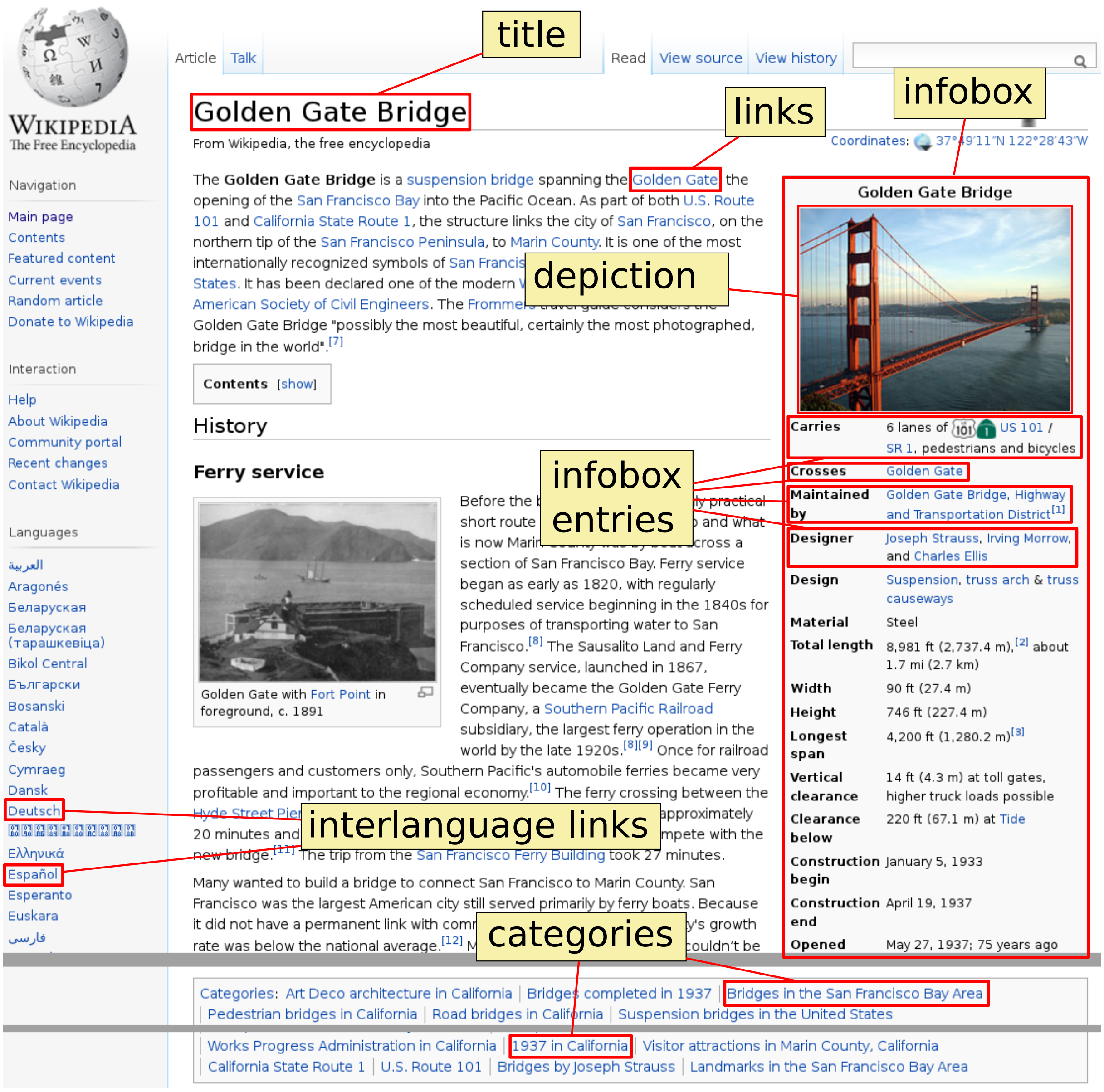
The drawing up of the pages is usually provided by community members, but visiting users also can add new pages or edit existing ones. Articles are written using *Wikitext*, a lightweight markup language, used as a simplified alternative to HTML. The conversion between Wikitext and HTML is achieved by MediaWiki, the web content management system used by Wikipedia. Recently, Wikimedia Foundation developers have implemented a Beta version of VisualEditor, a way to edit pages without needing to learn wiki markup, by using a WYSIWYG paradigm.

From an aesthetic point of view, a Wikipedia page is formed by some particular features useful for consultation: links (to other topics or to other languages), infoboxes, categories, and so on.

Infoboxes

Some years ago, a valuable new feature was added to Wikipedia, the *infobox*, i.e. a table on the top right-hand side of a page, summarizing the key concepts related to the subject of an article (see Figure 2.2). For example, biographies in Wikipedia have a specific *infobox* usually containing factual information about the person described in the article, e.g. date of birth, nationality, activity, and so on. Originally, it was designed for an aesthetic purpose (it fills the top right-hand

[23] `http://www.alexa.com/topsites`, visited the 8th of August, 2013.



Navigation

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Languages

العربية
Aragonés
Беларуская
Беларуская (тарашкевіца)
Bikol Central
Български
Bosanski
Català
Česky
Cymraeg
Dansk
Deutsch
Ελληνικά
Español
Esperanto
Euskara
فارسی

Golden Gate Bridge

From Wikipedia, the free encyclopedia

Coordinates: 37°49′11″N 122°28′43″W

The **Golden Gate Bridge** is a suspension bridge spanning the Golden Gate the opening of the San Francisco Bay into the Pacific Ocean. As part of both U.S. Route 101 and California State Route 1, the structure links the city of San Francisco, on the northern tip of the San Francisco Peninsula, to Marin County. It is one of the most internationally recognized symbols of San Francis States. It has been declared one of the modern American Society of Civil Engineers. The Fromme Golden Gate Bridge "possibly the most beautiful, certainly the most photographed, bridge in the world".[7]

Contents [show]

History

Ferry service

Golden Gate with Fort Point in foreground, c. 1891

Before the short route is now Marin section of San Francisco Bay. Ferry service began as early as 1820, with regularly scheduled service beginning in the 1840s for purposes of transporting water to San Francisco.[8] The Sausalito Land and Ferry Company service, launched in 1867, eventually became the Golden Gate Ferry Company, a Southern Pacific Railroad subsidiary, the largest ferry operation in the world by the late 1920s.[8][9] Once for railroad passengers and customers only, Southern Pacific's automobile ferries became very profitable and important to the regional economy.[10] The ferry crossing between the Hyde Street Pier approximately 20 minutes and mpete with the new bridge.[11] The trip from the San Francisco Ferry Building took 27 minutes.

Many wanted to build a bridge to connect San Francisco to Marin County. San Francisco was the largest American city still served primarily by ferry boats. Because it did not have a permanent link with comr y's growth rate was below the national average.[12] M couldn't be

Golden Gate Bridge	
Carries	6 lanes of 101 1 US 101 / SR 1, pedestrians and bicycles
Crosses	Golden Gate
Maintained by	Golden Gate Bridge, Highway and Transportation District[1]
Designer	Joseph Strauss, Irving Morrow, and Charles Ellis
Design	Suspension, truss arch & truss causeways
Material	Steel
Total length	8,981 ft (2,737.4 m),[2] about 1.7 mi (2.7 km)
Width	90 ft (27.4 m)
Height	746 ft (227.4 m)
Longest span	4,200 ft (1,280.2 m)[3]
Vertical clearance	14 ft (4.3 m) at toll gates, higher truck loads possible
Clearance below	220 ft (67.1 m) at Tide
Construction begin	January 5, 1933
Construction end	April 19, 1937
Opened	May 27, 1937; 75 years ago

Categories: Art Deco architecture in California | Bridges completed in 1937 | Bridges in the San Francisco Bay Area | Pedestrian bridges in California | Road bridges in California | Suspension bridges in the United States | Works Progress Administration in California | 1937 in California | Visitor attractions in Marin County, California | California State Route 1 | U.S. Route 101 | Bridges by Joseph Strauss | Landmarks in the San Francisco Bay Area

Figure 2.2: An example of Wikipedia page

corner box often present in Wikipedia pages) but then it became a very important source of structured information for research purposes. The data in the *infoboxes* can be easily extracted to build a knowledge base, and some research projects are born with this aim, such as DBpedia (see Section 2.5.2).

Categories

To organize Wikipedia for easy access to pages, contributors are given guidelines for categorizing articles and naming new categories. Categories allow articles to be placed in one or more groups, and allow those groups to be further categorized. For instance, the category *Bridges in the San Francisco Bay Area* is a subcategory of *Bridges in California*. Articles are tagged with a category and linked to the category page. For instance, the article "Golden Gate Bridge" is tagged with the categories *Bridges in the San Francisco Bay Area* and *1937 in California*. The thematic tagging of the articles according to their categories cannot be used as classification (the resulting graph, due to the crowdsourcing paradigm, leads to cycles), but can be exploited to set up the basis to build elementary taxonomies [NS08] or complex ontologies, like YAGO [SKW07] (see also Section 3).

Inter-language links

Inter-language links (or cross-language links) connect articles about the same subject in different languages.

Originally, inter-language links were added manually to each article of Wikipedia in each language. The problem with this approach was that each language had to maintain its own separate lists. So for example, if the name of a page in the English Wikipedia changed, then each language that linked to that page would have to separately notice this fact and then change their own links. Most of the inconsistencies were found using bots and solved manually by the Wikipedia community, resulting in a reduction of conflicts to less than 1% of pages [KBA+12].

Recently, the Wikidata project have centralized the cross-language links for all Wikipedias. The Wikidata entry for a page contains (among other things) a list of links for that entry in different languages (see Section 2.5.3).

These links are widely used for machine translation [DG07] and distribution estimations [SC12].

2.5.2 DBpedia

The DBpedia project [LIJ+14] is a database extracted from Wikipedia. Since Wikipedia pages contain a lot of data potentially useful for automatic programs, the Semantic Web community started an effort to extract data from Wikipedia and then to publish them in a structured format (following the open standards of the Semantic Web, see Section 2.2) to make them machine-readable. Taking benefit from Wikipedia wide corpus, DBpedia 3.9 release[24] describes:

> 4.0 million things, [. . .] including 832,000 persons, 639,000 places (including 427,000 populated places), 372,000 creative works (including 116,000 music albums, 78,000 films and 18,500 video games), 209,000 organizations (including 49,000 companies and 45,000 educational institutions), 226,000 species and 5,600 diseases.

At the early stages of the project, the construction of DBpedia was solely based on the English Wikipedia. Until 2011, the DBpedia dataset included data from non-English Wikipedia pages only if there existed an equivalent English page. However, since there are many pages in the non-English Wikipedia editions that do not have an equivalent English page, relying on English Wikipedia pages only had the negative effect that DBpedia did not contain data for these entities. More recently, other contributors around the world have joined the project to create localized and interconnected versions of the resource. The goal is to populate the same ontology used by the English project, taking articles from editions of Wikipedia in different languages. The DBpedia 3.7 release addressed this problem and provided 15 new localised editions of the dataset.

At the time of writing this thesis, there are 22 different mapping-based versions of DBpedia in its 3.8 release, containing around 20 million entities.

The most recent version (3.9), released on 17th September 2013, contains some improvements, such as:

- Enlarged ontology, now containing 529 classes and over 2,000 properties.
- 24 localized editions (included English), built from a total of 3,177 mappings.
- Extended type system using the algorithm proposed in [PB13].
- New RDF links to external resources.

Since our work has been done between August 2012 and August 2013, all the results in the subsequent Chapters refer to version 3.8 of DBpedia, the last available in that period of time.

The project involves researchers in various parts of the worlds and it is structured as follows.

Ontology

The DBpedia community releases and maintains a shallow, cross-domain ontology. Version 3.8 consists of 359 classes (e.g., `Person`, `City`, `Organization`) – organized in a subsumption

[24] http://blog.dbpedia.org/2013/09/17/dbpedia-39-released-including-wider-infobox-coverage-additional-type-statem

hierarchy – and 1,775 properties (e.g., `birthPlace`, `latitude`, `familyName`), and is populated using a semi-automatic rule-based approach that relies prominently on Wikipedia infoboxes (see Section 2.5.1).

The ontology is released in the Web Ontology Language (OWL) format[25], used for describing the data with maximum expressiveness. Inter alia, OWL provides inventory for describing classes and properties, defining new classes on top of existing ones, expressing relations of equivalence and non-equivalence.

Since DBpedia 3.5, the ontology is built using a public wiki, that allows external contributors to add classes and properties. A browseable and updated version is available on the mappings website.[26]

Since DBpedia 3.7, the ontology is a directed-acyclic graph, not a tree, that is classes may have multiple parents. A taxonomy (tree) can still be constructed by only considering the parent class that is specified first in the list, since it is considered the most important.

Mappings

As Wikipedia's infoboxes have evolved over time, different communities of Wikipedia editors use different templates to describe the same type of things. For example, the English Wikipedia uses *Persondata* template for biographies, while the Italian version of the encyclopedia uses *Bio*. The French chapter has no such a template. Similarly, different templates use different names for the same attribute (e.g. *birthplace* and *placeofbirth*). As many Wikipedia editors do not strictly follow the recommendations given on the page that describes a template, attribute values are expressed using a wide range of different formats and units of measurement.

In order to populate the ontology dealing with these issues, the DBpedia community uses a crowdsourcing paradigm to map infoboxes and infobox attributes to the classes and properties of the DBpedia ontology, respectively. For example, the *Bio* template in the Italian Wikipedia is mapped to the DBpedia class `Person`. Its properties are then mapped to the ontology properties of class `Person`. In this case, *Sesso* is mapped to `gender`, *LuogoNascita* to `birthPlace`, and so on. Finally, these mappings are collected and stored in an online repository.[27]

Since DBpedia 3.5 – similarly to the construction of the ontology – contributors around the world can use a public wiki to add new mappings.

As the number of mappings required to cover all the infoboxes is extremely large, the mapping process follows an approach based on the frequency of the infoboxes and infobox attributes; most frequent items are mapped first. This guarantees a good coverage, as infoboxes are distributed

[25] `http://www.w3.org/TR/owl-features/`
[26] `http://mappings.dbpedia.org/server/ontology/classes/`
[27] The repository is publicly available at `http://mappings.dbpedia.org`

according the Zipf's law [SHB+12]. Therefore, although the number of mappings may be small, a large number of articles can be added to the ontology.

DBpedia extraction framework

The DBpedia community develops and releases a flexible and extensible framework to extract different kinds of structured information from Wikipedia and convert them into RDF triples (see Section 2.3.1 for more information about this format). The software is written using Scala 2.8[28] and is available for download from the DBpedia Github repository.[29]

The extraction framework takes as input the Wikipedia dump, available for download through the WikiMedia downloads portal.[30] The tool then automatically downloads the last version of both ontology and mappings and uses them to create the resource: every page in Wikipedia that contains an infobox mapped to a specific class is automatically added to such class. For example, `Barack Obama` article in the Italian Wikipedia includes in its wiki source code the *Bio* infobox. Since *Bio* maps to the `Person` class, `Barack Obama` is automatically classified as `Person`, and the corresponding RDF triple is generated.

Consuming DBpedia

The datasets extracted from Wikipedia are available for download[31] in two different formats (n-triples and n-quads), and licensed under the terms of the Creative Commons Attribution-ShareAlike License and the GNU Free Documentation License.

In addition, the community maintains different SPARQL endpoints (almost one for each language[32]) that can be queried remotely. See Section 2.3.3 for more information about SPARQL.

2.5.3 Wikidata

Wikidata [Vra12] is collaborative knowledge bases manually compiled by their community members, and operated by the Wikimedia Foundation.

The project started at the end of 2012, with the aim of providing a common centralized source for certain information about pages. The development of Wikidata has been split into three phases:

[28]http://www.scala-lang.org/
[29]https://github.com/dbpedia/
[30]http://dumps.wikimedia.org/
[31]http://wiki.dbpedia.org/Downloads
[32]The SPARQL endpoint for the English DBpedia is available at http://dbpedia.org/sparql.

- Enhancing links to articles on the same topic in different languages (inter-language links, see Section 2.5.1).
- Adding consistency between different chapters of Wikipedia by getting infobox information directly from Wikidata. In practice, structured information about entities (such as birth dates for biographies, geographical coordinates for places, and so on) are stored in a unique database and local editors can use it to populate infoboxes.
- Allowing automatic list creation based on data in Wikidata, instead of creating and maintaining them by hand.

Currently, phase 1 is already completed and phase 2 has started.

Objects in Wikidata are characterized by a machine-readable unique identifier with URI `http://www.wikidata.org/entity/Q{id}`, where {`id`} is an incremental integer referring to a certain entity. For example, the URI `http://www.wikidata.org/entity/Q42` refers to the writer Douglas Adams.

At this stage of the development, the LOD representation of Wikidata is still under discussion,[33] but the dump is already available for download in XML/JSON format.[34]

[33] http://meta.wikimedia.org/wiki/Wikidata/Development/RDF
[34] http://dumps.wikimedia.org/wikidatawiki/

3 Related work

This chapter presents a selection of previous work, relevant with respect to the topics reported in this thesis, in particular Linked Open Data resources, schema matching, entity classification, Distant Supervision, and Question Answering.

3.1 LOD Resources

Besides DBpedia and Wikidata, already described in Chapter 2, there are hundreds of Linked Open Data resources. In the following sections, we describe YAGO and Freebase, two of the most significant ones in the context of this thesis.

3.1.1 YAGO

YAGO [SKW07], a project similar to DBpedia started in 2007, aims at extracting and mapping entities from Wikipedia using categories (for fine-grained classes) and WordNet [Fel98] (for upper-level classes).

First, each Wikipedia page is a candidate to become an individual in YAGO. In Wikipedia, page titles are always unique, therefore titles can be easily used as IDs for concepts. For example, the page titled `Steve Jobs` is a candidate to become the individual `SteveJobs` in YAGO. Then, Wikipedia categories are divided into different types: *conceptual categories*, *administrative categories*, and *relational categories*. Conceptual categories identify a class for the entity. For example, `Steve Jobs` is in the category *American industrial designers*. A shallow linguistic parsing of the category name breaks the category name into different parts and, heuristically, when a plural word is present (*designers*) the category is most likely a conceptual category. Wikipedia categories are organized in a sort of graph, but it is not acyclic and it is built using crowdsourcing, therefore it is not suitable for use as an ontology. YAGO uses WordNet to establish the hierarchy of classes. Leaf conceptual categories in Wikipedia are added as subclasses to the WordNet ones, by means of a heuristic algorithm. In addition, Wikipedia redirects are used to obtain alternative names of pages (for example, *CIA* for *Central Intelligence Agency*), and WordNet synsets are exploited to reveal the meaning of words (for example, “individual” and “mortal” both belong to the synset “person”). Finally, YAGO exploits some particular categories to map relations, by applying patterns to the category names. For example, the page `Steve Jobs` is in the category *1955 births*: this means that the individual Steve Jobs was born in 1955, resulting in the corresponding fact ⟨Steve Jobs, `bornIn`, 1955⟩. Compared to DBpedia, this method can cover only a small number of properties (52 in YAGO 2).

More recently [HSBW12], YAGO has been enriched with spatial information, imported from GeoNames.[1]

The accuracy of YAGO has been calculated by presenting randomly selected facts of the ontology to human judges and asking them to assess whether the facts were correct, resulting in a global accuracy of 95%. Currently, YAGO contains knowledge about 9.8 million entities and 447 million facts [HSBW12].

Although YAGO is an automatically built resource, its rules are mainly selected manually. The coverage over Wikipedia is higher than DBpedia, but this rule-based approach makes the resource language-dependent. In addition, its ontology contains thousands of hundreds of classes, resulting in difficulties on training a suitable classifier.

3.1.2 Freebase

Freebase [BEP+08] is a large collaborative knowledge base originally developed by Metaweb and subsequently acquired by Google in 2010. It contains knowledge automatically harvested from many sources (Wikipedia, MusicBrainz, NNDB, FDA, and others), and manually added by individual ’wiki’ contributions.

Freebase can be considered as a huge graph. Its subjects are called *topics*, and they have a similar meaning to “things” in DBpedia. Each topic is classified using *types*. Depending on its type, a topic can be connected with different additional data (including properties and facts). Each aspect of the Freebase ontology (types and topics) is user-editable, but some structural edits need the approval of a Metaweb/Google employee.

Currently, Freebase describes more than 23 million topics, to each of which a global unique identifier is assigned, using the pattern `/type/object`. For example, Steve Jobs identifier is `/m/06y3r`.

[1] http://www.geonames.org/

Although Freebase is patented and owned by a for-profit company (Google), the resource is freely downloadable and queryable under an open license.

3.2 Entity classification

There are various projects aiming to extract Wikipedia entity types boostrapping information contained in the categories. For example, [NCM08] uses extracted datatypes to train a named entity recognizer, while [NS08] examines Wikipedia categories and automatically cleans them.

The tool presented in [GNP+12], *Tipalo*, identifies the most appropriate class of a Wikipedia article by parsing its page abstract using natural language processing tools and resources. In this context, only English Wikipedia is considered, therefore this classifier cannot be easily adapted to other languages.

Similarly, [Poh12] considers only the English DBpedia and therefore does not take advantages from inter-language links. In addition, there is some manual effort to classify biographies (using tokens from categories[2]), that leads to very good results, but is not automatically portable to other languages; again linguistic tools are used to extract the definition from the first sentence.

The approach presented in [Giu09] classifies people over an excerpt of the WordNet ontology, using kernel functions that implicitly map entities, represented by aggregating all contexts in which they occur, into a latent semantic space derived from Wikipedia. This approach queries online the name of the entity to collect contextual information.

[GG07] proposes an unsupervised approach based on lexical entailment, consisting in assigning an entity to the category whose lexicalization can be replaced with its occurrences in a corpus preserving the meaning.

[Gan13] provides a deep analysis of several tools, either conceived specifically for knowledge extraction on the Semantic Web, or adaptable to it, or even acting as aggregators of extracted data from other tools.

3.3 Schema matching

Schema matching is a problem of data integration and consists in finding semantic relationships between objects defining different database schemas. In most implementations, schema matching is typically performed manually. Lately, research papers have proposed many techniques to achieve a partial automation of the match operation for specific application domains.

[2]For example, articles that belong to `Living people` category or categories ending with *births* or *deaths* were classified as `Person`

A general survey on the topic is presented by Rahm and Bernstein [RB01]. Their work compares and describes different techniques, establishing also a taxonomy that is used to classify schema matching approaches. Similarly, Shvaiko and Euzenat [SE05] present a new classification of schema-based matching techniques. It also overviews some of the recent schema/ontology matching systems, focusing on which part of the solution space they cover.

Bouma et al. [BDI09] propose a method for automatically completing Wikipedia templates. Cross-language links are used to add and complete templates and infoboxes in Dutch with information derived from the English Wikipedia. First, the authors show that the alignment between English and Dutch Wikipedia is accurate, and that the result can be used to expand the number of template attribute-value pairs in Dutch Wikipedia by 50%. Second, they show that matching template tuples can be generated automatically, and that an accurate set of matching template/attribute pairs can be derived using intersective bidirectional alignment. In addition, the alignment provides valuable information for normalization of template and attribute names and can be used to detect potential mistakes. The method extends the number of tuples by 50% (27% for existing Dutch pages).

