ISBN 978-82-471-4709-2 (printed ver.)
ISBN 978-82-471-4710-8 (electronic ver.)
ISSN 1503-8181

Naimdjon Takhirov

Doctoral thesis

NTNU – Trondheim
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2013:289

NTNU
Norwegian University of
Science and Technology
Thesis for the degree of
Philosophiae Doctor
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Computer and Information Science

NTNU

Doctoral theses at NTNU, 2013:289

Naimdjon Takhirov

Extracting Knowledge for Cultural Heritage Knowledge Base Population

NTNU – Trondheim
Norwegian University of
Science and Technology

Naimdjon Takhirov

Extracting Knowledge for Cultural Heritage Knowledge Base Population

Thesis for the degree of Philosophiae Doctor

Trondheim, July 2013

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Computer and Information Science

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

© Naimdjon Takhirov

ISBN 978-82-471-4709-2 (printed ver.)
ISBN 978-82-471-4710-8 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2013:289

Printed by NTNU-trykk

Abstract

The entity-oriented description of the world is a major, current trend motivated by the need for semantic services that can support the human need of finding information, learning and discovering new knowledge, and broadening the existing knowledge horizons. Entities, managed in semantic knowledge bases, have the potential to be the backbone for these new and innovative services. Therefore, automatically extracting facts from various data sources and populating knowledge bases a challenge studied in this work.

This thesis proposes methods for knowledge extraction for the cultural heritage domain. Extracting knowledge from the cultural heritage metadata is by no means a trivial task and there are often problems with missing or ambiguous information. Therefore, an inherent part of this work is dedicated to developing pattern-based techniques to extract knowledge from natural language documents to complement and supplement the knowledge we extract from metadata. However, the proposed framework is not limited to only work in conjunction with metadata extraction – it additionally supports independent, continuous mode operation, i.e. patterns learned during extraction are used to subsequently mine new knowledge.

In summary, the main contributions of this thesis are:

- FRBR-ML: a generic framework for exploiting metadata which includes: (i) a method to extract entities, attributes and relationships from existing legacy metadata, (ii) novel techniques for correction, enhancement and semantic enrichment of the metadata, and (iii) metrics to assess the quality of extraction.
- SPIDER: a prototype that supports extraction of relational facts at Web-scale. Contrary to most knowledge extraction approaches, we tackle the problem of uniquely identifying entities both to extend their list of spelling forms and to facilitate the matching to LOD entities. Furthermore, in addition to the flexible pattern definition scheme, SPIDER enables a provenance-aware extraction method, which prudently refines extracted facts by considering the PageRank and SpamScore as well as the relevance score of the source document.
- KIEV: a prototype that takes the development of SPIDER into the next stage, namely by enabling verification of facts using two evidence-based techniques:

(i) classification to check the type of relationship with a machine learning approach, and (ii) linking to discover local entity's correspondence in another data source was leveraged using existing semantic knowledge bases.

- FRBRpedia: a prototype that is developed to utilize the attribute-oriented linking of local entity to the corresponding entities in external semantic knowledge bases. As one of the most basic tasks of knowledge base population, linking demonstrates the power of Linked Data applications. Finally, linking is commonly seen as a required step for putting the data on LOD.

The methods and solutions proposed in this thesis provide a solid foundation for automatically populating knowledge bases using wide range of sources. The feasibility of the approaches presented have been tested through experimental evaluation using real-world datasets. A general conclusion is that complementing knowledge extraction from metadata with the external sources results in less amount of missing and ambiguous information and in a more complete knowledge base.

Keywords

Knowledge bases, Knowledge extraction, Metadata, FRBR, Entity matching,
Linked Data

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for the partial fulfilment of the requirements for the degree of philosophiae doctor.

This doctoral work has been conducted at the Data and Information Management group (DIF), Department of Computer and Information Science (IDI), Faculty of Information Technology, Mathematics and Electrical Engineering (IME). The work has been performed under the supervision of Associate Professor Trond Aalberg. Professor Ingeborg Sølvberg and Associated Professor Heri Ramampiaro were assigned as co-supervisors.

Acknowledgements

This work has been made possible through the assistance of number of people. First of all, I would like to thank Trond, my main supervisor, for his guidance and always being available for discussions, for giving a great deal of freedom and responsibility, and most importantly, for providing me with a framework for discovering my academic interests and means to completing my research. I would also like to thank my co-supervisors Ingeborg and Heri. Ingeborg has been a great source of advice for me from the time of my master's studies and I appreciate hers and Heri's constructive comments in the shaping of this work.

Fabien has been an unofficial supervisor, and through many discussions with him, we have developed several ideas which have been articulated in a number of publications. His almost always critical view on my ideas, greatly improved the way I think about research and a particular problem.