Adar et al. [ASW09] present Ziggurat, an automatic system for aligning Wikipedia infoboxes, creating new infoboxes as necessary, filling in missing information, and detecting inconsistencies between parallel articles. Ziggurat uses self-supervised learning to allow the content in one language to benefit from parallel content in others. Experiments demonstrate the method's feasibility, even in the absence of dictionaries.

Nguyen et al. [NMN$^+$11] propose WikiMatch, an approach for the infobox alignment task that uses different sources of similarity. The evaluation is provided on a subset of Wikipedia infoboxes in English, Portuguese and Vietnamese.

More recently, Rinser et al. [RLN13] propose a three-stage general approach to infobox alignment between different versions of Wikipedia in different languages. First, it aligns entities using inter-language links; then, it uses an instance-based approach to match infoboxes in different languages; finally, it aligns infobox attributes, again using an instance-based approach.

3.4 Distant supervision

A Distant Supervision learning algorithm is a semi-supervised learning algorithm that uses a weakly labeled training set (typically relying on a knowledge base). The idea [MBSJ09], presented in 2009, has been widely used for relation extraction purposes [NM11].

The *Intelligence in Wikipedia* project [WWA$^+$08] first explored this approach basing on Wikipedia and DBpedia, in order to improve infoboxes coverage for cases in which the required information can be found in the article text. In [HZW10] the same idea is used to mine the web and automatically collect unknown relations starting from unlabeled structured data, such as tables and lists extracted from the web.

In the SW community, the University of Leipzig developed BOA [GN11], a system that uses information from linked data knowledge bases to extract patterns expressing such relations. The extracted patterns are then applied to web articles to extract new instances of the relation described by the patterns. This knowledge is finally put back into the knowledge base, closing the loop. In practice, BOA extends the idea underlying *Intelligence in Wikipedia* [WWA+08] to the web-scale. The distant supervision paradigm has also been applied for different purposes, e.g. temporally anchored relation extraction [GPnCR12], slot filling tasks [SMT+10] and sentiment classification [GBH09].

In [RYM10] a new approach to distant supervision is proposed, dealing with the presence of noisy examples in the training, a consequence of the assumption that each sentence which mentions the entities involved in the relation is an expression of the relation itself. A survey on noise reduction methods for distant supervision is discussed in [RBWK13].

Recently, the distant supervision paradigm has been tested on the web, to obtain rule sets large enough to cover the actual range of linguistic variation, thus tackling the long-tail problem of real-world applications [KLUX12], for sentiment analysis in social networks [GBH09], and fact checking [LGMN12].

Finally, there were some preliminary works on applying distant supervision on Wikipedia and DBpedia in Portuguese [BFS+13] and Polish [ZP13]. In particular, the latter uses part-of-speech tagging and SVM machine learning algorithm for classification.

3.5 Question answering

Question Answering (QA) discipline consists in automatically answer to questions expressed in natural language.

A survey on the QA research field is provided in [LUSM11], with a focus on ontology-based QA. Moreover, they examine the potential of the open user friendly interfaces for the SW to support end users in reusing and querying the SW content. State of the art QA systems over Linked Data generally address the issue of question interpretation mapping a natural language question to a triple-based representation. For instance, Freya [DAC12] is an interactive Natural Language Interface for querying ontologies. It uses syntactic parsing in combination with the ontology-based lookup for question interpretation, partly relying on the user's help in selecting the entity that is most appropriate as match for some natural language expression. One of the problem of that approach is that often end-users can be unable to help, in case they are not informed about the modeling and vocabulary of the data. PowerAqua [LUSM09] accepts user queries expressed in Natural Language and retrieves answers from multiple semantic sources on the SW. It follows a pipeline architecture, according to which the question is *i)* transformed by the linguistic component into a triple based intermediate format, *ii)* passed to a set of components to identify potentially suitable semantic entities in various ontologies, and then *iii)* the various interpretations produced in different ontologies are merged and ranked for answer retrieval. The major shortcoming of PowerAqua is its limited linguistic coverage, that we address in our system exploiting the variability of relational patterns extracted from Wikipedia.

Pythia [UC11] relies on a deep linguistic analysis to compositionally construct meaning representations using a vocabulary aligned to the vocabulary of a given ontology. While it can handle linguistically complex questions, Pythia's major drawback is that it requires a lexicon, which up to this moment has to be created manually. It therefore fails to scale to very large datasets. More recently, Unger and colleagues [UBL+12] present an approach more similar to the one we adopt in QAKiS (see Chapter 10). Their system (based on Pythia [UC11]) relies on a linguistic parse of the question to produce a SPARQL template that directly mirrors the internal structure of the question (i.e. a SPARQL template with slots that need to be filled with URIs). This template is then instantiated using statistical entity identification and predicate detection (e.g. applying string similarity as well as natural language patterns extracted from structured data and text documents). However, differently from the other two approaches mentioned before, the last one has not yet been evaluated on the standard data sets of the Question Answering over Linked Data challenge.

While most of the approaches focus on the problem of deriving a structured query for a given natural language question, [EVS12] addresses the problem from the opposite perspective. More specifically, they propose a preliminary approach to query verbalization, i.e. the translation of a structured query into natural language expressions that are readable and understandable by the human day-to-day user.

Beside question answering, keyword-based approaches have been developed over the past years to query the SW, leading to semantic search engines. Among the others, Swoogle [DFJ+04], Sigma [TCC+10], Watson [dMS+08], and Sindice [TDO07], aim at index RDF across the Web and make it available for entity search.

4
Pre-processing data

Experiments described in this thesis have been performed using resources (such as Wikipedia and DBpedia) freely available on the web. In particular, Wikipedia dumps are released in XML format,[1] while DBpedia datasets are provided[2] in RDF format (see Section 2.3.1). We then develop a set of scripts, written in Java, to access this data and use them in our algorithms. In particular, XML and Wiki parsers have been used to extract templates, categories, sections and other semi-structured information from Wikipedia. An RDF parser has been developed to manage DBpedia datasets.

In addition, some pre-processing steps need some heuristics. In this Chapter, we describe such pre-processing steps, used in the algorithms described in Chapters 5 to 8.

4.1 Filtering Wikipedia templates

A template is a special Wikipedia page created to be included in other pages. Templates usually contain patterns that might need to show up on any number of articles. They are commonly used for boilerplate messages, standard warnings or notices, infoboxes, navigational boxes and similar purposes. We can divide them into two broad categories:

Infoboxes are fixed-format tables designed to be added to the top right-hand corner of articles to consistently present a summary of some unifying aspects that the articles share (see Section 2.5.1). The DBpedia project uses this category of templates: most of them are manually linked to a particular class in the DBpedia ontology.

[1]http://dumps.wikimedia.org/
[2]http://wiki.dbpedia.org/Downloads

Macro templates are used to give graphic coherence to the same element in different pages. For instance, templates of this class are used for country flags, dates, portals, and so on. This class is not useful for our purpose, therefore we will ignore them.

In our research we are particularly interested in infoboxes, but unfortunately Wikipedia does not provide a simple way to understand whether a particular template is then rendered as an infobox. Some Wikipedia editions use the *Infobox* prefix to identify them, but this is not a standard (for example, Italian chapter of Wikipedia does not use it [RLN13]). Thus we implement a simple rule-based hand-crafted classifier based on the following heuristics:

(i) If a template is an infobox, it is included only once in the page, so for each template we count the total number of occurrences and the total number of pages in which it appears. The ratio between these two values must be 1. There are some rare cases (like the `Bio` template in Italian) in which an infobox can be included twice, so we relaxed this constraint and considered 1.5 as a good ratio.

(ii) Templates can be represented using their parameters, that can be a single value or `key/value` pairs. Infoboxes use only the latter format, so we removed the others.

(iii) Finally, infoboxes are usually written one key/value pair per line, for readability purpose; we only keep templates in which the difference between the number of lines and the number of pairs is greater than or equal to zero.

In this way, we remove more than 90% of templates, obtaining few infoboxes for each page (on average 1.27 and 1.18 templates per page in English and Italian, respectively). Statistics about the extraction are shown in Table 4.1.

To assess the correctness of our approach, we compared the list of extracted infoboxes and the list of the mapped templates in DBpedia: the first set strictly contains the second one, therefore we can estimate that our heuristic rules are a good approximation of the required set. A similar approach for template filtering has been also used in [AS12].

4.2 Wikipedia and DBpedia entities representation

As already underlined in Section 2.5.2, the English and Italian Wikipedia have an infobox for biographies (`PersonData` and `Bio`, respectively), while Spanish and French do not. DBpedia stores the cross-language information, but it is not used to map the infoboxes. For example, Clint Eastwood is classified as `Actor` in the French DBpedia and as `Person` in the Italian one. We deal with this problem, trying to classify pages in all languages to the most specific class.

	EN	DE	IT	PT	FR	ES
Wikipedia templates	391,780	49,744	104,044	43,385	148,200	24,946
Wikipedia infoboxes	21,250	1,525	2,238	2,469	2,587	552
Wikipedia article pages	3,932,148	1,325,792	924,544	740,585	1,251,585	953,848
DBpedia mapped pages	1,716,555	205,903	607,842	226,497	15,463	15,987
DBpedia mapp. pag. after CL	1,902,585	482,747	652,395	430,603	518,874	419,168

Table 4.1: Statistics taken from different chapters of Wikipedia and DBpedia: number of templates in Wikipedia, number of filtered templates in Wikipedia, number of articles in Wikipedia, in DBpedia, and in DBpedia after using Wikidata cross-language information. Dump versions are listed in Table 7.2.1

en	de	it	...	DBpedia class
Xolile Yawa	Xolile Yawa	*null*	...	`Athlete`
The Locket	*null*	Il segreto del medaglione	...	`Film`
Barack Obama	Barack Obama	Barack Obama	...	`Politician`
null	*null*	Giorgio Dendi	...	`Person`
Secoya People	*null*	Secoya	...	`EthnicGroup`
...	...	...	...	...

Figure 4.1: A portion of the entity matrix

First, we build an entity matrix, using the cross-language information stored in the Wikidata dataset. Then, we supplement this matrix by assigning to each entity the most suitable DBpedia class.

4.2.1 Building the entity matrix

Let $\mathcal{L}$ be the set of languages available in Wikipedia, we first build a matrix E where the i-th row represents an entity e_i and j-th column refers to the corresponding language $l_j \in \mathcal{L}$. Cross-lingual information (see Section 2.5.1), extracted from Wikipedia (later Wikidata), is used to automatically align on the same row all articles that describe the same entity. The element $E_{i,j}$ of this matrix is *null* if a Wikipedia article describing the entity e_i does not exist in l_j. An entity in our system is therefore represented as a row of the matrix, where each j-th element is a Wikipedia article in language l_j. Figure 4.1 shows a portion of the entity matrix.

4.2.2 Assigning DBpedia class to entities

First, we need to assign a single class in the DBpedia ontology to each entity of the matrix. As said, we will use the classes already annotated by the DBpedia community. Using the DBpedia annotation tool, annotators can assign a unique class to each infobox. However, this is not necessarily true if we consider more than one language. For example, the British Library is classified as a `Library` (subset of `Building`) in the English DBpedia, and as an `EducationalInstitution` (subset of `Organisation`) in the German DBpedia. We deal with such cases filtering the classes from different DBpedias as follows.

- If the entity belongs to more than one ontology class and such classes have one or more ancestor class in common, then the most specific common class is used. For example, *Barack Obama* is `OfficeHolder` in the French DBpedia and `President` in the Spanish one. These two classes are both subclass of `Person`, so we only consider this class.
- If the more-than-one-class problem involves classes connected in a chain of subclass of relations, we consider the most specific class. For instance, the famous singer *Michael Jackson* is classified as a `Person` in the Italian and German DBpedia, an `Artist` in the English DBpedia and a `MusicalArtist` in the Spanish DBpedia. The most specific class is the last one, so the entity *Michael Jackson* is considered as a `MusicalArtist`.
- Finally, when an entity is classified using two classes not having a common ancestor, that entity is left as `Unknown`. Articles about battle tanks are examples of this kind, as different DBpedias classify them both as `Weapon` and `MeanOfTransportation`.

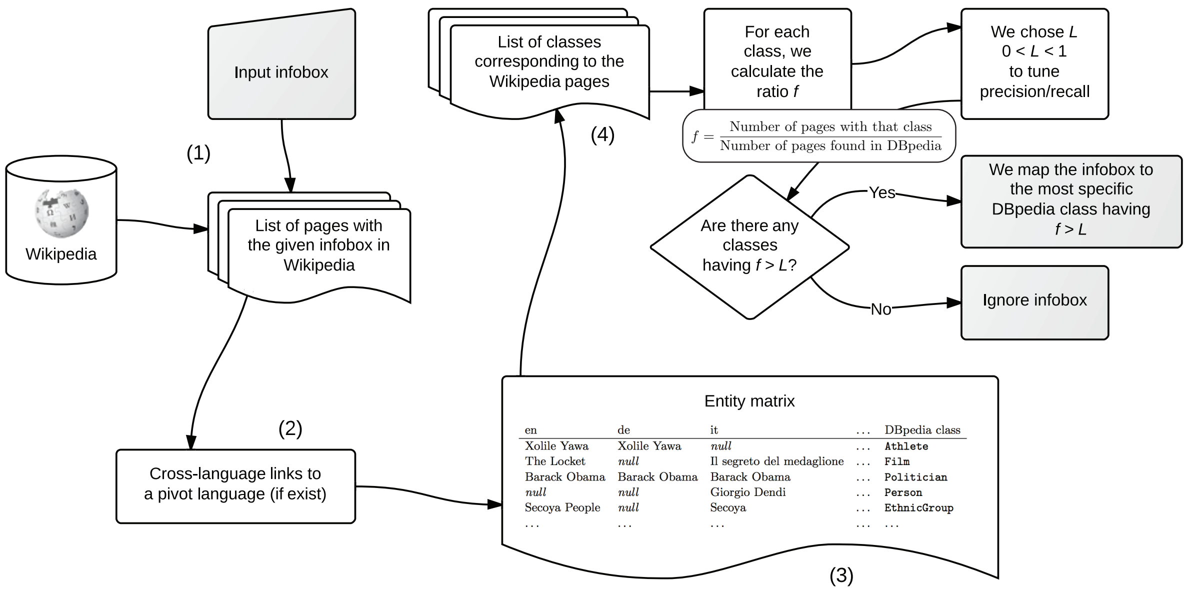
Table 4.1 shows statistics for each language, about this enriched DBpedia.

5 Automatic mapping generation for classes

As described in Chapter 2, DBpedia is a LOD resource built extracting structured information from Wikipedia. The main effort for its construction is the mapping task, that searches for a match between the Wikipedia infoboxes (see Section 2.5.1) and DBpedia classes. For example, the *Taxobox* infobox sets out the Biological classification (taxonomy) for a group of living things; therefore, pages containing this infobox should be placed under the `Species` class in DBpedia. This connection is now manually mapped by the DBpedia contributors. In addition, when the DBpedia community starts a new chapter of the resource (in a new language) the effort has to be done from scratch using infoboxes in Wikipedia in such language, that usually differ from the ones used in the English edition of the encyclopedia.

In this chapter, we present a method that automatically maps Wikipedia infoboxes to the corresponding classes in the DBpedia ontology. These mappings can be used for the deployment of new chapters of DBpedia (see Section 2.5.2). To achieve this goal, we devised an approach that exploits Wikipedia cross-language links in six pivot languages (English, Italian, German, Portuguese, Spanish, French) and uses the existing DBpedia mappings for these languages. The method uses a rule-based approach to map Wikipedia infoboxes, taken from versions of Wikipedia in different languages, to the most probable class in the DBpedia ontology (Section 5.1).

Evaluation has been performed on five languages (Bulgarian, Czech, Indonesian, Dutch and Catalan), already available in the DBpedia project: manually annotated Wikipedia infoboxes are used as test data for evaluation (Section 5.2). We show that our approach increases the number of mappings with high accuracy and can be tuned to vary the tradeoff between precision and recall. Despite the algorithm used is simple to understand and implement, it has never been used in real applications; the results show that the approach is reliable and can save time in the mapping task.


Input infobox
(1)
Wikipedia
List of pages with the given infobox in Wikipedia
(2)
Cross-language links to a pivot language (if exist)
List of classes corresponding to the Wikipedia pages
(4)
For each class, we calculate the ratio *f*
We chose *L* $0 < L < 1$ to tune precision/recall
$f = \frac{\text{Number of pages with that class}}{\text{Number of pages found in DBpedia}}$
Are there any classes having $f > L$?
Yes
No
We map the infobox to the most specific DBpedia class having $f > L$
Ignore infobox
Entity matrix
(3)

en	de	it	...	DBpedia class
Xolile Yawa	Xolile Yawa	*null*	...	`Athlete`
The Locket	*null*	Il segreto del medaglione	...	`Film`
Barack Obama	Barack Obama	Barack Obama	...	`Politician`
null	*null*	Giorgio Dendi	...	`Person`
Secoya People	*null*	Secoya	...	`EthnicGroup`
...	...	...	...	...

Figure 5.1: Workflow of the system

5.1 Infobox mapping

Given a Wikipedia template classified as infobox using the algorithm described in Section 4.1, our goal is to map it, when possible, to a DBpedia ontology class. To this aim, we use the matrix built in Section 4.2 as information source to find the mapping. The approach is instance based: we exploit the Wikipedia pages (instances) already mapped to a DBpedia class and their cross language links to the pages that contain the template to be mapped. A simple method based on the frequencies of the resulting classes allows us to tune the tradeoff between precision and recall. The mapping algorithm (Figure 5.1) is implemented as follows.

(i) Given as input an infobox t taken from a version of Wikipedia in a specific language l (where l is not a pivot language), we collect all the pages P_l that include t. See Figure 5.2, parts (A) and (B).

(ii) We use the cross-language links contained in P_l to retrieve the pages P_E in the matrix E, for which we know the DBpedia classes (Section 4.2). See Figure 5.2, part (C). Let d be this value.

(iii) From the matrix E, we collect the DBpedia classes C of the pages P_E and count the number of their occurrences n_c ($c \in C$). See Figure 5.2, part (D). We also collect and count the occurrences of the parent classes in C. For example, if the classes `Person` and `Organisation`

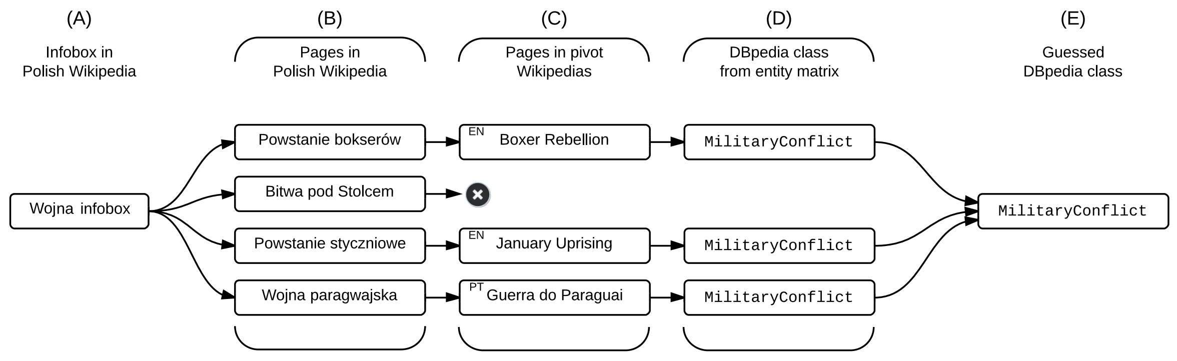


Figure 5.2: The automatic mapping algorithm applied to `Wojna infobox` in Polish

occur 5 and 3 times, respectively, we also consider the class `Agent` with frequency 8, as the latter class is the common ancestor of the first two.

(iv) For each found class, we compute the frequency of this class with respect to the number of pages included in the DBpedia dataset in one of the pivot languages. Let $f_c = n_c/d$ be this value.

(v) t is then mapped to most specific and most frequent class (max f_c). See Figure 5.2, part (E). A parameter L ($0 \leqslant L \leqslant 1$) is used to filter c such that $f_c \leqslant L$.

If $f_c > L$, then we map the infobox t to c, otherwise we climb one level up and recalculate f_c until it is bigger than L. If we reach the root of the ontology taxonomy without any class percentage exceeding L, then the system abstains, the infobox is discarded. The parameter L can be used to tune the tradeoff between precision and recall: the higher L, the higher precision and the lower recall; the lower L, the higher recall and the lower precision. See Section 5.2 for further details.

Consider, for example, the Polish template `Wojna_infobox` ("War" in English). It is included in 3,770 pages in the Polish Wikipedia. By using cross-language links, we found that 2,842 of these articles have a corresponding page in one of the six pivot languages, and 2,716 are classified in one of the corresponding DBpedia.

Table 5.1(a) shows the classes mapped from the pages in this set of entities. We can see that 2,697 pages are classified as `MilitaryConflict`. Since 2,697 out of a total of 2,716 classified pages corresponds to 99%, we can assume that this is the class that best approximates the possible mapping of the `Wojna_infobox` template. In particular, in this case the Polish word "Wojna" means "War", clearly a synonym of `MilitaryConflict`.

Wojna_infobox	
Pages in Polish Wikipedia	3,770
Pages found using CLL	2,842
Pages classified in DBpedia	2,716
MilitaryConflict (2)	2,697
Event (1)	2,697
Person (2)	9
Agent (1)	10
Place (1)	8
PopulatedPlace (2)	4
...	...

(a)

Park_infobox	
Pages in Polish Wikipedia	321
Pages found using CLL	83
Pages classified in DBpedia	52
Park (3)	24
ArchitecturalStructure (2)	24
Place (1)	52
PopulatedPlace (2)	2
ProtectedArea (2)	2
NaturalPlace (2)	1
...	...

(b)

Table 5.1: The distribution of the pages having Wojna_infobox (a) and Park_infobox (b) on the six pivot languages (in brackets, the depth of the class in DBpedia ontology)

Let us consider another example, involving a more ambiguous template, Park_infobox ("Park" in English). Although its translation does not give rise to ambiguity, the cross-language links bring to the situation shown in Table 5.1(b). In this case, the Park class surely has the majority, but its percentage is low (46%), therefore using a parameter $L = 0.5$ this solution is discarded and Place is given instead.

5.2 Experiments and evaluation

Experiments have been carried out on 5 languages (Bulgarian, Czech, Indonesian, Dutch, and Catalan) for which manually mapped infoboxes can be downloaded from the DBpedia official mapping website.[1] Specifically, we used the version made available on 5th April 2013.

Precision and recall values are calculated using these sets of mappings as gold standard.

Figure 5.4 shows the precision/recall curves, the grey dashed lines join points with the same F_1, showing that F_1 values range from 0.8 and 0.9. The different precision/recall points are obtained by varying the parameter L.

These curves confirm the differences between the various versions of Wikipedia: in some cases the precision is high (in Catalan we reach 100%), while in other languages it does not exceed 95%. Also, the precision/recall curves differ in shape and direction when our algorithm is applied to different languages. These differences reflect the different structure of infoboxes in the Wikipedia editions, as the policies on infoboxes change from language to language. [NMN+11]

[1] http://mappings.dbpedia.org/

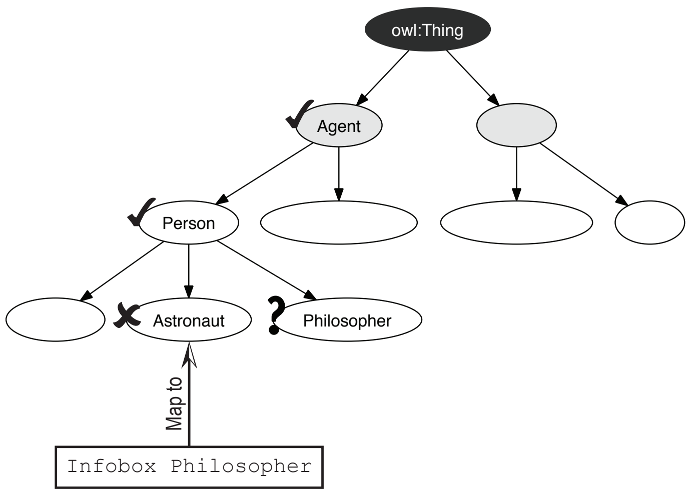


Figure 5.3: Description of the evaluation.

The evaluation is performed as proposed by [MR00] for a similar hierarchical categorisation task. Figure 5.3 shows an example of the evaluation. The system tries to classify the infobox `Philosopher` and map it to the ontology class `Astronaut`, while the correct classification is `Philosopher`. The missing class (question mark) counts as a false negative, the wrong class (cross) counts as a false positive, and the correct classes (ticks) count as true positives.

Results shows the reliability of our system. Once generated, the mappings can be checked by a human and can be used to start a new chapter of DBpedia or to extend an existing one, saving time in the mapping task.

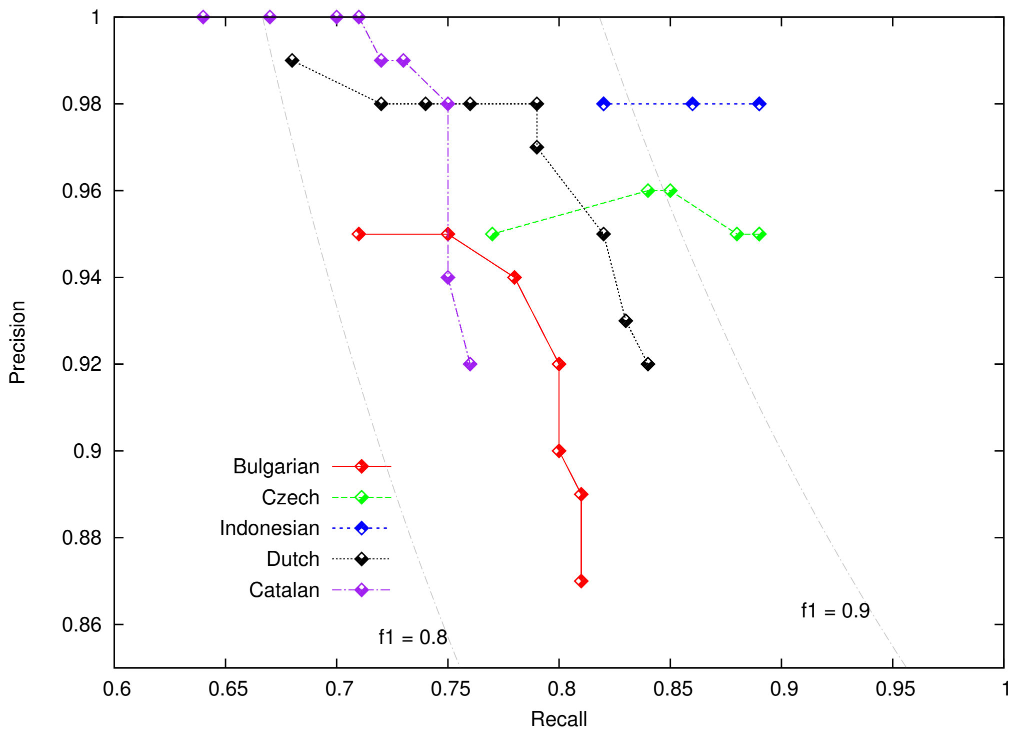


Figure 5.4: Evaluation of the infobox mapping system

6 Automatic mapping generation for properties

As discussed in Chapter 2, DBpedia is built by assigning a property of a Wikipedia infobox (ideally, an attribute of the entity described by the page, see Section 2.5.1 to a property in the DBpedia ontology. For example, in the *Television episode* infobox there are attributes such as *Title*, *Airdate*, and so on. The corresponding class in DBpedia is `TelevisionEpisode`, and the corresponding properties are `name`, `releaseDate`, and so on. This task is currently performed manually using crowdsourcing and there are no published attempts to perform it automatically.

In this chapter, we focus on the problem of automatically mapping infobox attributes to properties into the DBpedia ontology for extending the coverage of the existing localized versions (e.g., Italian, Spanish) or building from scratch versions for languages not yet covered (e.g., Swedish, Norwegian, Ukranian). The above problem can be classified as schema matching, limited to alignment as we do not perform any successive merging or trasforming.

We propose an instance-based approach, that exploits the redundancy of Wikipedia in different editions (languages), assuming that attributes and properties are equivalent if their values are similar. Specifically, the mapping is cast as a binary classification task in which instances are infobox attribute/ontology property pairs extracted from versions of Wikipedia and DBpedia in different languages and cross-language links are used to represent the instances in a unified space. This allows us to learn the mapping function, for example, from existing mappings in English and German and predict Swedish instances. Attributes and properties are compared using their values taking into account their types (i.e., date, integer, object, etc.). For attributes, the type is inferred from the value; for properties, the type is given by the ontology. We show that this approach is robust with respect to changes in Wikipedia, differently from approaches that first map infoboxes among Wikipedia in different languages. The evaluation has been performed on the Italian mappings. We compared our results with the current mappings on a random sample re-annotated by three different annotators. We report results comparable to the ones obtained by a human annotator in term of precision (around 87%), but our approach leads to a significant improvement in recall (around 80%) and speed.

6.1 Problem Formalization

We consider the problem of automatically mapping attributes of Wikipedia infoboxes into properties of the DBpedia ontology. The problem can be classified as schema/ontology matching in which we are interested in equivalence relations between attributes and properties.

An infobox is a set of *attribute/value* pairs that represent a summary of the most salient characteristics Wikipedia articles have in common. For example, the infobox `Officeholder` in the English Wikipedia contains generic attributes, such as `name`, `birth_date`, and `birth_place`, and specific ones, such as `term_start`, `party`, and `office`. Notice that each Wikipedia edition is maintained by different communities and has different guidelines that can have a strong impact on the mapping results. For example, in the Italian edition, `Carica_pubblica` (`Officeholder`) does not contain generic attributes that are usually contained in the infobox `Bio`. In addition, there are no constraints on types, therefore in some editions of Wikipedia there can be a single attribute `born` containing both place and date of birth, while other languages decide to split this information into different attributes.

A DBpedia property is a relation that describes a particular characteristic of an object. It has a *domain* and a *range*. The domain is the set of objects where such property can be applied. For instance, `birthDate` is a property of `Person`, therefore `Person` is its domain. Around 20% of the DBpedia properties use the class `owl:Thing` as domain. The range is the set of possible values of the property. It can be a scalar (date, integer, etc.) or an object (`Person`, `Place`, etc.). For example, the range of `birthDate` is date and the range of `spouse` is `Person`.

Manual mappings are performed as follows. First, human annotators assign an infobox to a class in the DBpedia ontology. Then, they map the attributes of the infobox to the properties of the ontology class (or of its ancestors). An example of mapping is shown in Figure 6.1.

The rest of the section is devoted to analyze the difficulties to adapt existing systems that perform infobox matching and completion (e.g., [RLN13, BDI09, ASW09]) to solve this task. We could use existing approaches to map infoboxes between different Wikipedia editions and, then, use the existing DBpedia mappings to extend the mappings to languages not yet covered. An example is shown in Figure 6.2, where the template `Persondata` in English has been mapped to `Bio` in Italian, and similarly `Officeholder` to `Carica_pubblica`. Suppose that Italian mappings do not exist yet, they can be derived using the existing English DBpedia mappings. However, approaching the problem in this manner leads to a series of problems.

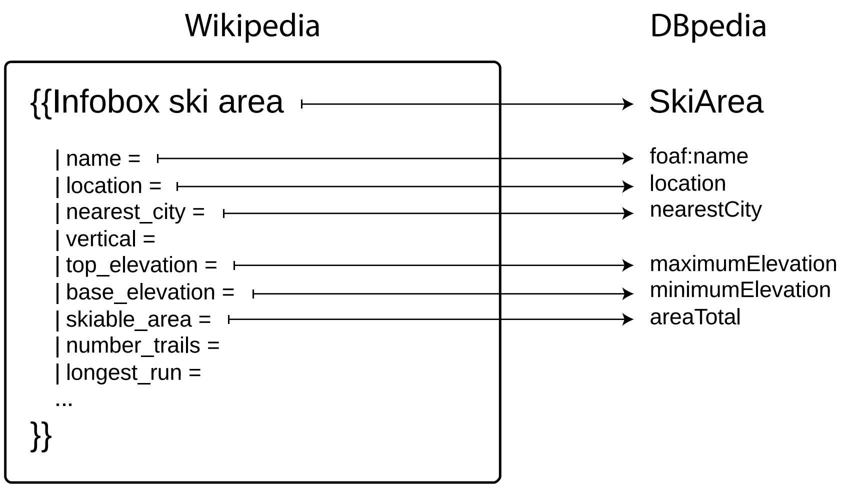


Figure 6.1: Example of DBpedia mapping

- Alignment of Wikipedia templates in different languages is often not possible, because there are no shared rules among the different Wikipedia communities on the management of infoboxes. In the example of Figure 6.2, `Carica_pubblica` only refers to politician, while `Officeholder` is more general.

- Properties may be mapped to different infoboxes in different languages. For example, the Italian DBpedia uses attributes of the `Bio` template to map generic biographical information, because specialized templates, such as `Carica_pubblica`, in the Italian Wikipedia do not contain generic information. This is not true in the English edition and in many other languages.

- Due to the previous point, some infoboxes are not mapped to any DBpedia class. This is the case of the `Persondata` template in English: since its information is repeated in the more specialized templates (for example, date of birth, name, occupation), the DBpedia annotators ignored it. A system that should align `Bio` and `Persondata`, and then transfer the mappings from English to Italian, would not map `Bio` to any DBpedia class since there is no mapping available for `Persondata`; therefore, all the generic biographical information would be lost.

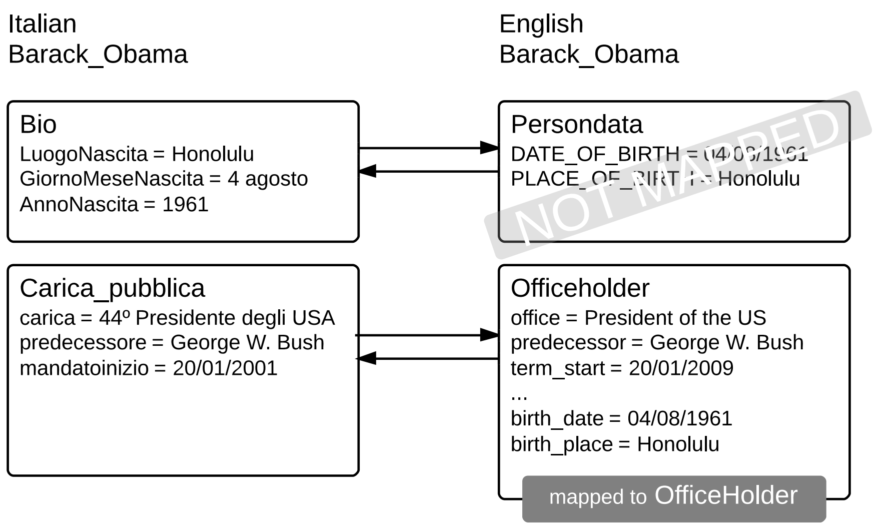

Italian
Barack_Obama
English
Barack_Obama
Bio
LuogoNascita = Honolulu
GiornoMeseNascita = 4 agosto
AnnoNascita = 1961
Persondata
DATE_OF_BIRTH = 04/08/1961
PLACE_OF_BIRTH = Honolulu
NOT MAPPED
Carica_pubblica
carica = 44º Presidente degli USA
predecessore = George W. Bush
mandatoinizio = 20/01/2001
Officeholder
office = President of the US
predecessor = George W. Bush
term_start = 20/01/2009
...
birth_date = 04/08/1961
birth_place = Honolulu
mapped to OfficeHolder

Figure 6.2: An example of infobox alignment

6.2 Workflow of the System

In this work, we propose an automatic system for generating DBpedia mappings. Formally, given an infobox I and an attribute A_I contained in I, our system maps the pair $\langle I, A_I \rangle$ to a relation R in the DBpedia ontology.

Our approach exploits the redundancy of Wikipedia across editions in different languages, assuming that, if values of a particular infobox attribute are similar to values of a particular DBpedia property, then we can map the attribute to the property.

This approach requires existing versions of DBpedia to train the system, in particular we exploit the English, German, French, Spanish, and Portuguese editions. Given a target language l, the system extracts the mappings between DBpedia properties and infobox atttributes in such language. Note that the target language l can also be included in the set of languages chosen as training data; however, in our experiments we do not use this approach since we are interested in building mappings for those chapters of Wikipedia for which the corresponding DBpedia does not exist yet. Our system consists of three main modules: pre-processing, mapping extraction, and post-processing. Figure 6.3 depicts the workflow of the system.

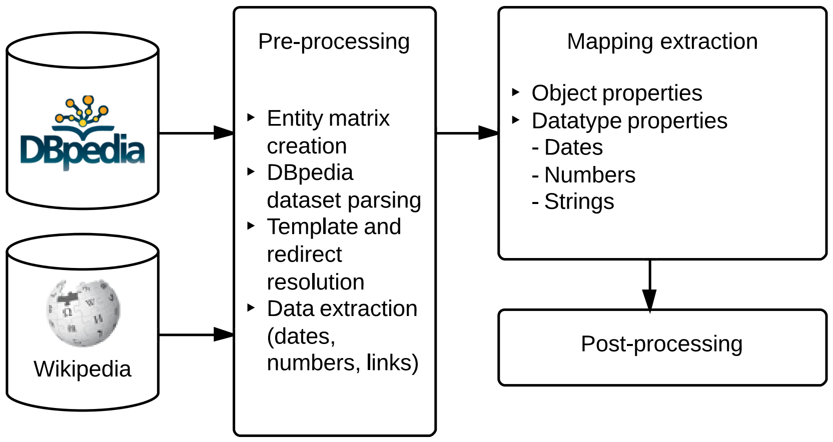


Figure 6.3: Workflow of the system

6.3 Pre-processing

This section describes how we collect and normalize the data needed for the mapping between DBpedia and Wikipedia.

6.3.1 Cross-language information

The proposed approach makes considerable use of the redundancy of information among different versions of Wikipedia. In particular, we focus on the semi-structured information contained in the infoboxes. For example, the English Wikipedia page of Barack Obama contains an infobox with his birth date, birth place, etc. The same information is often included in the infoboxes of the corresponding pages in other Wikipedia editions. To obtain the correspondences between pages in different editions of Wikipedia, we use the entity matrix described in Section 4.2. In the rest of this chapter, $P_{l_1}, P_{l_2}, \ldots$ denote the Wikipedia pages in languages $l_1, l_1, \ldots$, and P denotes the entity described by the corresponding row in the entity matrix.

6.3.2 DBpedia dataset extraction

DBpedia releases its ontology description in OWL format. The source file contains the description of the classes and properties, with all their characteristics. In our case, we search for the type (range) of each property. Depending on this feature, we can split them into two categories:

- *Datatype properties*, when the relation connects instances of classes to literals of XML (scalar values). For example `birthDate` connects a `Person` to a date.
- *Object properties*, when the relation connects instances of two classes (not necessarily different). For example, `birthPlace` connects a `Person` to a `Place` and `spouse` connects a `Person` to a `Person`.

Performing the mapping task, we use different strategies depending on the range of the property.

6.3.3 Template and redirect resolution

In Wikipedia, templates are particular pages created to be included into other pages (see Section 4.1). Infoboxes are a particular subset of templates that are usually rendered as a table in the upper-right corner of the corresponding Wikipedia article. Although this particular subset of templates is useful for information extraction from Wikipedia, only around 10% of templates belong to this category: the majority of them is used to give graphic coherence to the same types of elements in different articles. For example, countries are often shown in Wikipedia infoboxes as the flag of the country followed by the name. These templates are often used as values for the infobox attributes. Since different languages have different strategies in using templates, the alignment between values containing templates is not trivial. During the alignment phase, these discrepancies may lead to errors. To address this problem, we pre-process the attribute values using the *Bliki engine*,[1] a parser that converts templates to their expanded text. After this operation, templates such as {{`EGY`}} are rendered as the Egypt flag followed by the name of the country linked to its page.

6.3.4 Data Extraction

In our approach, the main difficulty consists in the comparison between data obtained from DBpedia and attribute values stored in Wikipedia infoboxes. This is due to the fact that DBpedia is strongly typed, while Wikipedia does not have an explicit type system. Attribute values often contain a mixture of dates, numbers, and text, represented, formatted, and approximated in different ways depending on the Wikipedia edition and on the users who edit articles. These types of data can be formatted in different ways in different languages. For example, in English, we can express a date using different patterns, such as, "June 4th, 1983", "04/06/1983", or even "06/04/1983." Furthermore, numeric values can be approximated using variable precision

[1]`https://code.google.com/p/gwtwiki/`

attribute	value
name	Diego Maradona
image	Maradona at 2012 GCC Champions League final.JPG
image_size	250
birth_place	[[Lans]], [[Buenos Aires province|Buenos Aires]], [[Argentina]]
birth_date	{{Birth date and age|1960|10|30|df=yes}}
height	{{height|m=1.65}}
youthyears1	19681969
youthyears2	19701974
youthyears3	19751976
...	...

Figure 6.4: `Infobox_football_biography` attributes for Diego Maradona

depending on a particular edition of Wikipedia. For instance, the total area of Egypt is 1,002,450 in the English Wikipedia and 1.001.449 in the Italian one, where both the value and the format are different.

To tackle these problems, we defined a function e that, using a set of heuristics for numbers and dates, extracts structured information for each attribute value. In particular, the domain of e is the plain text used in Wikipedia to express properties, while the range is a set of four different sets of elements: numbers, dates, links and text tokens.

In Figure 6.4 an example of `Infobox_football_biography` is presented. In the birth_place value, the value "[[Lans]], [[Buenos Aires province|Buenos Aires]], [[Argentina]]" of the attribute *birth_place* is converted into the bag of links $\{\texttt{Lans}, \texttt{Buenos\_Aires\_province}, \texttt{Argentina}\}$ and the set of tokens $\{\text{Lans}, \text{“,”}, \text{Buenos}, \text{Aires}, \text{“,”}, \text{Argentina}\}$, leaving the remaining sets (dates and numbers) empty. In the birth_date value, the template "Birth date and age" is parsed using the Bliki engine (see Section 6.3.3), resulting in "30 October 1960 (age 52)"; then, the string is converted into the set of dates {1960-10-30}, the set of numbers $\{30, 1960, 52\}$, and the set of tokens $\{30, \text{October}, 1960, (, \text{age}, 52,)\}$, leaving the links set empty.

6.4 Mapping extraction

In this section, we describe the matching algorithm used to determine whether an attribute A_I contained in the infobox I in Wikipedia can be mapped to a given property R in DBpedia. To find the mappings, we have to calculate the pairwise similarities between the elements in the set of all the possible attributes A_I and the elements in the set of all the possible properties R. The candidates are represented as pairs (A_I, R), the pairs with the highest similarity $S(A_I, R)$ are considered correct mappings. The similarity is an average result calculated using instance-based similarities between the values of property R in different DBpedia editions and the values of the attribute A_I in different Wikipedia pages in the target language. This process can lead to large number of comparisons to determine if a pair (A_I, R) can be mapped. The rest of the section provides a detailed and formal description of the algorithm.

Given a relation R in DBpedia in languages $L = \{l_1, l_2, \dots, l_n\}$ and a target language l, the algorithm works as follows.

(i) We build the following set, discarding entities that are not involved in the relation:

$$\begin{aligned}\Pi_R = \{P_{l_i} : P_{l_i} & \text{ has its corresponding } P_l \\ & \text{and exists at least an instance of } R \text{ in DBpedia in language } l_i.\}\end{aligned}$$

(ii) For each pair (A_I, R), we compute S_l:

$$S_l(A_I, R) = \frac{\sum_{P_{l_i} \in \Pi_R} \sigma_l(e(A_I, P_l), v(R, P_{l_i}))}{|\Pi_R|}$$

where the function σ_l is defined in Section 6.5 and the division by $|\Pi_R|$ is used to calculate the average similarity between attributes and properties based on their values in different languages.

(iii) All pairs A_I, R for which $S_l(A_I, R) < \lambda$ are discarded. Varying λ, we can change the trade-off between precision and recall.

(iv) For each infobox I, for which at least a pair (A_I, R) exists, we select A_I^* such that the pair (A_I^*, R) maximizes the function S.

(v) Finally, we obtain the set M_R of the selected pairs (A_I, R).

6.5 Inner similarity function

The inner similarity $\sigma_l(e(A_I, P_l), v(R, P_{l_i})) \to [0, 1]$ is computed between (i) the sets of values extracted and normalized by the function e defined in Section 6.3.4, starting from A_I in language l, and (ii) the values of R in the DBpedia editions in languages $l_1, l_2, \dots, l_n$, extracted by the function v. In sections 6.5.1 and 6.5.2, the function σ_l is formally defined depending on the two categories used to classify the property R (see Section 6.3.2). We use V_W and V_D to indicate the values returned by the functions e and v, respectively.

6.5.1 Similarity between object properties

When the range of the property R is an object, the value V_D corresponds to a Wikipedia page. Using the entity matrix E, we look for the equivalent page V_D^l in the target language l. Then, we search V_D^l in the links set of V_W, and we set $\sigma_l(V_D, V_W) = 1/k$ if we find it – k is the cardinality of the links subset of V_W. By dividing by k, we downgrade the similarity in case of partial matching. If the links set of V_W does not contain V_D^l, or if V_D does not have a corresponding article in the target language (and therefore V_D^l does not exist), we compare the string representations of V_D and V_W (see Section 6.5.2).

6.5.2 Similarity between datatype properties

When the range of the property R is not an object, we handle 9 types of data: calendar related (*date*, *gYearMonth*, *gYear*), numeric (*double*, *float*, *nonNegativeInteger*, *positiveInteger*, *integer*), and *string*. We discard the `boolean` type, as it affects only 4 properties out of 1,775, and it is never used in languages different from English.

Calendar related data.

Given the value V_D of type *date* and the set V_W, we compute $\sigma_l(V_D, V_W)$ by searching the day, the month and the year of V_D in the set V_W. We look at the date parts separately, because some Wikipedia editions split them into different infobox attributes. We assign a value of $1/3$ to each part of the date V_D that appears in V_W. In addition, the month is given only if it appears as text, or if it is included in the numbers set of V_W together with the day and the year. Similarly, we look at the day only if it appears with the month. Instead, the year usually would not cause ambiguity, therefore the match is considered also when present without day and month.

$$\sigma_l(V_D, V_W) = \begin{cases} 1 & \text{if day-month-year are present in } V_W \\ 2/3 & \text{if day-month are present in } V_W \\ 2/3 & \text{if month-year are present in } V_W \\ 1/3 & \text{if year is present in } V_W \end{cases}$$

Similarly, for *gYearMonth* we set $\sigma_l(V_D, V_W) = 1$ if both month and year appear in the dates set of V_W, and $\sigma_l(V_D, V_W) = 0.5$ if V_W contains only one of them. Finally, for *gYear* we set $\sigma_l(V_D, V_W) = 1$ if the year is included in the numbers set of V_W.

Numeric data.

While for calendar related data we expect to find the exact value, often properties involving numbers can have slightly different values in different languages (see Section 6.3.4 for an example). If $V_D = 0$, we check if the numbers subset of V_W contains 0. If true, then $\sigma_l(V_D, V_W) = 1$, otherwise $\sigma_l(V_D, V_W) = 0$. If $V_D \neq 0$, we search for values in V_W near to V_D, setting a tolerance $\nu > 0$. For each n in the numbers set of V_W, we calculate $\varepsilon = |V_D - n| / |V_D|$. If $\varepsilon < \nu$, then we set $\sigma_l(V_D, V_W) = 1$ and exit the loop. If the end of the loop is reached, we set $\sigma_l(V_D, V_W) = 0$.

Strings.

String kernels are used to compare strings. To compute the similarity, this family of kernel functions takes into account two strings and looks for contiguous and non-contiguous subsequences of a given length they have in common. Non contiguous occurrences are penalized according to the number of gaps they contain. Formally, let Σ be an alphabet of $|\Sigma|$ symbols, and $s = s_1 s_2 \ldots s_{|s|}$ a finite sequence over Σ (i.e., $s_i \in \Sigma, 1 \leqslant i \leqslant |s|$). Let $\mathbf{i} = [i_1, i_2, \ldots, i_n]$, with $1 \leqslant i_1 < i_2 < \ldots < i_n \leqslant |s|$, be a subset of the indices in s, we will denote as $s[\mathbf{i}] \in \Sigma^n$ the subsequence $s_{i_1} s_{i_2} \ldots s_{i_n}$. Note that $s[\mathbf{i}]$ does not necessarily form a contiguous n-gram of s. The length spanned by $s[\mathbf{i}]$ in s is $l(\mathbf{i}) = i_n - i_1 + 1$. The gap-weighted subsequences kernel (or string kernel) of length n is defined as

$$K_n(s,t) = \langle \phi^n(s), \phi^n(t) \rangle = \sum_{u \in \Sigma^n} \phi^n_u(s) \phi^n_u(t), \tag{6.1}$$

where

$$\phi^n_u(s) = \sum_{\mathbf{i}: u = s[\mathbf{i}]} \mu^{l(\mathbf{i})}, u \in \Sigma^n \tag{6.2}$$

and $\mu \in]0,1]$ is the decay factor used to penalize non-contiguous subsequences.[2] An explicit computation of Equation 6.1 is unfeasible even for small values of n. To evaluate more efficiently K_n, we use the recursive formulation based on a dynamic programming implementation [LStC02, STT02, CGGR03].

In our implementation, subsequences are n-grams (strings are tokenized), where $n = \min\{|V_D|, |V^*_W|\}$ and V^*_W is the tokenized set of V_W where some n-grams have been replaced with their translation when cross-language links exist. The similarity function is defined as the first strictly positive value returned by the following loop:

$$\sigma_l(V_D, V_W) = \frac{K_i(V_D, V^*_W)}{n - i + 1} \qquad \text{for each } i = n, n-1, \ldots, 1.$$

[2]Notice that by choosing $\mu = 1$ sparse subsequences are not penalized. The algorithm does not take into account sparse subsequences with $\mu \to 0$.

For example, the Italian Wikipedia page for the Mona Lisa famous painting contains the infobox attribute *tecnica* (technique) with value “Pittura a olio su tavola”. DBpedia has a corresponding property *type* with value “Oil on poplar”. Leaving the string as they are, whatever token-based string matching algorithm we use, the value $s_P(V_D, V_W)$ would be 0, as there are no words in common between the two strings. We collect the links in V_W and try to find them in the entity matrix. “Pittura a olio” is linked to *Pittura_ad_olio* that has a cross-language link to *Oil_painting*. Then, “tavola” is linked to *Tavola* that has a cross-language link to *Poplar*. We then replace, in V_W the tokens involved in the links with the found links in the language used in the DBpedia triple. In the example, we obtain the hybrid set of tokens “Oil painting su poplar”: now there are two common tokens, “oil” and “poplar”, therefore for our string matching algorithm $s_P(V_D, V_W)$ can be greater than 0.

6.6 Post-processing

Some infoboxes contain attributes with multiple values. For example, the musical genre of a particular album can be “rock” and “pop”, or a book can have more than one author. In these cases, Wikipedia provides more than one attribute describing the same relation, and adds an incremental index after the name of the attribute (sometimes also adding an underscore between the attribute name and the index). For example, the `Infobox_settlement` template contains the attribute `twinX` used for twin cities, where X can vary from 1 to 9. In our system, if M_R contains a mapping $A_I \to R$, we also add the set of mappings $A'_I \to R$ where the name of attribute A' differs from A only for an added or replaced digit. This filter is applied on the set M of mappings built in the mapping phase (Section 6.4) and is only used to increase recall.

6.7 Evaluation

Experiments have been carried on Italian, using existing DBpedia editions in five languages (English, Spanish, Portuguese, German, and French) as training data. To perform the evaluation, three annotators created a gold standard by manually annotating 15 infoboxes (for a total of 100 different attributes), randomly extracted from the first 100 most frequent infoboxes in the Italian Wikipedia. The inter annotator agreement is 91%, with respect to Fleiss’ kappa measure [Fle71]. The gold standard is available online on the Airpedia website.[3] As baseline, we use the manually mapped Italian infoboxes that can be downloaded from the DBpedia official website.[4] Specifically,

[3]http://www.airpedia.org/download/dbpedia-property-mappings-in-14-languages/
[4]http://mappings.dbpedia.org/

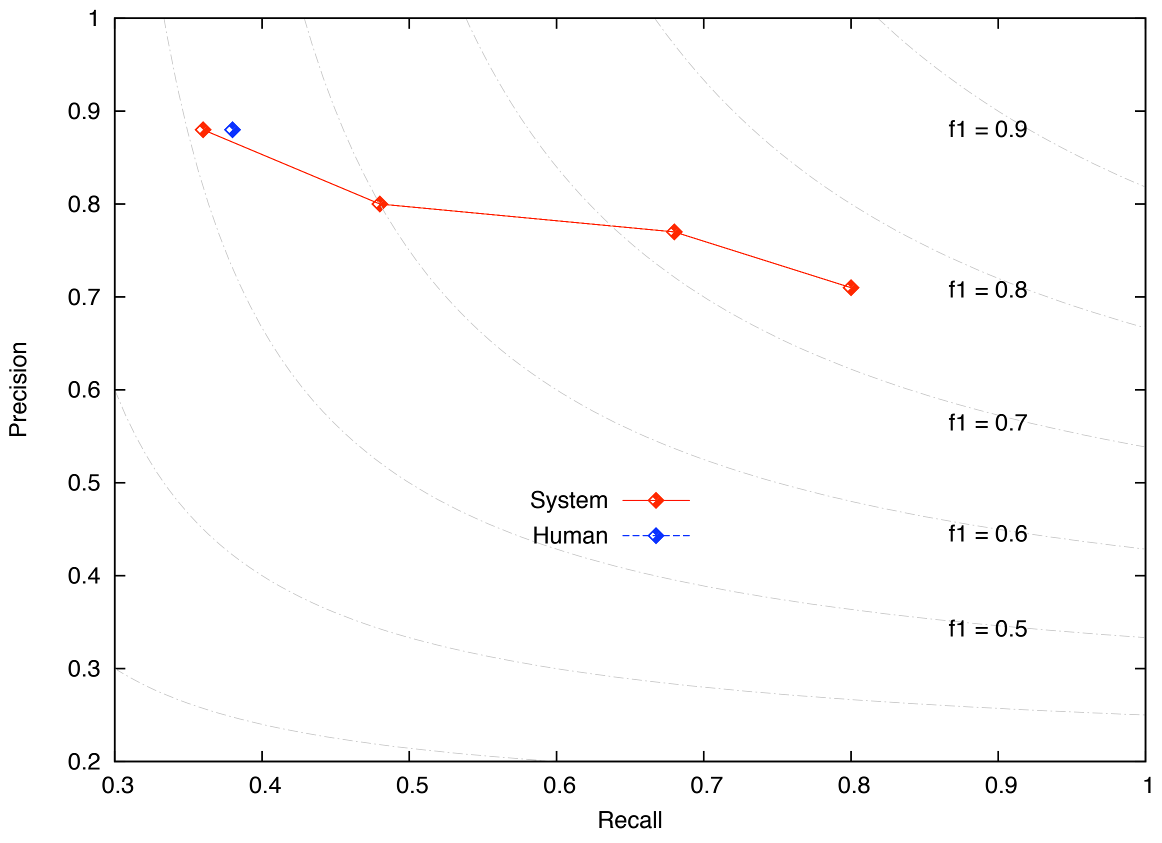


Figure 6.5: Precision/recall curve of our system compared with the DBpedia original manual mapping in Italian. From left to right, λ value is 0.9, 0.7, 0.5, and 0.3.

we used the version available on 5th April 2013, made available by the Italian DBpedia project,[5] consisting of around 50 infoboxes and 469 attributes (extracted from 18 of the 50 used infoboxes) mapped by one annotator during the spring 2012.

Figure 6.5 shows the precision/recall curve. Different precision/recall points are obtained by varying the parameter λ described in Section 6.4. The grey dashed lines join points with the same F_1. The results show that the coverage of the baseline (Human) is around 38% with a precision of around 88%. Our system is able to achieve comparable results in term of precision (87%), but it leads to a significant improvement in recall maintaining acceptable precision. Specifically, we can see that, by exploiting existing mappings, we can cover up to 70% of the attributes with a precision around 80%. Even though the procedure is not generally error-prone, these mappings can speedup the mapping process, because they can be used as a starting step for releasing new DBpedia editions (or extending existing ones). A subsequent step should only manually check the obtained mappings, in order to remove the wrong ones. Such task is quicker than the mapping one starting from scratch (i.e. Wikipedia).

[5]http://it.dbpedia.org/

7
Extending DBpedia coverage on classes

In Chapter 5, we provided a method to automate the mapping generation between Wikipedia infoboxes and DBpedia classes. The approach can be useful to start new DBpedia chapters and to extend the smallest ones, but considers only pages including an infobox. What happens when a page does not have it? Currently, these pages are not considered in the DBpedia ontology, and are classified as the top-level class `owl:Thing`.

In this chapter, we propose an automatic approach that exploits Wikipedia cross-language links to assign pages to a DBpedia class, even if they do not contain an infobox, extending the DBpedia coverage over Wikipedia in six different languages. Cross-language links are first used to add Wikipedia entities that are not included in a DBpedia edition for one language but present in others (see Section 4.2.2).

Finally, we boost the coverage by training a supervised kernel-based classifier using both the articles included in DBpedia and the ones extracted in the first stage, and then classify those articles for which cross-language links do not exist. Experiments have been performed on a dataset of 500 articles manually annotated by the authors. Starting from 5.6M total entities extracted from Wikipedia in the six languages, around 2.2M are added (with respect to the original DBpedia) using the first step. We show that our approach further increases the coverage of the DBpedia with high accuracy. Our algorithm can be tuned to have different tradeoffs between precision and recall. The resulting resource contains a total of nearly 4M entities, 1.7M of them not included in the original DBpedia for the six languages considered for the experiment.

7.1 Kernels for Entity Classification

The strategy adopted by kernel methods [STC04a, SS02] consists of splitting the learning problem in two parts. First they embed the input data in a suitable feature space, and then use a linear algorithm (e.g., the perceptron) to discover nonlinear patterns in the input space. Typically, the mapping is performed implicitly by a so-called *kernel function*. The kernel function is an inner product, which can intuitively be considered as a similarity measure between the input data, that depends on the specific data type and domain. A typical similarity function is the inner product between feature vectors. Characterizing the similarity of the inputs plays a crucial role in determining the success or failure of the learning algorithm, and it is one of the central questions in the field of machine learning.

Formally, the kernel is a function $k : X \times X \to \mathbb{R}$ that takes as input two data objects (e.g., vectors, texts, parse trees) and outputs a real number characterizing their similarity, with the property that the function is symmetric and positive semi-definite. That is, for all $x_1, x_2 \in X$, it satisfies

$$k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle,$$

where ϕ is an explicit mapping from X to an (inner product) feature space $\mathcal{F}$.

In the remainder of this section, we define and combine different kernel functions that calculate the pairwise similarity between entities using their corresponding Wikipedia articles as source of information. They are the only domain specific elements of our classification system, while the learning algorithm is a general purpose component. Many classifiers can be used with kernels, we use k-nearest neighbor (k-nn).

7.1.1 Bag-of-features Kernels

The simplest method to calculate the similarity between two entities is to compute the inner product of their vector representation in the vector space model (VSM). Formally, we define a space of dimensionality N in which each dimension is associated with one feature, and the entity e is represented in the language $l_j \in \mathcal{L}$ by a row vector

$$\phi_j(e_i) = (w(f_1, E_{i,j}), w(f_2, E_{i,j}), \ldots, w(f_N, E_{i,j})), \tag{7.1}$$

where the function $w(f_k, E_{i,j})$ records whether a particular feature f_k is active in the Wikipedia article $E_{i,j}$. Using this representation we define the *bag-of-features kernel* between entities as

$$K_F(e_1, e_2) = \sum_{j=1}^{|\mathcal{L}|} \langle \phi_j(e_1), \phi_j(e_2) \rangle, \tag{7.2}$$

Notice that this kernel computes the similarity between e_1 and e_2 as the sum of their similarities in those languages for which Wikipedia articles exist. Based on this general formulation, we define 4 basic kernel functions as follows.

Bag-of-templates Kernel.

To define the similarity between pairs of entities, we count how many occurrences of templates their corresponding Wikipedia articles in a specific language share. Templates are commonly used for boilerplate messages, standard warnings or notices, infoboxes, navigational boxes and similar purposes. In our experiments, we take into consideration solely the infoboxes (Section 7.2.1 describes the set of heuristics used to extract the infoboxes). The *Bag-of-templates kernel* (K_T) is defined as in Equation (7.2), where the function $w(f_k, E_{i,j})$ in Equation (7.1) is a binary function that records whether a particular infobox f_k is used in the Wikipedia article $E_{i,j}$.

Bag-of-categories Kernel.

Wikipedia categories are intended to group together articles on similar subjects and have proven useful in text classification [WHZC09], ontology learning [NS08], and ontology population [SKW07]. The *bag-of-categories kernel* (K_C) is defined as in Equation (7.2) where the function $w(f_k, E_{i,j})$ in Equation (7.1) is a binary function that records whether a particular category f_k is used in the Wikipedia article $E_{i,j}$.

Bag-of-sections Kernel.

Wikipedia articles are structured in several sections that might provide relevant cues for classification. For example, biographical articles typically include sections like *Early life*, *Career*, and *Personal life*; while articles referring to cities usually include sections like *Places of interest*, *Demographic evolution*, and *Administration*. The *bag-of-sections kernel* (K_C) is defined as in Equation (7.2) where the function $w(f_k, E_{i,j})$ in Equation (7.1) is a binary function that records whether a particular section name f_k is used in the Wikipedia article $E_{i,j}$.

Bag-of-words Kernel.

The use of infoboxes, categories, and sections ensures highly accurate classification, however it produces extremely sparse feature spaces that compromises the recall. To overcome this problem, we also exploit content words of the text article as additional sources of information. The *bag-of-words kernel* (K_W) is defined as in Equation (7.2) where the function $w(f_k, E_{i,j})$ in

Equation (7.1) is the standard *term frequency - inverse document frequency* (tf $\times$ idf) of the word f_k in the Wikipedia article $E_{i,j}$.

7.1.2 Latent Semantic Kernel

Given that the bag-of-words representation does not deal well with lexical variability, in the following we introduce the latent semantic kernels and show how to define an effective semantic VSM using (unlabeled) external knowledge. It has been shown that semantic information is fundamental for improving the accuracy and reducing the amount of training data in many natural language tasks, including fine-grained classification of named entities [FH02, Giu09], question classification [LR05], text categorization [GS05], word sense disambiguation [GGS05].

In the context of kernel methods, semantic information can be integrated considering linear transformations of the type $\tilde{\phi}_j(c_t) = \phi_j(c_t)\mathbf{S}$, where $\mathbf{S}$ is a $N \times k$ matrix [STC04a]. The matrix $\mathbf{S}$ can be rewritten as $\mathbf{S} = \mathbf{WP}$, where $\mathbf{W}$ is a diagonal matrix determining the word weights, while $\mathbf{P}$ is the *word proximity matrix* capturing the semantic relations between words. The proximity matrix $\mathbf{P}$ can be defined by setting non-zero entries between those words whose semantic relation is inferred from an external source of domain knowledge. The *semantic kernel* takes the general form

$$\tilde{k}_j(e_1, e_2) = \phi_j(e_1)\mathbf{SS}'\phi_j(e_2)' = \tilde{\phi}_j(e_1)\tilde{\phi}_j(e_2)'. \tag{7.3}$$

It follows directly from the explicit construction that Equation (7.3) defines a valid kernel.

To define the proximity matrix for the latent semantic kernel, we look at co-occurrence information in a (large) corpus. Two words are considered semantically related if they frequently co-occur in the same texts. We use singular valued decomposition (SVD) to automatically derive the proximity matrix Π from a corpus, represented by its term-by-document matrix $\mathbf{D}$, where the $\mathbf{D}_{i,j}$ entry gives the frequency of term p_i in document d_t.[1] SVD decomposes the term-by-document matrix $\mathbf{D}$ into three matrixes $\mathbf{D} = \mathbf{U}\Sigma\mathbf{V}'$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices (i.e., $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\mathbf{V}'\mathbf{V} = \mathbf{I}$) whose columns are the eigenvectors of $\mathbf{DD}'$ and $\mathbf{D}'\mathbf{D}$ respectively, and Σ is the diagonal matrix containing the singular values of $\mathbf{D}$. Under this setting, we define the proximity matrix Π as follows:

$$\Pi = \mathbf{U}_k\Sigma_k,$$

where $\mathbf{U}_k$ is the matrix containing the first k columns of $\mathbf{U}$ and k is the dimensionality of the latent semantic space and can be fixed in advance. By using a small number of dimensions, we

[1] SVD has been first applied to perform latent semantic analysis of terms and latent semantic indexing of documents in large corpora by [DDL+90].

can define a very compact representation of the proximity matrix and, consequently, reduce the memory requirements while preserving most of the information.

The matrix Π is used to define a linear transformation $\pi_j : \mathbb{R}^N \to \mathbb{R}^k$, that maps the vector $\phi_j(e_t)$, represented in the standard VSM, into the vector $\tilde{\phi}_j(e_t)$ in the latent semantic space. Formally, π_j is defined as follows

$$\pi_j(\phi_j(e_t)) = \phi_j(e_t)(\mathbf{W}\Pi) = \tilde{\phi}_j(e_t), \tag{7.4}$$

where $\phi_j(e_t)$ is a row vector, $\mathbf{W}$ is a $N \times N$ diagonal matrix determining the word weights such that $\mathbf{W}_{i,i} = \log(\mathrm{idf}(w_i))$, where $\mathrm{idf}(w_i)$ is the *inverse document frequency* of w_i.

Finally, the *latent semantic kernel* is explicitly defined as follows

$$K_L(e_1, e_2) = \sum_{j=1}^{|\mathcal{L}|} \langle \pi_j(\phi_j(e_1)), \pi_j(\phi_j(e_2)) \rangle,$$

where ϕ_j is the mapping defined in Equation (7.1) and π_j is the linear transformation defined in Equation (7.4) in language $l_j \in \mathcal{L}$. Note that we have used a series of successive mappings each of which adds some further improvement to the entity representation.

7.1.3 Composite Kernel

Having defined all the basic kernels, representing different characteristics of entity descriptions, we finally define the composite kernel as

$$K_{\mathrm{COMBO}}(e_1, e_2) = \sum_{n=1} \frac{K_n(e_1, e_2)}{\sqrt{K_n(e_1, e_2) K_n(e_1, e_2)}}, \tag{7.5}$$

where K_n is a valid basic kernel. The individual kernels are normalized. This plays an important role in allowing us to integrate information from heterogeneous feature spaces.

7.2 Experiments

In this section, we evaluate different setups on the task of DBpedia expansion for six languages (English, Italian, German, French, Spanish, and Portuguese). The evaluation only concerns the second stage of our approach, because the first stage has precision almost 100% (see Section 7.2.1).

	English	Italian	German	French	Spanish	Portuguese
Wikipedia	2012-10-01	2012-09-21	2012-10-09	2012-10-07	2012-09-27	2012-10-06
DBpedia	2012-06-04	2012-10-12	2012-06-04	2012-06-04	2012-06-04	2012-06-04

Table 7.1: Versions of DBpedia and Wikipedia used for our tests

7.2.1 Pre-processing Wikipedia and DBpedia

Our experiments and results refer to the versions of Wikipedia and DBpedia available when this work started in mid October 2012. Table 7.1 lists the dumps used.

Wikipedia.

We parsed the dump files to extract information about each single article and we built the matrix E using cross-language links (see Section 4.2). We manually check the accuracy of these links on 100 random pages: all of them were correct, so we can assume that the precision of this step is 100%. The matrix E build upon six languages (English, Italian, German, French, Spanish, and Portuguese) contains 5,626,515 entities.

As we only want infoboxes for our classification, we use the algorithms described in Section 4.1.

DBpedia.

Starting from DBpedia dumps, we created a mapping that combines the entities in E with the ontology class(es) they belong to. Using entities instead of Wikipedia pages allows us to automatically extend and improve the DBpedia coverage. For instance, *Michael Jackson* is classified as a `Person` in the Italian and German DBpedia, an `Artist` in the English DBpedia and a `MusicalArtist` in the Spanish DBpedia. The most specific class is the last one, so the *entity* Michael Jackson becomes `MusicalArtist` in every language. The final mapping contains 2,193,742 entities: comparing this figure with the size of the matrix E, this means that there are around 3,432,773 entities in Wikipedia that are not classified in DBpedia. In our experiments we always refer to this set for the classification part that makes use of kernel methods. Data concerning the enriched DBpedia is shown in Table 7.2.

7.2.2 Benchmark

Experiments are carried out on a benchmark extracted from the entity matrix introduced in Section 4.2. Specifically, the data set contains 400 randomly extracted entities not already present in DBpedia in any language. The data set is split in development set (100 entities) and test

	Matrix E	EN	IT	DE	FR	ES	PT
Wikipedia	5,626,515	3,932,148	924,544	1,325,792	1,251,585	953,848	740,585
DBpedia	-	1,716,555	607,842	205,903	15,463	15,987	226,497
DBpedia CL	2,193,742	1,902,585	652,395	482,747	518,874	419,168	430,603
Not classified	3,432,773	2,029,563	272,149	843,045	732,711	534,680	309,982

Table 7.2: Total number of pages in Wikipedia, in DBpedia, and in DBpedia after using Wikipedia cross-language links. Quantities in the last row represent, for each language, the number of pages not included in DBpedia in any language considered

set (300 entities). All entities have been annotated with the most specific available class in the version 3.8 of the DBpedia ontology by the author of this thesis. 50 more entities have been annotated by three different annotators, resulting in an inter-agreement of 78% (Fleiss' kappa measure, see [Fle71]). An additional `Unknown` class has been introduced to annotate those entities that cannot be assigned to any class in the ontology. When an entity is assigned to a class, it is also implicitly assigned to all its super-classes. For instance, classifying *Michael Jackson* as a `MusicalArtist` we implicitly classify him as `Artist`, `Person` and `Agent`.

The evaluation is performed as proposed by [MR00] (see Section 5.2) for a similar hierarchical categorization task. In the example above, classifying *Michael Jackson* as an `Athlete`, we obtain a false positive for this wrong classified class, two false negatives for missing classes `MusicalArtist` and `Artist`, and two true positives for `Person` and `Agent`.

7.2.3 Latent Semantic Models

For each language, we derive the proximity matrix Π (Section 7.1) from the 200,000 most visited Wikipedia articles. After removing terms that occur less than 5 times, the resulting dictionaries contain about 300,000 terms. We use the SVDLIBC package[2] to compute the SVD, truncated to 100 dimensions.

7.2.4 Learning Algorithm

We use a k-nn classifier[3] to classify novel entities into the DBpedia ontology. The optimization of the parameter k is performed on the development set, and $k = 10$ results as the best choice, because it maximizes the F_1 value. Entities are classified by a majority vote of their neighbors. To change the tradeoff between precision and recall, we set the minimum number of votes z

[2] `http://tedlab.mit.edu/~dr/svdlibc/`

[3] During the preliminary experiments, we used our algorithms with two test classes: `Person` and `Place`. In this phase, Support Vector Machine (SVM) produced very good results. When we applied our approach to the entire DBpedia ontology (359 classes), SVM performance dramatically dropped.

($1 \leqslant z \leqslant k$) a class needs to obtain to be assigned. The algorithm has maximum precision with $z = k$, maximum recall with $z = 1$, and maximum F_1 with $z = 8$.

To train the classifier, we randomly select 100,000 entities from the matrix E included in DBpedia. Each entity is then labelled according to the corresponding DBpedia class.

7.2.5 Classification Schemas

We compare three alternative training and classification schemas.

Bottom-up. A single training and classification step is performed. k-nn is trained using entities annotated with the most specific classes in DBpedia. In classification, an entity is annotated with the finer-grained class c if c receives $v_c \geqslant z$ votes; `Unknown` otherwise.[4] Note that the algorithm also considers the super-classes of c: if a fine-grained class cannot be assigned with a given confidence level z, it could return a more generic one s ($s \subseteq c$) such that $v_s \geqslant z$. For instance, if $z = 10$ and the 10 votes are divided 5 to `Astronaut` and 5 to `Scientist`, our system answers `Unknown` because none of the classes obtains 10 votes. However, ascending the ontology, we find that the class `Person` receives 10 votes, as both `Astronaut` and `Scientist` belong to it. The system then classifies it as `Person`, instead of `Unknown`. In case this process does not find any class at any level with a sufficient number of votes, the `Unknown` answer is given.

Top-down. Multiple training and classification steps are performed. k-nn is trained using entities annotated with the most generic classes in DBpedia (ontology top-level). In classification, an entity is annotated with a generic class c if it receives $v_c \geqslant z$ votes; `Unknown` otherwise. The procedure is recursively repeated on all subtrees of the ontology using the previous classification to limit the number of classes to consider.

Hybrid. This variant consists in first training a k-nn as defined in the Bottom-up schema. Then, a set of specialized k-nns are trained for the most populated classes, such as, `Person`, `Organisation`, `Place`, `Work`, etc. In classification, let P be one of these classes, the Bottom-up schema is applied first. Then, if an entity is annotated with the class c such that $c \in P$, then a specialized k-nn is applied.

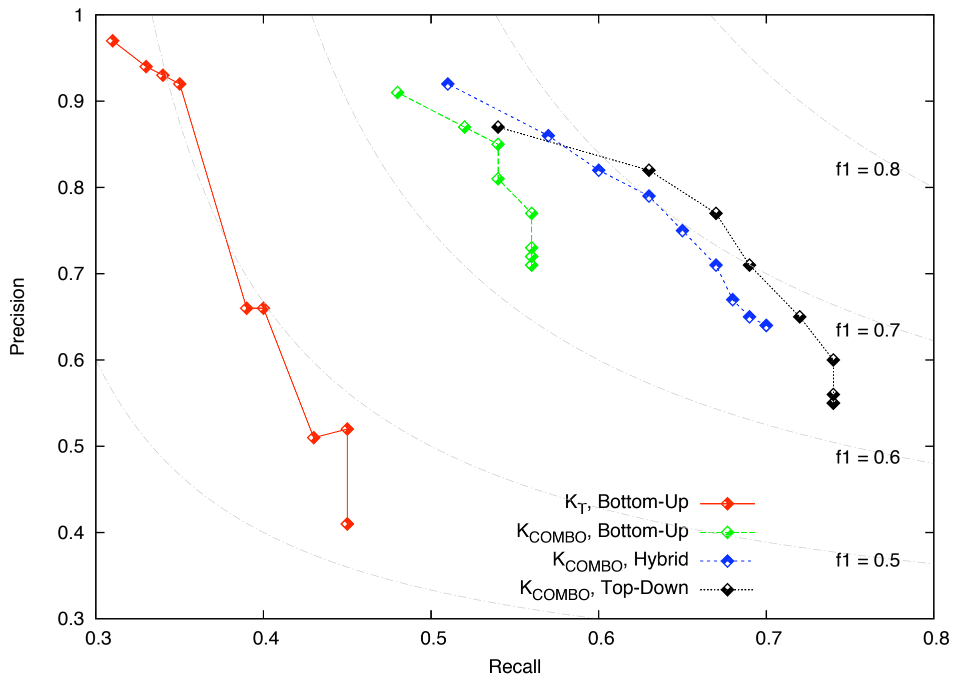


Figure 7.1: Precision/recall curve of the system

	MF	K_T	K_C	K_S	K_W	K_L	K_{COMBO}
Precision	0.35	0.97	0.90	0.94	0.81	0.84	0.91
Recall	0.38	0.31	0.40	0.16	0.22	0.41	0.48
F_1	0.31	0.47	0.55	0.27	0.34	0.55	0.63

Table 7.3: Results of the most frequent class baseline (MF), the basic kernels (see Section 7.1.1) and the composite kernel K_{COMBO}, using $z = 10$

7.2.6 Results

First, we investigate the contribution of the kernel combination (Section 7.1) and then the one of the different training and classification schemas (Section 7.2.5).

Table 7.3 reports the results of the most frequent class baseline, the basic kernels (K_T, K_C, K_S, K_W, and K_L), and the composite kernel K_{COMBO}. The experimental results show that the composite kernel K_{COMBO} significantly outperforms the basic kernels. We use approximate randomization [Nor89] to assess the statistical significance between the obtained results (p-value = 0.05).

Figure 7.1 shows the precision/recall curves obtained by varying the parameter z. We also draw, in grey in the background, the contour lines joining points with the same F_1, so that one

[4]Assigning the class Unknown is equivalent to abstention.

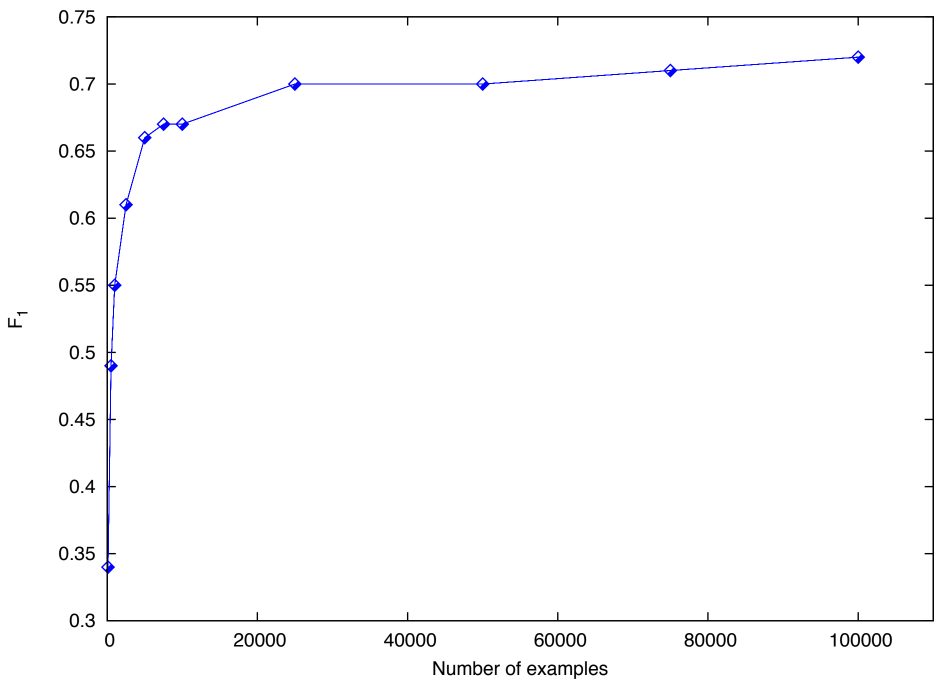


Figure 7.2: Learning curve of the system

can quickly visualize this value. Four different setups are compared in order to determine the one that produces the best tradeoff between precision and recall.

K_T, Bottom-up uses only the template information (as in the DBpedia framework) and the Bottom-up schema, obtaining the maximum precision of 97% at the expense of low recall of 31% ($z = 10$).

K_{COMBO}, Bottom-up uses all the sources of information and the Bottom-up schema, obtaining a significant improvement in recall (48%) preserving a high precision of 91% ($z = 10$).

K_{COMBO}, Hybrid uses all the sources of information and the Hybrid schema, obtaining a further improvement of precision (92%) and recall (51%).

K_{COMBO}, Top-down uses all the sources of information and the Top-down schema, obtaining the maximum recall (54%), however the precision (87%) is significantly lower than the one obtained in the other experiments.

Figure 7.2 shows the learning curve of the system in terms of F_1 in the configuration that maximizes the F_1 score (in our experiments, this happens in all configurations, when $z = 8$).

Finally, we perform some preliminary error analysis. Errors mostly depend on the following factors: (i) the Wikipedia article is too short; (ii) an appropriate class for the entity does not exist (this often happens with common nouns); (iii) some Wikipedia pages represent lists (for example, `Liste_des_conseillers...`) and our system often classifies them as the objects listed (in the example, `Person`); (iv) nesting of DBpedia classes is not optimal (for example, `Astronaut` and `Scientist` are disjoint classes). The most common factor is (iii), as it is the cause of more than half of the errors in the experiments on the test set. A subsequent work [PB13] tackles this problem, by first split the set of Wikipedia pages in “typeable” and “non-typeable”.

8
Extending DBpedia coverage on properties

In this chapter, we extend the approach used in Chapter 7 to properties. To populate the whole DBpedia set of properties (over 1,700 in version 3.8), we need to find the relevant information right from the page article, using Natural Language Processing (NLP) tools. This is a relation extraction (RE) task, i.e. the identification in a text of relevant entities and relationships between them. For example, given the sentence "Barack Obama was born on August 4, 1961", we need to identify "Barack Obama" as a named entity of type person, the value "August 4, 1961" as a date, and the `birthDate` relation between the two entities.

Supervised machine learning techniques are widely used to approach the RE task, but the lack of manually annotated texts to use as training data often limits the applicability of such techniques. In 2009, a new paradigm, called *distant supervision* [MBSJ09], has been proposed to deal with the limited availability of manually annotated data. The intuition behind distant supervision is that any sentence containing a pair of entities that participate in a known DBpedia property relation is likely to express such relation in some way. Using the example above, the assumption is that a sentence that includes both "Barack Obama" and "August 4, 1961" is expressing the `birthDate` relation. Since there are thousands of such pairs of entities in the DBpedia resource, we can extract very large numbers of (potentially noisy [RYM10]) examples for each relation.

In this work, we first collect this set of sentences starting from DBpedia and extracting the relevant sentences from the corresponding Wikipedia articles. Then, we train a RE tool (jSRE [GLR07], freely available on the web) using positive and negative examples extracted from such sentences. Finally, we apply the model on unseen text articles and extract the relations contained in the sentences.

We evaluate our system on seven DBpedia properties, using cross-validation over a small set of pages excluded from the training, demonstrating the suitability of the approach with high precision and recall.

8.1 Workflow

As introduced before, the work presented in this thesis relies on the intuition that jointly exploiting interlinked structured and unstructured data sources can offer a great potential for both NLP and Semantic Web applications. In particular, we focus on the pair Wikipedia-DBpedia as corpus-KB and on distant supervision as paradigm.

The distant supervision paradigm is based on the assumption that there is a high probability that the structured information present in the *infobox* is also expressed using natural language sentences in the same Wikipedia page. Given for example the triple

$$\langle \text{Barack Obama} \rangle \, \langle \text{is\_born\_in} \rangle \, \langle \text{Honolulu} \rangle \, ,$$

we expect that there is a sentence in the text article expressing the same relation with the same entity mentions, like

> Obama was born on August 4, 1961 at Kapiolani Maternity & Gynecological Hospital in Honolulu, Hawaii, and is the first President to have been born in Hawaii.

Summarizing, for each DBpedia property we apply the following procedure:

(i) all the triples expressing the relation are considered;

(ii) for each triple, the corresponding Wikipedia article is analyzed using Stanford CoreNLP (Section 8.2);

(iii) the sentences containing both the subject and the object of the triple are extracted and collected as positive examples (Section 8.2.1);

(iv) a set of negative examples is collected, too (Section 8.2.2);

(v) a RE tool is trained using the dataset built according to the procedure outlined above (Section 8.2.3);

(vi) the trained model is then applied to extract the desired relation from article pages where the infobox is missing or where the infobox does not contain such relation.

DBpedia	Stanford	DBpedia	Stanford
`Person`	PER	`date`	DATE
`Organisation`	ORG	`integer`	NUMBER
`Place`	LOC	`nonNegativeInteger`	NUMBER
`gYear`	DATE	`double`	NUMBER
`positiveInteger`	NUMBER	`[all other classes]`	MISC

Table 8.1: Type conversion between Stanford NER and DBpedia

The main part of the procedure is the RE task. Formally, given a sentence S which consists of a sequence of words, we need to find a relation R that involves two sequences of words E_1 and E_2, respectively the subject and the object of the relation. In our experiments, we use jSRE,[1] a state-of-the-art open source RE tool, made freely available on the web.

To assess to performance of the approach, we test the accuracy of the entire workflow on a dataset consisting of seven DBpedia properties (Section 8.3).

8.2 Pre-processing

The preliminary phase of our approach consists in collecting Wikipedia pages where a particular relation is (likely to be) expressed. The list of such pages can be easily extracted from DBpedia.

Given the Wikipedia page, the plain text article is obtained by removing tables, images and all the wiki markup. We use JWPL[2] for such purpose.

The cleaned up text is then analyzed using Stanford CoreNLP[3] with the following processors: tokenization, sentence splitting, part-of-speech tagging, lemmatization and Named Entity Recognition (NER). In particular, the NER module of Stanford CoreNLP annotates persons, organizations, locations, numbers and dates. In addition, we use the Stanford CoreNLP tag MISC for all the other DBpedia classes (`Work`, `Event`, and so on). Finally, we connect each of these types to the corresponding DBpedia type/class. Boolean properties are not considered as they affect only 4 relations out of over 1,700 in the DBpedia ontology. See Table 8.1 for more information.

8.2.1 Retrieving sentences

Given an instance of a DBpedia relation, e.g. ⟨Barack Obama⟩ ⟨is_born_in⟩ ⟨Honolulu⟩, we examine the Wikipedia article text of the subject (i.e., "Barack Obama") looking for the two entities involved in the relation.

[1] `http://hlt.fbk.eu/en/technology/jSRE`
[2] `https://code.google.com/p/jwpl/`
[3] `http://nlp.stanford.edu/software/corenlp.shtml`

First, the sentences of the given article are pre-processed as described in Section 8.2 and we identify those sentences containing entities of the types involved in the relation. In the above example, the domain of ⟨is_born_in⟩ is `Person`, while its range is `Place`.

Then, we collect all the sentences containing entities classified both with domain and range types of the relation, where the corresponding strings of the triple match.

In the above example

> Obama was born on August 4, 1961 at Kapiolani Maternity & Gynecological Hospital in Honolulu, Hawaii, and is the first President to have been born in Hawaii

Stanford CoreNLP annotates the sentence in the following way:

> ⟨PER⟩Obama⟨/PER⟩ was born on ⟨DATE⟩August 4, 1961⟨/DATE⟩ at ⟨ORG⟩Kapiolani Maternity & Gynecological Hospital⟨/ORG⟩ in ⟨LOC⟩Honolulu⟨/LOC⟩, ⟨LOC⟩Hawaii⟨/LOC⟩, and is the first President to have been born in ⟨LOC⟩Hawaii⟨/LOC⟩.

The sentence contains both ⟨PER⟩ and ⟨LOC⟩, the conversion of domain and range type of relation ⟨is_born_in⟩ in DBpedia, respectively (see Section 8.2). Therefore it can be a candidate as a positive example for the relation. While there is a ⟨LOC⟩ part containing the range of the relation (Honolulu), the complete string of the domain (Barack Obama) never appears in the sentence, so an approach based on exact string matching would erroneously discard this sentence. To avoid this behavior and increase the recall of this extraction step, we apply different matching strategies. First of all, we perform the exact match of the entire string as provided by Wikipedia. If the algorithm does not find such string in the sentence, we clean it deleting the part between brackets, used to disambiguate pages with the same title, as in "Carrie (novel)" and "Carrie (1976 film)". We use the resulting string for matching the domain in the sentence. If this fails, the original string is tokenized and, given the set of obtained tokens, new strings are built by combining the tokens (preserving the original word order). For instance, starting from "John Fitzgerald Kennedy", we obtain the new strings "John Fitzgerald", "John Kennedy", "Fitzgerald Kennedy", "John", "Fitzgerald" and "Kennedy". Using this rule, in our example we can identify the ⟨PER⟩ part containing the string "Obama".

For numeric entities, we do not use exact matching between the value stored in DBpedia and the number extracted by Stanford CoreNLP, as for some relations (such as `populationTotal`) they may be slightly different. Given two numeric values a and b, we then consider a positive match between them when a and b are both different from 0, and the ratio $|a - b| / |b|$ is less than 0.05 (5%).

8.2.2 Selecting sentences

Supervised machine learning tools need annotated data to be trained. Training (and test) sets consist of both *positive* and *negative* examples: the former are examples where the relation is present; the latter are examples where the relation is not expressed. The distant supervision paradigm uses structured data to collect positive examples, following the assumption that, if two entities participate in a relation, then all the sentences containing the two entities express such relation; however, this is not always true [RYM10] (see below).

In our experiment, we use the hypothesis that a sentence containing the domain part of the relation, and not containing its range but another entity of the type of the range, is a good negative example for the relation ⟨is_born_in⟩. For example, the sentence

> Following high school, Obama moved to Los Angeles in 1979 to attend Occidental College.

contains “Obama”, and does not contain “Honolulu” but contains “Los Angeles”. Therefore we pick this sentence as a negative example for the relation.

This simple rule is sufficient for building a training set for relations where there is no ambiguity. For example, in a biographical article in Wikipedia the date of birth is usually used in the birth date relation only. In addition, other dates in the same sentences certainly refer to different relations, as the birth date of a person is unique.

However, there are relations where these assumptions are not necessarily true, since the same pair subject-object can be involved in more than one relation. Therefore, we apply different strategies for the extraction of the training data.

First strategy: positives cannot be negatives.

In the sentence

> Obama was born on August 4, 1961 at Kapiolani Maternity & Gynecological Hospital in Honolulu, Hawaii, and is the first President to have been born in Hawaii.

the entity “Honolulu” refers to the birth place. Unfortunately, also the other ⟨LOC⟩ instance (“Hawaii”) refers to the birth place (although it may not be included in the DBpedia resource). To avoid this problem, when collecting our training set we discard potential negative examples taken from a sentence already used to extract a positive one.

Second strategy: only one sentence per relation.

In the sentence

> In 1971, Obama returned to Honolulu to live with his maternal grandparents, Madelyn and Stanley Dunham.

both "Obama" and "Honolulu" are present but the relation between them is different from birth place.

[RYM10] tackle this problem by assuming that, if two entities participate in a relation, *at least one sentence* that mentions these two entities might express that relation. Using this assumption, they trained a graphical model with the optimization of the parameters to ensure that predictions will satisfy a set of user-defined constraints. In their work, they use the New York Times corpus, where pages are less standardized than in Wikipedia. In our experiment we can rely on a stronger assumption: if two entities participate in a relation, *there is one and only one sentence* expressing such relation. We can then discard those pages not complying with this assumption, i.e. having more than one sentence containing both subject and object of the relation.

Third strategy: only one relation for each value.

Finally, we can take advantage from the complete set of properties available in DBpedia, by removing from the training set those pages having more than one relation sharing the same object. For instance, the two relations

$$\begin{array}{lll} \langle\text{Mark Zuckerberg}\rangle & \langle\text{work\_for}\rangle & \langle\text{Facebook}\rangle \\ \langle\text{Mark Zuckerberg}\rangle & \langle\text{founded}\rangle & \langle\text{Facebook}\rangle \end{array}$$

involve the same pair subject-object, therefore we cannot disambiguate whether a sentence in Mark Zuckerberg's Wikipedia page containing both his name and the company he founded refers to the former or to the latter relation.

8.2.3 Training algorithm

As learning algorithm, we use jSRE, a state-of-the-art RE tool described in [GLR07]. The RE task is treated as a classification problem in supervised learning, using kernel methods [STC04b] to embed the input data into a suitable feature space, and then run a classical linear algorithm to discover nonlinear patterns. The learning algorithm used is Support Vector Machines (SVM) [Vap99, CST10].

Property	Instances	Without strategies			With strategies		
		P	R	F_1	P	R	F_1
`birthDate`	258,609	0.92	1.00	0.95	0.91	1.00	0.95
`capital`	3,921	0.82	0.67	0.74	-	-	-
`deathDate`	82,060	0.94	0.99	0.96	0.93	1.00	0.96
`headquarter`	27,283	0.71	0.84	0.77	0.81	0.80	0.80
`populationTotal`	237,701	0.68	0.86	0.76	0.70	0.87	0.78
`region`	66,269	0.87	0.90	0.88	0.91	0.91	0.91
`deathPlace`	123,705	0.51	0.61	0.56	0.59	0.77	0.67

Table 8.2: Evaluation of the system on seven DBpedia properties

The tool uses two families of kernels: *global context kernel* and *local context kernel*. The first one adapts the ideas in [BM05] and uses a bag-of-words of three sets of tokens: fore-between (tokens before and between the two entities), between (only tokens between the two entities), and between-after (tokens between and after the two entities). The second kernel represents the local context using NLP basic features such as: lemma, part-of-speech, stem, capitalization, punctuation, and so on.

8.3 Experiments and evaluation

We choose seven different DBpedia properties to evaluate the system, covering different domain-range pairs. For each relation, we extract 50,000 pages for training, 1,000 for development and 1,000 for test (except for the `capital` property, for which not enough instances are available). All the experiments are conducted on a test set built following the same steps used for training.

The strategy used to calculate precision and recall is *One Answer per Slot* [LCC$^+$08]:

- if the system finds in the document the correct value for the desired relation, we ignore the remaining false negatives for the same relation;
- if the system does not find it (or the value is wrong), a false negative is considered;
- false positives are not affected by this method, therefore they are all counted when computing precision and recall.

Table 8.2 shows the results obtained on the seven considered relations, with and without the application of the strategies described in Section 8.2.2. The second column reports the number of occurrences in DBpedia for the relation. The property `capital` is not very frequent, but we choose to use it in our test to demonstrate that our approach is also feasible when we do not have a large number of examples. Unfortunately, in this case the three strategies are not applicable, because the number of examples (positive and negative) drops due to the required constraints (see Section 8.2.2).

The results show that the application of the three strategies increases the F_1 value. In some cases (`birthDate`, `deathDate`, `populationTotal`) the increase is negligible, because the corresponding relations are not affected by the issue described in [RYM10].

9 Airpedia, an automatically built LOD resource

As the algorithms described in the previous Chapters have been used to expand the coverage of DBpedia over Wikipedia, we decided to make the resulting resource freely available for download.

The project has been called Airpedia, and the datasets are available on the website (`http://www.airpedia.org/`) as downloadable packages, under both the Creative Commons Attribution-ShareAlike License[1] and the GNU Free Documentation License.[2]

The resource consists of the following parts:

- Mapping generation for classes in 25 languages, 14 of them not yet included in DBpedia.
- Mapping generation for properties in 14 languages non included in DBpedia.
- Classification of Wikipedia pages with respect to the DBpedia ontology (version 1) in 6 languages, also available as SPARQL endpoint.
- Classification of Wikipedia pages with respect to the DBpedia ontology (version 2) in 31 languages.

In Sections 9.1 and 9.2 we will describe in detail how each dataset is built and what it contains.

In the mapping generation cases (Section 9.1), even if the precision is not 100% and the process still needs human supervision, our approach can drastically reduce the time required, estimated in around 5 minutes per mapping per language if performed from scratch.[3]

[1] `http://creativecommons.org/licenses/by-sa/3.0/`

[2] `http://www.gnu.org/copyleft/fdl.html`

[3] This is an average time evaluated during the mapping of the Italian DBpedia, and discussed in a personal communication with one of the DBpedia annotators.

In addition, the framework that underlies the scripts used to build the resource has been found helpful in finding errors in the existing DBpedia mappings. Section 9.3 will focus on this topic.

9.1 Mapping generation

9.1.1 Classes (released April 2013)

This Section presents the resource obtained by automatically mapping Wikipedia infoboxes in 25 languages to the corresponding classes in the DBpedia ontology, as described in Chapter 5. This resource reproduces the mapping task achieved by the DBpedia community, therefore it is immediately ready for the extraction framework (it only needs format conversion, see Section 2.5.2).

For each language, we only mapped the infoboxes that appear more than 50 times in the corresponding Wikipedia and released 3 different mappings corresponding to 3 distinct L parameters, 0.1, 0.9, 0.5, corresponding to the maximum recall, maximum precision, and maximum F_1, respectively (see Section 5.1). These values are based on the evaluation performed on the five languages considered in Section 5.2.

Table 9.1 shows the number of mappings extracted for each language. The first 11 languages in the table have already mappings in DBpedia (the second column shows the number of templates already mapped), while the remaining 14 languages do not have mappings. The last three columns show the number of extracted templates by our method, respectively, with $L = 0.1$, $L = 0.9$ and $L = 0.5$. For example, we can quadruplicate the number of Catalan mappings with 100% precision.

The resource is available in tab separated values format.

9.1.2 Properties (released May 2013)

In this section, we present the resource obtained by mapping 45,978 Wikipedia infobox attributes to DBpedia properties in 14 different languages for which mappings do not yet exist, as described in Chapter 6. Again, the task reproduces the one achieved by the DBpedia community.

For each language, we only consider infoboxes that appear more than 10 times in the corresponding Wikipedia and release the mappings paired with the value of the function f, described in the Section 6.4. The system has been trained on the DBpedia datasets in 6 languages (English, Italian, French, German, Spanish and Portuguese).

Language	DBpedia	L = 0.1	L = 0.9	L = 0.5	Language	DBpedia	L = 0.1	L = 0.9	L = 0.5
Bulgarian	58	106	109	109	Estonian	-	51	52	51
Catalan	49	191	198	197	Finnish	-	122	127	126
Czech	66	131	137	135	Icelandic	-	18	18	18
Croatian	36	113	115	113	Lithuanian	-	112	113	113
Hungarian	108	176	185	184	Latvian	-	69	71	71
Indonesian	48	159	161	160	Norwegian	-	154	160	159
Dutch	99	362	372	371	Romanian	-	134	138	136
Polish	340	216	228	226	Slovak	-	101	102	102
Russian	30	340	354	352	Albanian	-	27	28	28
Slovenian	160	88	91	91	Serbian	-	164	166	165
Turkish	215	129	133	132	Swedish	-	175	182	182
Belarusian	-	56	59	59	Ukranian	-	236	247	242
Danish	-	109	112	111					

Table 9.1: Infoboxes estracted and available as a resource

Table 9.2 shows the number of mappings extracted for each language ($\lambda = 0.3$, see Section 6.4).

The resource is available in tab separated values format.

Language	Mappings	Language	Mappings
Belarusian	1,895	Norwegian	4,226
Danish	3,303	Romanian	4,563
Estonian	1,297	Slovak	2,407
Finnish	3,766	Albanian	1,144
Icelandic	646	Serbian	4,343
Lithuanian	3,733	Swedish	5,073
Latvian	2,085	Ukranian	5,760

Table 9.2: Mappings extracted and available as a resource

9.2 Wikipedia page classification

This Section describes the resource obtained by classifying the Wikipedia articles using the method described in Chapter 7. Each page in Wikipedia is classified through our system, that gives the most probable DBpedia class to which the page belongs. We apply the algorithm twice, and we release two different versions of the resource.

9.2.1 Version 1 (released December 2012)

This resource includes classification for Wikipedia articles in 6 languages (English, German, Italian, French, Spanish, and Portuguese). Pages already present in the DBpedia release are not considered.

The resource contains more than 1 million new entities (with respect to the original DBpedia 3.8 resource), available for download as a package using standard RDF specifications. A SPARQL endpoint where to make complex queries is available.[4]

We use the N-Quads format,[5] that add the "context" to the common N-Triples format. Usually, pure N-Triples encode standard subject-predicate-object tuples in each line, using the syntax:

```
<subject> <predicate> <object> .
```

Quadruples (N-Quads) are syntactically almost identical to N-Triples, but add a fourth element to them, typically denoting the context used for the triple generation. The typical syntax is:

```
<subject> <predicate> <object> <context> .
```

In our implementation, that fourth element corresponds to the values used for the extraction, resulting in different precision/recall tradeoffs.

An example of quadruple in the Airpedia dataset is:

```
<http://dbpedia.org/resource/Bob_Frankston>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://dbpedia.org/ontology/Person>
<http://airpedia.org/extraction/10-all> .
```

where the first three parts correspond to the typical DBpedia triple, while the context URI `http://airpedia.org/extraction/10-all` refers to the **K_{COMBO} Bottom-up** setup (see Section 7.2.6).

Globally, there are three extraction contexts available, related to the different setups described in Section 7.2.6:

- `10-all` refers to **K_{COMBO} Bottom-up**.
- `10-all-top` refers to **K_{COMBO} Top-down**.
- `10-tpl` refers to **K_T Bottom-up**.

An additional package is released with context URI `dbpedia-cl`, and it contains all the pages included in DBpedia in the six pivot languages, extending the coverage over other languages (see Section 4.2).

[4] http://www.airpedia.org/sparql/
[5] http://www.w3.org/TR/2013/NOTE-n-quads-20130409/

9.2.2 Version 2 (released June 2013)

This second version uses the algorithm applied to extract the previous version, with some adaptation and enhancements. The main differences between the two versions are described in the following subsections.

Languages

First of all, the number of languages involves has been expanded to 31, the original six, plus 25 new (Belarus, Bulgarian, Catalan, Czech, Danish, Estonian, Finnish, Croatian, Hungarian, Indonesian, Icelandic, Lithuanian, Latvian, Dutch, Norwegian, Polish, Romanian, Russian, Slovak, Slovenian, Albanian, Serbian, Swedish, Turkish, Ukrainian).

The training has been carried out on all the DBpedia editions available in version 3.8 (16 different languages).

Format

The new version of the resource is released in the N-Triples format. The setup of the classification has been defined in the predicate part, that now does not belong to DBpedia anymore, but is defined in the Airpedia namespace.

Its format is `http://airpedia.org/ontology/type_with_conf#N` where `N` can take the values 6 to 10, depending on the accuracy: the higher the number, the more precise the classification. Value from 6 to 9 refers to Wikipedia pages newly classified using our algorithms. The value 10 means that this article is already classified in DBpedia in some language, therefore the corresponding class has been assigned using the manually-generated mappings and the cross-language links.

An example of triple in the version 2 of the Airpedia dataset is:

```
<http://lt.dbpedia.org/resource/Hantelio_Åkas>
<http://airpedia.org/ontology/type_with_conf#9>
<http://dbpedia.org/ontology/CelestialBody> .
```

It means that the page in the Lithuanian Wikipedia `Hantelio Åkas` is classified as `CelestialBody` with accuracy 9 (that means around 90% of precision, see Section 7.2.6).

There is no SPARQL endpoint available for this version of Airpedia yet.

9.2.3 Integration with the Italian DBpedia

In June 2013, the Italian community of DBpedia releases the new version of the dataset, including triples from the Airpedia project. From the DBpedia website:[6]

> Enriched `rdf:type` information from Project Airpedia; that permitted resources with unmapped templates to show an estimated type, derived with a Machine Learning algorithm; those resources are marked with property `http://airpedia.org/ontology/is_estimated_type` (set to "true" or unavailable). Moreover we will have also confidence information triples like `http://airpedia.org/ontology/type_with_conf#9`, where `conf#` is in range from 6 to 10 (10 meaning that other international DBpedias have a similar page with manually mapped type).

This addition to the Italian DBpedia has been performed by merging classification from both versions of Airpedia. When the page has not any DBpedia classification, a triple with the original predicate <`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`> is added, and the property `is_estimated_type` is set to "true", so that the user can infer that the classification does not derive from the Italian DBpedia mappings, but it originates from the Airpedia dataset. Following these rules, 2,954,328 new type classifications were added to the Italian DBpedia. Figure 9.1 shows an example of such page.

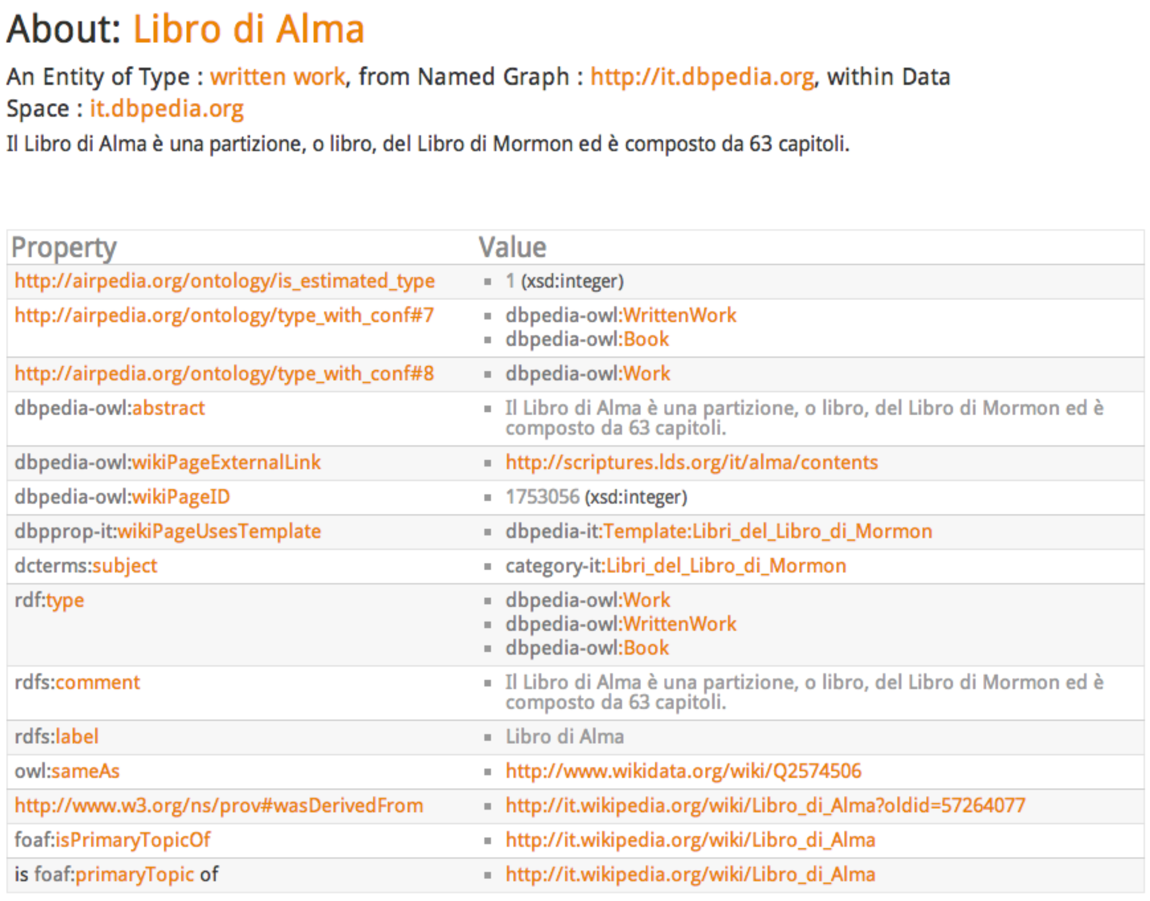
Looking at the property table, we can see that:

- `http://airpedia.org/ontology/type_with_conf#7` gives the two classifications `WrittenWork` and `Book` with accuracy 7 (around 80% of precision).
- `http://airpedia.org/ontology/type_with_conf#8` means that the classification `Work` has an accuracy of 8 (around 85% of precision).
- Finally, <`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`> (published in its abbreviated form `rdf:type`) is given as `Work`, `WrittenWork` and `Book`. Since `http://airpedia.org/ontology/is_estimated_type` is set to 1, it means that the given classification is taken from Airpedia.

9.3 DBpedia error reporting

The basic idea of our approaches is the use of DBpedia resource (existing mappings and dataset) as training data. For this reason, we have to be sure that the mappings are correct. While working

[6] `http://it.dbpedia.org/2013/06/nuova-release-dbpedia-3-2-airpedia-wikidata/?lang=en`



About: Libro di Alma

An Entity of Type : written work, from Named Graph : http://it.dbpedia.org, within Data Space : it.dbpedia.org

Il Libro di Alma è una partizione, o libro, del Libro di Mormon ed è composto da 63 capitoli.

Property	Value
http://airpedia.org/ontology/is_estimated_type	▪ 1 (xsd:integer)
http://airpedia.org/ontology/type_with_conf#7	▪ dbpedia-owl:WrittenWork ▪ dbpedia-owl:Book
http://airpedia.org/ontology/type_with_conf#8	▪ dbpedia-owl:Work
dbpedia-owl:abstract	▪ Il Libro di Alma è una partizione, o libro, del Libro di Mormon ed è composto da 63 capitoli.
dbpedia-owl:wikiPageExternalLink	▪ http://scriptures.lds.org/it/alma/contents
dbpedia-owl:wikiPageID	▪ 1753056 (xsd:integer)
dbpprop-it:wikiPageUsesTemplate	▪ dbpedia-it:Template:Libri_del_Libro_di_Mormon
dcterms:subject	▪ category-it:Libri_del_Libro_di_Mormon
rdf:type	▪ dbpedia-owl:Work ▪ dbpedia-owl:WrittenWork ▪ dbpedia-owl:Book
rdfs:comment	▪ Il Libro di Alma è una partizione, o libro, del Libro di Mormon ed è composto da 63 capitoli.
rdfs:label	▪ Libro di Alma
owl:sameAs	▪ http://www.wikidata.org/wiki/Q2574506
http://www.w3.org/ns/prov#wasDerivedFrom	▪ http://it.wikipedia.org/wiki/Libro_di_Alma?oldid=57264077
foaf:isPrimaryTopicOf	▪ http://it.wikipedia.org/wiki/Libro_di_Alma
is foaf:primaryTopic of	▪ http://it.wikipedia.org/wiki/Libro_di_Alma

Figure 9.1: An Italian DBpedia page with the Airpedia classification

on the mapping generation, we implemented a cross-language validation to discover wrong mappings. The task is trivial under the hypothesis that different Wikipedia editions shares the same infoboxes. Unfortunately, the real situation is chaotic, since the infobox distribution in the various editions is much different (see Section 1.3.2).

The basic idea of this simple tool is that the given classifications for a particular entity in different editions of DBpedia should be the same or, at most, on the same path. For example, `Barack Obama` is classified as `Person` in the German DBpedia and as `Politician` in the Spanish one. This is correct: although the classifications are different, the class `Person` is parent of `Politician`. In other cases, the path is not the same, therefore we guess that the mapping that generates the classification is not correct. For example, the page `Avena` is classified as `FloweringPlant` in the Italian and Portuguese versions of DBpedia, `Plant` in the English one, and `Tax` in the Spanish one. Clearly, the last classification is wrong (`Avena` is a plant, not a tax). The mistake is due to the name of the involved infobox, named `Ficha de taxón`, that could be misunderstood and confused with “tax”. In this example, our system collects the wrong classifications whenever a situation like this is encountered. If a certain class in a certain language is classified wrongly most of the times, the whole corresponding mapping is considered wrong, and therefore all the pages originated from it are discarded from the training.

Using this heuristics, we found out some obvious errors, that has been sent to the DBpedia community[7] and have been corrected before the release of DBpedia 3.9.

Note that not all the errors are the result of a wrong mapping. For example, some pages describing atolls in English DBpedia were classified as `Island`, while the same entities were correctly assigned to `Atoll` in the German version. A human can notice that indeed the classification is correct, because in English Wikipedia an infobox for atolls does not exist, and they use always `Infobox Islands`. In this case, the problem was in the ontology: the class `Atoll` was not a a child of class `Island`, as it should be. Again, we reported that issue to the DBpedia community, and in the new version of the ontology the two classes have been correctly set.

[7] `https://groups.google.com/forum/#!topic/thosch/6s44rcDu4cM`

10
Case study: QAKiS

A Question Answering system based on Linked Open Data

This chapter describes the results of the work started in Fondazione Bruno Kessler (Trento) in 2011 during an internship program [MWB+11] named WikiFramework. Subsequently, the author of this thesis revised and extended the preliminary work (see Section 10.1) and this has become the core engine of QAKiS, a question answering system developed jointly with Elena Cabrio, Julien Cojan and Fabien Gandon from INRIA (Sophia Antipolis). QAKiS is an ongoing work, to be released as open source software in the next months. The system represents the starting point of all the studies on Semantic Web, Natural Language Processing and Linked Data described in this thesis, and the author still continues to collaborate with the QAKiS project.

NLP and Semantic Web research fields are becoming more and more interconnected, even if they address, in a way, opposite perspectives. NLP tools focus on unstructured information (e.g. text documents), while Semantic Web tools typically deal with more structured information on a more granular level. Many important problems span the two worlds of structured and unstructured information where the combination of NLP and Semantic Web tools would be highly complementary, for instance in the use of NLP tools to automatically tag web pages with RDF descriptions, or the possibility to query the SW to retrieve object related services using current search engines [Din04].

A very important case of such web sites offering strongly tied texts and data is the pair Wikipedia-DBpedia. Collaboratively constructed resources, such as Wikipedia, have grown into central knowledge sources providing a vast amount of updated information accessible on the Web essentially as pages for human consumption. From such corpora, structured information has been extracted and stored into knowledge bases (e.g. DBpedia), that cover a wide range of different domains and connect entities across them. Exploiting these interlinked structured and unstructured data sources in parallel offers a great potential for both NLP and Semantic Web applications. Many researchers have indeed used this kind of resources on the web as a basis for training statistical algorithms, and we should expect that much more brilliant results can be obtained if the learning base can rely on the fine grained grid proposed by the SW [Din04]. Moreover, NLP applications such as information extraction have in fact advanced to the point where they can ease the acquisition bottleneck, by creating semantic annotations from text. Such techniques are therefore able to provide the requisite RDF data for the SW from existing unstructured text resources in the web. At the same time, NLP systems can also consume semantic web data and schemas, providing for instance natural language query systems that take advantage of semantic web meta-data to provide the answers to a user question.

In line with the direction of filling the gap between NLP and Semantic Web, in this chapter we address the problem of enhancing interactions between non-expert users and data available on the Web. This research challenge can be broken down into the following sub-questions: *i)* how to automatically extract structured text from unstructured documents to populate RDF triple stores? and *ii)* in a Question Answering setting, how to map natural language expressions (e.g. user queries) to concepts and relations in a structured knowledge base?

Given the complexity of the research question, our work narrows its scope focusing on the pair Wikipedia-DBpedia as a case study of parallel sources of structured and unstructured data. In particular, we present ongoing work on *i)* WikiFramework, i.e. a methodology to collect relational patterns in several languages, for the relations defined in the DBpedia ontology, and *ii)* QAKiS, a system for open domain Question Answering over linked data. A crucial issue in Question Answering over linked data is the interpretation of the question in order to convert it into a corresponding query in a formal language (e.g. SPARQL). Most of current approaches (e.g. PowerAqua, Freya [LUSM11]) base this conversion on some form of flexible matching between words of the question and names of concepts and relations of a triple store. One of the limitations of such approach, however, is that a word-based match may fail to detect the relevant context around a word, without which the match might be wrong. The generic approach behind QAKiS addresses the task of QA over structured knowledge bases (i.e. DBpedia) where the relevant information is expressed also in unstructured form (i.e. Wikipedia pages). Its major novelty is that it implements a relation-based match, where fragments of the question are matched to relational textual patterns automatically collected from Wikipedia (i.e. WikiFramework repository). The underlying intuition is that a relation-based matching would provide more precision with respect to the matching on single tokens, as done by current QA systems on linked data.

10.1 WikiFramework: collecting relational patterns

Relational patterns capture different ways to express a certain relation in a given language. For instance, the relation `birthDate(Person, Date)` can be expressed in English by the following relational patterns: [`Person` was born in `Date`], [`Person`, (`Date`)], [`Person`, whose date of birth is `Date`].

The goal of WikiFramework [MWB+11] we present in this section is to establish a robust methodology to collect relational patterns in several languages, for the relations defined in DBpedia ontology (similarly to [GN11], [WW10]) that were presented in Section 3.5.

As already stated in Section 8.1, we assume that there is a high probability that the structured information present in the *infobox* is also expressed using natural language sentences in the same Wikipedia page. Therefore, we collect all the triples from all the DBpedia ontology relations, and for each relation we apply the following procedure:

(i) the subject of each triple is extracted (e.g. the instance `Golden Gate Bridge`);

(ii) each DBpedia relation is automatically matched with the Wikipedia pages that represents the subject, and in which such relation is reported in the *infobox*. For instance, given the relation `crosses`, the Wikipedia page about the Golden Gate Bridge is selected (Figure 10.1);

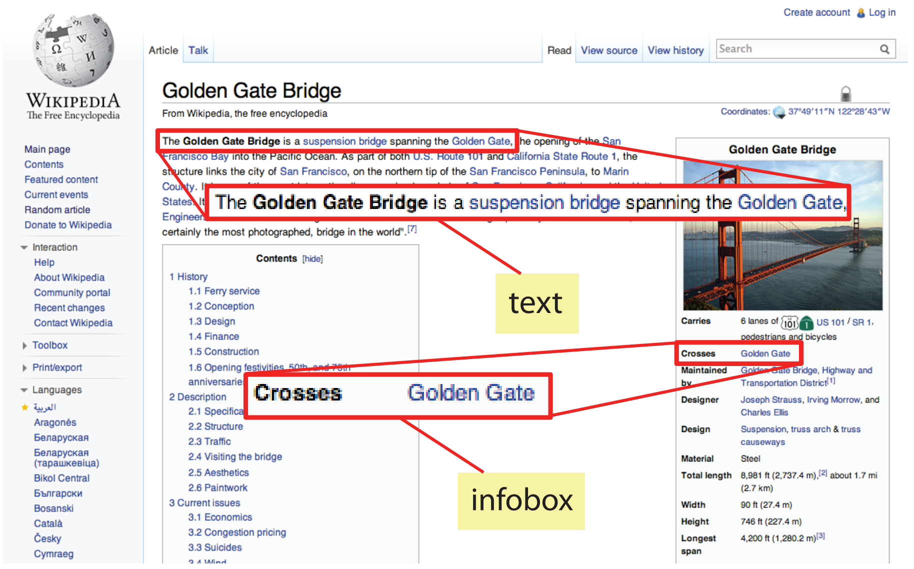
(iii) all the sentences in the selected Wikipedia pages where both the strings of subject and object of the relation match are collected. For instance, in the page about the Golden Gate Bridge (Figure 10.1), the sentence “The Golden Gate Bridge is a suspension bridge spanning the Golden Gate” is detected, since both entities match (i.e. subject: Golden Gate Bridge, object: Golden Gate);

(iv) such matching sentences are extracted and the subject and object are substituted with the corresponding DBpedia ontology classes (i.e. for the relation `crosses`, `Bridge` is the domain and `River` is the range)

(v) the pattern [The `Bridge` is a suspension bridge spanning the `River`] is obtained and stored in a pattern repository.

To increase the recall of the pattern extraction algorithm outlined above, we apply the matching strategies already described in Section 8.2.1.

Once the patterns for all relations are collected, we cluster them according to the lemmas that are present between the subject and the object. Then, we sort such patterns according to the frequency they appear in Wikipedia pages, and we wipe out: *i)* the ones whose frequency is



Create account Log in

Article Talk Read View source View history Search

Golden Gate Bridge

From Wikipedia, the free encyclopedia

Coordinates: 37°49′11″N 122°28′43″W

The **Golden Gate Bridge** is a suspension bridge spanning the Golden Gate, the opening of the San Francisco Bay into the Pacific Ocean. As part of both U.S. Route 101 and California State Route 1, the structure links the city of San Francisco, on the northern tip of the San Francisco Peninsula, to Marin County. It … States. It … Engineers … certainly the most photographed, bridge in the world".[7]

The **Golden Gate Bridge** is a suspension bridge spanning the Golden Gate,

Contents [hide]
1 History
1.1 Ferry service
1.2 Conception
1.3 Design
1.4 Finance
1.5 Construction
1.6 Opening festivities, 50th, and 75th anniversaries
2 Description
2.1 Specifica…
2.2 Structure
2.3 Traffic
2.4 Visiting the bridge
2.5 Aesthetics
2.6 Paintwork
3 Current issues
3.1 Economics
3.2 Congestion pricing
3.3 Suicides
3.4 Wind

Golden Gate Bridge

Carries	6 lanes of US 101 / SR 1, pedestrians and bicycles
Crosses	Golden Gate
Maintained by	Golden Gate Bridge, Highway and Transportation District[1]
Designer	Joseph Strauss, Irving Morrow, and Charles Ellis
Design	Suspension, truss arch & truss causeways
Material	Steel
Total length	8,981 ft (2,737.4 m),[2] about 1.7 mi (2.7 km)
Width	90 ft (27.4 m)
Height	746 ft (227.4 m)
Longest span	4,200 ft (1,280.2 m)[3]

Crosses Golden Gate

Figure 10.1: An example of Wikipedia page with infobox

Relation	Patterns	Relation	Patterns

`spouse`	`Person` is the wife of `Person`
	`Person` married `Person`
	`Person`, the husband of `Person`
		`birthDate`	`Person` was born on `Date`
			`Person` (`Date`
			`Person` (born `Date`

`crosses`	`Bridge` spans the `River`
	`Bridge` is a bridge over the `River`
	`Bridge` crossing `River`
		`location`	`Thing`, located in `PopulatedPlace`
			`Thing` founded in `PopulatedPlace`
			`Thing` corporate headquarters in `PopulatedPlace`

`author`	`Person` wrote `Work`
	`Work` is a novel by `Person`
	`Work` was written by `Person`
		`numberOf Students`	`University` student population is `Number`
			`University` school enrolls `Number`
			`Number` students attend `University`

Table 10.1: Examples of relational patterns

less than 2, and *ii)* the ones that contain only non discriminative words (e.g. punctuation marks, prepositions, articles, etc. as in the pattern [`Person` (`Date`)]). Table 10.1 reports a sample of the set of patterns collected for a few DBpedia relations.

This section has shown how the coupling of documents and data in a web source can be mined to collect natural language relational patterns corresponding to data schema relational patterns. In the next section, we will show how these patterns can be used to enhance natural language interactions with the data collection, i.e. how WikiFramework pattern repository is integrated and exploited by the QAKiS system for answer retrieval.

10.2 QAKiS: a system for data answer retrieval from natural language questions

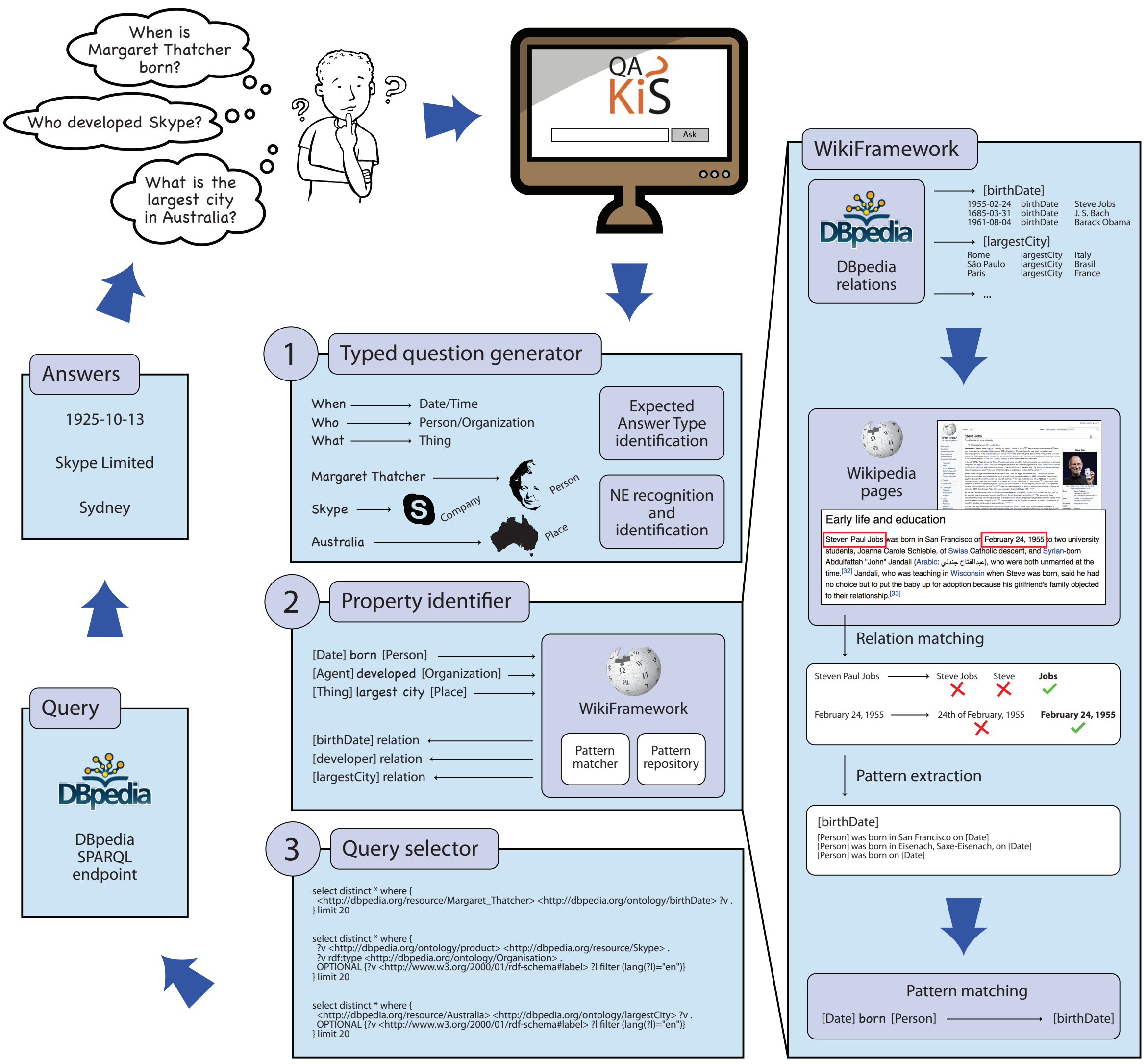
QAKiS (Question Answering wiKiframework-based System) is a question answering system that allows end users to submit a query to an RDF triple store in English and obtain the answer in the same language, hiding the complexity of the non intuitive formal query languages involved in the resolution process. At the same time, the expressiveness of these standards is exploited to scale to the huge amounts of available semantic data. In its current implementation, QAKiS addresses the task of QA over structured Knowledge Bases (KBs) (e.g. DBpedia) where the relevant information is expressed also in unstructured form (e.g. Wikipedia pages). It implements a relation-based match for question interpretation, to convert the user question into SPARQL. The underlying intuition is that a relation-based matching would provide more precision with respect to matching on single tokens, as done by current QA systems.

In this section we present the QAKiS architecture, and a step by step description of the system workflow. QAKiS is composed of two main modules (Figure 10.2): *i)* the *typed question generator* takes the user question as input, generates the typed questions to be matched with the patterns, and then generates the SPARQL queries from the retrieved patterns; *ii)* the *property identifier* takes as input a typed question, and retrieves the patterns (among those stored in the pattern repository) that match such question with the highest similarity.

The current version of QAKiS targets questions containing a Named Entity (NE) that is related to the answer through one property of the ontology, as *Which river does the Brooklyn Bridge cross?*. According to our WikiFramework-based approach, each question matches a single pattern (i.e. one relation). Before running the pattern matcher component, we replace *i)* the NE present in the question by its types, and *ii)* the question keywords by the Expected Answer Type (EAT), obtaining what we call a *typed question* (e.g. `[River]` does the `[Bridge]` cross?). The answer can then be retrieved with a SPARQL query over a single triple.

10.2.1 NE identification and Expected Answer Type (EAT)

Before running the *pattern matcher* component, we identify the target of the question with a NER (Named Entity Recognition) tool. We apply the Stanford Core NLP NE Recognizer together with a set of strategies based on the comparison with the labels of the instances in the DBpedia ontology. We plan to test the use of other NER tools in the future. At the same time, simple heuristics are applied to infer the EAT from the question keyword. For example, if the question starts with “When”, the EAT is `[Date]` or `[Time]`, with “Who”, the EAT is `[Person]` or `[Organisation]` and so on.


When is Margaret Thatcher born?
Who developed Skype?
What is the largest city in Australia?
QA KiS
Ask
1 Typed question generator
When → Date/Time
Who → Person/Organization
What → Thing
Expected Answer Type identification
Margaret Thatcher → Person
Skype → Company
Australia → Place
NE recognition and identification
2 Property identifier
[Date] born [Person] →
[Agent] developed [Organization] →
[Thing] largest city [Place] →
[birthDate] relation ←
[developer] relation ←
[largestCity] relation ←
WikiFramework
Pattern matcher
Pattern repository
3 Query selector
Query
DBpedia SPARQL endpoint
Answers
1925-10-13
Skype Limited
Sydney
WikiFramework
DBpedia relations
→ [birthDate]
1955-02-24 birthDate Steve Jobs
1685-03-31 birthDate J. S. Bach
1961-08-04 birthDate Barack Obama
→ [largestCity]
Rome largestCity Italy
São Paulo largestCity Brasil
Paris largestCity France
→ ...
Wikipedia pages
Relation matching
Steven Paul Jobs → Steve Jobs ✗ Steve ✗ **Jobs** ✓
February 24, 1955 → 24th of February, 1955 ✗ **February 24, 1955** ✓
Pattern extraction
[birthDate]
[Person] was born in San Francisco on [Date]
[Person] was born in Eisenach, Saxe-Eisenach, on [Date]
[Person] was born on [Date]
Pattern matching
[Date] born [Person] → [birthDate]

```
select distinct * where {
  <http://dbpedia.org/resource/Margaret_Thatcher> <http://dbpedia.org/ontology/birthDate> ?v .
} limit 20

select distinct * where {
  ?v <http://dbpedia.org/ontology/product> <http://dbpedia.org/resource/Skype> .
  ?v rdf:type <http://dbpedia.org/ontology/Organisation> .
  OPTIONAL {?v <http://www.w3.org/2000/01/rdf-schema#label> ?l filter (lang(?l)="en")}
} limit 20

select distinct * where {
  <http://dbpedia.org/resource/Australia> <http://dbpedia.org/ontology/largestCity> ?v .
  OPTIONAL {?v <http://www.w3.org/2000/01/rdf-schema#label> ?l filter (lang(?l)="en")}
} limit 20
```

Early life and education

Steven Paul Jobs was born in San Francisco on February 24, 1955 to two university students, Joanne Carole Schieble, of Swiss Catholic descent, and Syrian-born Abdulfattah "John" Jandali (Arabic: عبدالفتاح جندلي), who were both unmarried at the time.[32] Jandali, who was teaching in Wisconsin when Steve was born, said he had no choice but to put the baby up for adoption because his girlfriend's family objected to their relationship.[33]

Figure 10.2: QAKiS system architecture

10.2.2 Typed questions generation

We generate a *typed question* by replacing the question keywords (e.g. who, where) and the NE by the types and supertypes. Given the question "Who is the husband of Amanda Palmer?" 9 typed questions are generated, since *i)* both `[Person]` or `[Organisation]` (subclasses of `[owl:Thing]`) are considered as EAT, and *ii)* `[MusicalArtist]`, `[Artist]` and `[owl:Thing]` are the types of the NE *Amanda Palmer*.

10.2.3 WikiFramework pattern matching

The typed questions are lemmatized, tokenized, and stopwords are removed. The same procedure is applied to the patterns stored in WikiFramework repository, and three sets of keywords for each relation are created, respectively for most frequent tokens, lemmas and stems. Each set contains 20 words, sorted by the frequency of presence in the collected patterns. To further improve recall, we append to the sets of keywords the tokens, lemmas and stems extracted from the CamelCase[1] name of the relation (e.g. the tokens "birth" and "date" are added to the keywords of the relation `birthDate`). A Word Overlap algorithm is then applied to match the typed questions with such patterns for each relation. A similarity score is provided for each match: the highest represents the most likely relation.

10.2.4 Query selector

A set of patterns (max. 5) is retrieved by the pattern matcher component for each typed question, and sorted by decreasing matching score. For each of them, one or two SPARQL queries are generated, either *i)* `select ?s where`{`?s` <property> <NE>}, *ii)* `select ?s where`{<NE> <property> `?s`} or *iii)* both, according to the compatibility between their types and the property domain and range. Such queries are then sent to the SPARQL endpoint for answer retrieval. If the query produces no results, we try with the next pattern, until a successful query is found or no more patterns are retrieved.

In the previous sections we have described step by step the complex workflow that allows us to build a question answering system on top of a collectively maintained encyclopedia. In the next section, we experiment the combination of all those processes, and we evaluate the quality of the results on a standard data set.

[1]CamelCase refers to the practice of writing compound words or phrases without using trailing spaces and so that each word or abbreviation begins with a capital letter.

10.3 Experimental evaluation

Table 1 reports QAKiS's results on the Question Answring over Linked Data (QALD-2) data sets[2] (DBpedia track). As introduced before, the current version of QAKiS targets only questions containing a Named Entity that is related to the answer through one property of the ontology. However, QAKiS performance is in line with the results obtained by the other participating systems (avg. precision, recall and F-measure on the test set are respectively 0.40, 0.42 and 0.41).

	Precision	*Recall*	*F-measure*	*# answered*	*# right answ.*	*# partially right*
train	0.476	0.479	0.477	40/100	17/40	4/40
test	0.39	0.37	0.38	35/100	11/35	4/35

Table 10.2: QAKiS performances on DBpedia datasets (participation to QALD-2 [CAC+12])

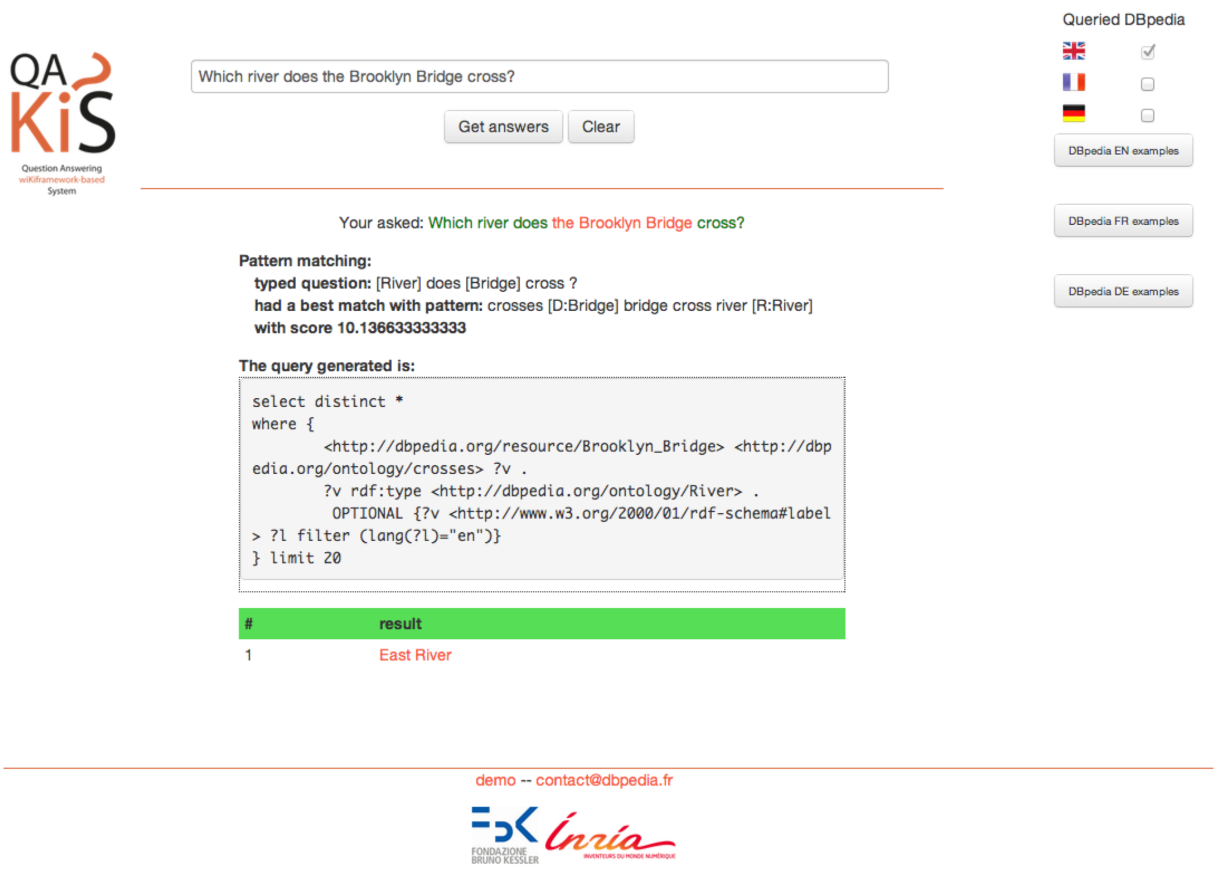
Most of QAKiS' mistakes concern wrong relation assignment (i.e. wrong pattern matching). We plan to replace the Word Overlap algorithm with approaches considering the syntactic structure of the question. Another issue concerns question ambiguity, i.e. the same grammar form can in fact refer to different relations in the DBpedia ontology (e.g. *Who is the owner of Universal Studios?* and *Who owns Aldi?* rely respectively on the relations `owner` and `keyPerson` for answer retrieval). We plan to cluster relations with several patterns in common, to allow QAKiS to search among all the relations in the cluster.

The partially correct answers concern questions involving more than one relation: the current version of the algorithm detects indeed only one of them. We plan to target questions as *Give me all people that were born in Vienna and died in Berlin* in a short time, since the two relations are easily separable and can be joint using a JOIN function in the SPARQL query. On the contrary, we need more complex strategies to answer questions with nested relations (e.g. *Who is the daughter of Bill Clinton married to?*), for which at the moment we answer with the name of Bill Clinton's wife (we match only the relation `spouse`). We plan short term solutions also for boolean questions, as *Is the wife of president Obama called Michelle?* (id=7, training set): we correctly match the relation `spouse` but we provide as answer *Michelle_Obama*, instead of the boolean *true*.

10.4 Demo

A demo of QAKiS based on Wikipedia for patterns extraction and on DBpedia as RDF data set to be queried using a natural language interface is available online[3], and shown in Figure 10.3. The

[2] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`
[3] `http://dbpedia.inria.fr/qakis/`



Which river does the Brooklyn Bridge cross?

Your asked: Which river does the Brooklyn Bridge cross?

Pattern matching:
typed question: [River] does [Bridge] cross ?
had a best match with pattern: crosses [D:Bridge] bridge cross river [R:River]
with score 10.136633333333

The query generated is:

```
select distinct *
where {
        <http://dbpedia.org/resource/Brooklyn_Bridge> <http://dbp
edia.org/ontology/crosses> ?v .
        ?v rdf:type <http://dbpedia.org/ontology/River> .
         OPTIONAL {?v <http://www.w3.org/2000/01/rdf-schema#label
> ?l filter (lang(?l)="en")}
} limit 20
```

#	result
1	East River

Figure 10.3: QAKiS demo interface

user can either write a question or select among a list of examples, and click on *Get Answers!*. QAKiS outputs: *(i)* the user question (the recognized Named Entity (NE) is linked to its DBpedia page), *(ii)* the generated typed question (see Section 10.2), *(iii)* the pattern matched, *(iv)* the SPARQL query sent to the DBpedia SPARQL endpoint, and *(v)* the answer (below the green rectangle *results*).

11 Conclusions and future work

In this thesis, we have studied the problem of automatically expanding DBpedia using different approaches to reach two different goals: automating the mapping process, and extracting information from a Wikipedia page when the infobox is missing or incomplete.

A key concept underlying our approaches is the multi-linguality of Wikipedia, as it allows us to automatically obtain information on entities in different Wikipedias to collect training data (classes and properties) where infoboxes were missing, and therefore to bootstrap the coverage of the DBpedia resource.

In addition, this multi-linguality approach has been used to find wrong mappings in DBpedia and to enhance its precision in future releases.

In the next Sections, we will describe the achievements of our system in terms of mapping generation, page classification and question answering. For each topic, we also give possible forthcoming research directions for the future work.

11.1 Mapping generation

In Chapter 5, we have proposed a three-step approach that automatically maps templates in Wikipedia into DBpedia classes. After expanding the population of DBpedia using cross-language links, we extracted the list of infoboxes from Wikipedia, and finally defined an algorithm that maps these infoboxes to the most probable class in the DBpedia ontology, starting from the existing mappings on six pivot languages. The experiments have been evaluated on the already present mappings on five languages, showing high precision and recall. Tradeoff between precision and recall can be varied by means of a single parameter.

In Chapter 6, we have extended this approach to properties, devising an instance-based approach that uses existing DBpedia editions as training data. We evaluated the system on

Italian data, using 100 manually annotated infobox attributes, demonstrating that our results are comparable with the current mappings in term of precision (87% versus 88% for the human annotation). In average, the results lead to a significant improvement in term of recall (70%) and speed (a single mapping may need up to 5 minutes by a human), maintaining an acceptable precision (80%). The system has been used to map 45,978 infobox attributes in 14 different languages for which mappings were not yet available; the resource is made available in an open format.

There remains room for further improvements. For example, the similarity function can be refined with a smarter normalization and a better recognition of typed entities (like temporal expressions, units, and common abbreviations).

We will also evaluate to what extent, in terms of precision and recall, DBpedia class mappings can be generated from the property mappings automatically found by our system.

Finally, we will adapt the proposed approach to detect errors in the DBpedia mappings (during our tests we encountered a relevant number of wrong mappings in DBpedia), or to maintain the mappings up-to-date whenever the corresponding Wikipedia templates are updated by the Wikipedia editors.

11.2 Wikipedia page classification

In Chapter 7, we have proposed a machine learning approach that automatically extends the classification of Wikipedia pages with respect to the DBpedia ontology. After extending the population of DBpedia using cross-language links, we used this broader classification as training data to classify the remaining pages using a kernel-based supervised method. The experiments have been evaluated on a manually annotated test set containing 400 Wikipedia pages, resulting in high precision and recall, with different tradeoffs of these values depending on the parameters of the algorithm.

In addition, Chapter 8 has proposed a method to extract missing DBpedia properties from the article text of Wikipedia pages. The approach is based on the distant supervision paradigm, and makes use of supervised machine learning for the extraction.

The accuracy of our approach is comparable to other systems'. However, a precise comparison is hard to make, because they are applied on different resources and tasks. In [NM11], YAGO is used as resource to collect training sentences, while [SMT$^+$10] uses DBpedia and the distant supervision paradigm for the TAC-KBP slot filling task.

Due to the high variability and complexity of the task, much work is still to be done, and different issues should be addressed:

- Disambiguation tools and Wikipedia links could be used for sentence retrieving (see Section 8.2.1).
- In our experiments we have used jSRE as an out-off-the-shelf tool. We plan to investigate the use of kernels exploiting Wikipedia-related features, such as internal links.
- To increase the number of sentences that can be used for training, some approaches (e.g., [SMT+10]) use shallow coreference resolution using animate pronouns,. In real world applications, where the number of relations is high and the number of examples is not, a more sophisticated coreference resolution tool can help to obtain more training data.
- Distant supervision is a language-independent paradigm, although most of the resources and approaches concerns only English, and the multi-linguality of the approach has not been deeply investigated. DBpedia releases its resource in 16 languages, therefore it can be in principle used to apply distant supervision on languages for which suitable natural language tools are available (such as TextPro[1], OpenNLP[2] or Stanbol[3]). There were some preliminary works on applying distant supervision on Wikipedia and DBpedia in Portuguese [BFS+13] and Polish [ZP13].

11.3 Question Answering

In Chapter 10, we observe that exploiting the interlinked structured and unstructured data sources available on the web in parallel offers a great potential for both Natural Language Processing and Semantic Web applications, and we have investigated such intuition following two (converging) directions. On the one hand, we described WikiFramework, i.e. a robust methodology to automatically collect relational patterns in several languages, for the relations defined in DBpedia ontology. On the other hand, we presented the preliminary work on QAKiS, a QA system over linked data, that takes advantage of the relational patterns resulting from the application of the WikiFramework approach.

Several research lines can be considered for future work. With respect to WikiFramework, we plan to improve the pattern extraction algorithm following [WW10], and to extend it to collect patterns for other languages, for instance French and Italian. As discussed, multilingual versions of the DBpedia data set are now being released as stable versions, and can therefore be used to mine multilingual information. The variability (both cultural and of knowledge) coming from

[1]http://textpro.fbk.eu/
[2]http://opennlp.apache.org/
[3]http://stanbol.apache.org/

the exploitation of such multilingual and different data sets can be used to enrich the current knowledge bases (e.g. with pieces of information present for a certain language and not for another). A first step in such direction has already been done, since the new version of QAKiS also considers French and German DBpedia datasets as a resource where to get the information used to answer natural language questions, still expressed in English [CCG13]. For example, the question “How tall is Margaret Simpson?” cannot be answered using DBpedia in English, as in the corresponding Wikipedia this information is missing. Nevertheless, the French Wikipedia contains it, therefore a system that can query more than one DBpedia chapter can enhance its coverage over some specific properties.

We also plan to extend our pattern repository recursively applying and matching the patterns we collected and the instances present in DBpedia on other corpora on the web, to acquire more variability (in particular, for relations expressed in a standard way in Wikipedia such as *BirthDate*). Moreover, other social and collaborative content (e.g. Wiktionary[4]) can be mined for constructing and extending structured lexical semantic resources (following e.g. [ZMG08]). We are currently considering to publish the WikiFramework relational patterns as RDF triples, organized according to an RDF vocabulary describing the structure of the patterns (similarly to [GN11]).

Concerning QAKiS, we are planning to: *(i)* investigate the applicability of the Textual Entailment approach (a framework for applied semantics, where linguistic objects are mapped by means of semantic inferences at a textual level [DDMR09]) to improve the question-pattern matching algorithm; *(ii)* improve the system coverage, addressing boolean and n-relation questions (considering also the syntactic structure of the question); *(iii)* explore the use of Spotlight[5] [MJGSB11] or OpenCalais[6] tools to recognize Named Entities in the question, and to annotate mentions of DBpedia resources in text. We are also currently exploring QAKiS applicability in real application scenarios.

[4]http://www.wiktionary.org/
[5]http://spotlight.dbpedia.org
[6]http://www.opencalais.com/

Bibliography

[ABK$^+$07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: a nucleus for a web of open data. In *Proceedings of the 6th international Semantic Web Conference and 2nd Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[AS12] Maik Anderka and Benno Stein. A breakdown of quality flaws in Wikipedia. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, WebQuality '12, pages 11–18, New York, NY, USA, 2012. ACM.

[ASW09] Eytan Adar, Michael Skinner, and Daniel S. Weld. Information arbitrage across multi-lingual Wikipedia. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 94–103, New York, NY, USA, 2009. ACM.

[BCCH13] Paul Buitelaar, Key-Sun Choi, Philipp Cimiano, and Eduard H. Hovy. The Multilingual Semantic Web (Dagstuhl Seminar 12362). *Dagstuhl Reports*, 2(9):15–94, 2013.

[BDI09] Gosse Bouma, Sergio Duarte, and Zahurul Islam. Cross-lingual alignment and completion of Wikipedia templates. In *Proceedings of the Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies*, CLIAWS3 '09, pages 21–29, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[BEP$^+$08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[BFS$^+$13] David Soares Batista, David Forte, Rui Silva, Bruno Martins, and Mario Silva. Extracção de Relações Semânticas de Textos em Português Explorando a DBpédia e a Wikipédia. *Linguamatica*, 5(1):41–57, Julho 2013.

[BL06] Tim Berners-Lee. Linked data. *W3C Design Issues*, 2006.

[BL09] Tim Berners-Lee. TED Talk: Tim Berners-Lee on the next Web. http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html, 2009.

[BLHL+01] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific American*, 284(5):28–37, 2001.

[BM05] Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *NIPS*, 2005.

[CAC+12] Elena Cabrio, Alessio Palmero Aprosio, Julien Cojan, Bernardo Magnini, Fabien Gandon, and Alberto Lavelli. QAKiS at QALD-2. In *Proceedings of the Interacting with Linked Data (ILD) Workshop at ESWC 2012*, Heraklion, Greece, 2012.

[CCA+12] Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. QAKiS: an open domain QA system based on relational patterns. In Birte Glimm and David Huynh, editors, *International Semantic Web Conference (Posters & Demos)*, volume 914 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[CCAG12] Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, and Fabien Gandon. Natural language interaction with the web of data by mining its textual side. *Intelligenza Artificiale*, 6(2):121–133, 2012.

[CCG13] Julien Cojan, Elena Cabrio, and Fabien Gandon. Filling the gaps among DBpedia multilingual chapters for question answering. In *Proceedings of the 5th Annual ACM Web Science Conference*, WebSci '13, pages 33–42, New York, NY, USA, 2013. ACM.

[CGGR03] Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. Word sequence kernels. *J. Mach. Learn. Res.*, 3:1059–1082, March 2003.

[CST10] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2010.

[DAC12] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Freya: an interactive way of querying linked data using natural language. In *Proceedings of the 8th international conference on The Semantic Web*, ESWC'11, pages 125–138, Berlin, Heidelberg, 2012. Springer-Verlag.

[DDL$^+$90] Scott C. Deerwester, Susan T. Dumais, Thoms K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[DDMR09] Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering (JNLE)*, 15(Special Issue 04):i–xvii, 2009.

[DFJ$^+$04] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 652–659, New York, NY, USA, 2004. ACM.

[DG07] Ludovic Denoyer and Patrick Gallinari. The Wikipedia XML corpus. In Norbert Fuhr, Mounia Lalmas, and Andrew Trotman, editors, *Comparative Evaluation of XML Information Retrieval Systems*, volume 4518 of *Lecture Notes in Computer Science*, pages 12–19. Springer Berlin Heidelberg, 2007.

[Din04] Luca Dini. NLP technologies and semantic web: Risks, opportunities and challenges. *Intelligenza Artificiale*, 1(1):67–71, 2004.

[dMS$^+$08] Mathieu d'Aquin, Enrico Motta, Marta Sabou, Sofia Angeletou, Laurian Gridinoc, Vanessa Lopez, and Davide Guidi. Toward a new generation of semantic web applications. *IEEE Intelligent Systems*, 23(3):20–28, May 2008.

[EVS12] Basil Ell, Denny Vrandecic, and Elena Simperl. Spartiqulation: Verbalizing sparql queries. In *Proceedings of the Interacting with Linked Data (ILD) Workshop at ESWC 2012*, Heraklion, Greece, 2012.

[Fam11] Andrea Fama. *Open Data – Data Journalism*. Simplicissimus Book Farm, 2011.

[Fel98] C. Fellbaum. Wordnet: An electronic lexical database. In *Language, Speech and Communication*, MIT Press, 1998.

[FH02] Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

[Fle71] Joseph L. Fleiss. Measuring Nominal Scale Agreement Among Many Raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[Gan13] Aldo Gangemi. A comparison of knowledge extraction tools for the semantic web. In Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *The Semantic Web: Semantics and Big Data*, volume 7882 of *Lecture Notes in Computer Science*, pages 351–366. Springer Berlin Heidelberg, 2013.

[GBH09] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.

[GG07] Claudio Giuliano and Alfio Gliozzo. Instance based lexical entailment for ontology population. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 248–256, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[GGS05] Alfio Massimiliano Gliozzo, Claudio Giuliano, and Carlo Strapparava. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 403–410, Ann Arbor, Michigan, June 2005.

[Giu09] Claudio Giuliano. Fine-grained classification of named entities exploiting latent semantic kernels. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 201–209, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[GLR07] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Relation extraction and the influence of automatic named-entity recognition. *ACM Transactions on Speech and Language Processing*, 5(1):2:1–2:26, December 2007.

[GN11] Daniel Gerber and Axel-Cyrille Ngonga Ngomo. Bootstrapping the linked data web. In *Proceedings of the 1st Workshop on Web Scale Knowledge Extraction ISWC 2011*, Bonn, Germany, 2011.

[GNP$^+$12] Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. Automatic typing of DBpedia entities. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7649 of *Lecture Notes in Computer Science*, pages 65–81. Springer, 2012.

[GPnCR12] Guillermo Garrido, Anselmo Peñas, Bernardo Cabaleiro, and Álvaro Rodrigo. Temporally anchored relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 107–116, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[GS05] Alfio Gliozzo and Carlo Strapparava. Domain kernels for text categorization. In *Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 56–63, Ann Arbor, Michigan, June 2005.

[HI11] Bernhard Haslhofer and Antoine Isaac. data.europeana.eu: The europeana linked open data pilot. *International Conference on Dublin Core and Metadata Applications*, 0, 2011.

[HSBW12] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 2012.

[HZW10] Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. Learning 5000 relational extractors. In *ACL*, pages 286–295, 2010.

[KBA$^+$12] Dimitris Kontokostas, Charalampos Bratsas, Sören Auer, Sebastian Hellmann, Ioannis Antoniou, and George Metakides. Internationalization of Linked Data: The case of the Greek DBpedia edition. *Web Semantics: Science, Services and Agents on the World Wide Web*, 15(0):51 – 61, 2012.

[KLUX12] Sebastian Krause, Hong Li, Hans Uszkoreit, and Feiyu Xu. Large-scale learning of relation-extraction rules with distant supervision from the web. In Philippe Cudr-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jrme Euzenat, Manfred Hauswirth, JosianeXavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web ISWC 2012*, volume 7649 of *Lecture Notes in Computer Science*, pages 263–278. Springer Berlin Heidelberg, 2012.

[LCC$^+$08] Alberto Lavelli, MaryElaine Califf, Fabio Ciravegna, Dayne Freitag, Claudio Giuliano, Nicholas Kushmerick, Lorenza Romano, and Neil Ireson. Evaluation of machine learning-based information extraction algorithms: criticisms and recommendations. *Language Resources and Evaluation*, 42(4):361–393, 2008.

[LGMN12] Jens Lehmann, Daniel Gerber, Mohamed Morsey, and Axel-Cyrille Ngonga Ngomo. DeFacto - Deep fact validation. In *International Semantic Web Conference (1)*, pages 312–327, 2012.

[LIJ^{+}14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal*, 2014.

[LR05] Xin Li and Dan Roth. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249, 2005.

[LStC02] Huma Lodhi, John Shawe-taylor, and Nello Cristianini. Text classification using string kernels. *Journal of Machine Learning Research*, 2:563–569, 2002.

[LUSM09] Vanessa Lopez, Victoria S. Uren, Marta Sabou, and Enrico Motta. Cross ontology query answering on the semantic web: an initial evaluation. In *K-CAP*, pages 17–24, 2009.

[LUSM11] Vanessa Lopez, Victoria S. Uren, Marta Sabou, and Enrico Motta. Is question answering fit for the semantic web?: A survey. *Semantic Web*, 2(2):125–155, 2011.

[MBSJ09] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[MJGSB11] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA, 2011. ACM.

[MR00] I. Dan Melamed and Philip Resnik. Tagger evaluation given hierarchical tag sets. *Computers and the Humanities*, pages 79–84, 2000.

[MWB^{+}11] Rahmad Mahendra, Lilian Wanzare, Raffaella Bernardi, Alberto Lavelli, and Bernardo Magnini. Acquiring relational patterns from Wikipedia: A case study. In *Proceedings of the 5th Language and Technology Conference*, Poznan, Poland, 2011.

[NCM08] Joel Nothman, James R. Curran, and Tara Murphy. Transforming Wikipedia into named entity training data. In *Proceedings of the Australasian Language Technology Workshop*, Hobart, Australia, 2008.

[NM11] Truc-Vien T Nguyen and Alessandro Moschitti. End-to-end relation extraction using distant supervision from external semantic repositories. In *ACL (Short Papers)*, pages 277–282, 2011.

[NMN$^+$11] Thanh Nguyen, Viviane Moreira, Huong Nguyen, Hoa Nguyen, and Juliana Freire. Multilingual schema matching for Wikipedia infoboxes. *Proc. VLDB Endow.*, 5(2):133–144, October 2011.

[Nor89] Eric W. Noreen. *Computer-Intensive Methods for Testing Hypotheses : An Introduction*. Wiley-Interscience, 1989.

[NS08] Vivi Nastase and Michael Strube. Decoding Wikipedia categories for knowledge acquisition. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pages 1219–1224. AAAI Press, 2008.

[PAGL13a] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Automatic expansion of DBpedia exploiting Wikipedia cross-language information. In *Proceedings of the 10th Extended Semantic Web Conference*, 2013.

[PAGL13b] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Automatic Mapping of Wikipedia Templates for Fast Deployment of Localised DBpedia Datasets. In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*, 2013.

[PAGL13c] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Extending the Coverage of DBpedia Properties using Distant Supervision over Wikipedia. In *Proceedings of the 1st Workshop on NLP & DBpedia (ISWC)*, 2013.

[PAGL13d] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Towards an Automatic Creation of Localized Versions of DBpedia. In *Proceedings of the 12th International Semantic Web Conference*, 2013.

[PB13] Heiko Paulheim and Christian Bizer. Type Inference on Noisy RDF Data. In *Proceedings of the 12th International Semantic Web Conference*, 2013.

[Poh12] A. Pohl. Classifying the Wikipedia Articles into the OpenCyc Taxonomy. In *Proceedings of the Web of Linked Entities Workshop in conjuction with the 11th International Semantic Web Conference*, 2012.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.

[RBWK13] Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. A survey of noise reduction methods for distant supervision. In *Automated Knowledge Base Construction 2013. Proceedings of the 3rd Workshop on Knowledge Extraction at CIKM 2013*, California, USA, 2013.

[RLN13] Daniel Rinser, Dustin Lange, and Felix Naumann. Cross-lingual entity matching and infobox alignment in Wikipedia. *Information Systems*, 38(6):887 – 907, 2013.

[RYM10] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 148–163, Berlin, Heidelberg, 2010. Springer-Verlag.

[S$^+$05] Oreste Signore et al. Representing knowledge in the semantic web. *Open Culture Conference (organised by the Italian office of W3C)*, pages 27–29, 2005.

[SC12] Philipp Sorg and Philipp Cimiano. Exploiting Wikipedia for cross-lingual and multilingual information retrieval. *Data Knowl. Eng.*, 74:26–45, 2012.

[SE05] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171, 2005.

[SHB$^+$12] Afroza Sultana, Quazi Mainul Hasan, Ashis Kumer Biswas, Soumyava Das, Habibur Rahman, Chris Ding, and Chengkai Li. Infobox suggestion for Wikipedia entities. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 2307–2310, New York, NY, USA, 2012. ACM.

[SHBL06] N. Shadbolt, W. Hall, and T. Berners-Lee. The semantic web revisited. *Intelligent Systems, IEEE*, 21(3):96–101, 2006.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[SMT$^+$10] Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. A simple distant supervision approach for the TAC-KBP slot filling task. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*, Gaithersburg, Maryland, USA, November 2010.

[SS02] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.

[STC04a] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[STC04b] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[STT02] Craig Saunders, Hauke Tschach, and John S. Taylor. Syllables and other String Kernel Extensions. In *Proc. 19th International Conference on Machine Learning (ICML'02)*, pages 530–537, 2002.

[TCC$^+$10] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. Sig.ma: Live views on the web of data. *J. Web Sem.*, 8(4):355–364, 2010.

[TDO07] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: weaving the open linked data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 552–565, Berlin, Heidelberg, 2007. Springer-Verlag.

[UBL$^+$12] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 639–648, New York, NY, USA, 2012. ACM.

[UC11] Christina Unger and Philipp Cimiano. Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In *NLDB*, pages 153–160, 2011.

[Vap99] Vladimir Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[Vra12] Denny Vrandečić. Wikidata: a new platform for collaborative data collection. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 1063–1064, New York, NY, USA, 2012. ACM.

[WHZC09] Pu Wang, Jian Hu, Hua-Jun Zeng, and Zheng Chen. Using Wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 19:265–281, 2009. 10.1007/s10115-008-0152-4.

[WW10] Fei Wu and Daniel S. Weld. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL ’10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[WWA$^+$08] Daniel S. Weld, Fei Wu, Eytan Adar, Saleema Amershi, James Fogarty, Raphael Hoffmann, Kayur Patel, and Michael Skinner. Intelligence in Wikipedia. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI’08, pages 1609–1614. AAAI Press, 2008.

[ZMG08] Torsten Zesch, Christof Müller, and Iryna Gurevych. Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC), electronic proceedings*, Mai 2008.

[ZP13] Marcin Zajc and Adam Przepirkowski. Distant supervision learning of DBpedia relations. In Ivan Habernal and Vclav Matouek, editors, *Text, Speech, and Dialogue*, volume 8082 of *Lecture Notes in Computer Science*, pages 193–200. Springer Berlin Heidelberg, 2013.