I would also like to thank Krisztian. I have learned a great deal from his IR expertise during our brief collaboration.

I was fortunate enough to intern at Yahoo! during my studies. The privilege of having worked with the bright people and in a friendly environment made my stay an incredibly valuable experience. Specifically, I would like to express my gratitude to my mentors Peter and Roi.

I am indebted to colleagues at the department both fellow PhD students and research, technical and administrative staff. All the nice sets of informal chats during coffee breaks and lunches with Massimiliano, Simon, João, Robert, Nattiya, Orestis along with friends and academic colleagues too numerous to enumerate here, made my stay at the department a memorable experience. In particular, I would like to acknowledge Massimiliano for sharing office with me, for providing a fine working environment and for many fruitful discussions on various topics.

My family has been the primary source of support and encouragement and I appreciate the great interest my parents have shown in my work. I will be ever grateful to my grandfather for helping me discover the field of study, and I am sad that he has not lived to see me graduate. Finally, working all these years on this dissertation often required being away from my family and I could not have completed it without the constant support, patience and unwavering love of my wife Oksana, and our two wonderful children.

Naimdjon Takhirov
Trondheim July 1, 2013

Contents

List of Tables

List of Figures

List of Algorithms

Part I

Introduction, Background and Preliminaries

1 Introduction

"Information wants to be free, because the cost of getting it out is getting lower and lower all the time."

- Stewart Brand.

The entity-oriented description of the world is a major, current trend motivated by the need for semantic services that can facilitate the human need of finding information, learning and discovering new knowledge, and broadening the existing knowledge horizons. Knowledge bases contain semantic descriptions of entities, and they typically include tools and services to support their exploitation. To utilize knowledge bases at large-scale, they must be automatically populated with entities, their attributes and relationships. This thesis tackles this issue through a framework that considers cultural heritage data for knowledge base population.

The framework proposed in this thesis accepts cultural heritage metadata as input and extracts described entities, attributes and their relationships. To address the issues of ambiguity and missing information about entities, the approach exploits natural language documents as an additional source to not only complement and supplement the results of metadata extractions, but also to extract new information about entities.

This chapter presents motivation behind this work, research questions and objectives and the main contributions.

1.1 Motivation

In the recent years, knowledge bases have had an increasing impact on the way we create, distribute and make use of information. A knowledge base is a key asset in making information understandable to machines and provides a basis for many new and innovative semantic-aware services such as semantic search [49, 159], question answering [193] and other advanced applications that require rich semantics. Furthermore, semantic information encoded in knowledge bases plays a significant role as an aid in data cleaning applications [46], record linkage / entity resolution [51], entity linking [175], named entity disambiguation [40, 107] and other data integration tasks in general [8, 160, 161].

However, populating a knowledge base is a tedious task that on the one hand requires a variety of structured and unstructured sources, and on the other hand, the task depends on tools and services that are able to mine semantic information from those sources. Although, manual construction and maintenance of knowledge bases is possible and may result in high accuracy, it is a slow process and major bottleneck to applications that require large amount of semantically structured knowledge.

Using a large body of existing unstructured and structured data to automatically extract knowledge is a solution, but this requires methods that are able to cope with noise in the text and the scale of input data. To be of any large-scale use, knowledge bases should be populated and linked automatically within a larger digital information space in an effective and an efficient manner.

The increased interest in knowledge bases that we have witnessed in the last decade is to a large degree encouraged by the development of the Semantic Web in which information is given an explicit meaning in order to make the processing easier for machines. The Semantic Web established a technological framework that has enabled building knowledge bases. A knowledge base is a special type of database of semantically organized knowledge with an additional feature of being machine processable.

Knowledge bases are connected to other knowledge bases within a larger space of structured data that exists on the Web of Data, which also commonly is called *Linked Open Data* (LOD) [27]. By connecting a knowledge base to other sources, the knowledge is shared, easily discovered by others and properly integrated.

The use of RDF and URIs in LOD is a foundation that enables linking of entities – the bricks of knowledge bases – not only among and between entities within the same knowledge base, but also across the knowledge bases in the LOD cloud, which consists of many datasets of diverse nature and a total of more than 31 billion factual assertions (triples)[1].

Figure 1.1 illustrates a typical layered architecture of populating a knowledge base and publishing it in the LOD cloud. The different layers of this architecture are:

- **Data input**. There are two main types of data that are interesting to exploit as a source for LOD. The structured data as input is the data that resembles some kind of a structure (e.g. metadata records in XML or as CSV files). The second type of source that can be used is textual sources such as Web documents[2] accessible online.

- **Data preparation**. The data preparation is a crucial phase as it affects all subsequent steps in the pipeline. For text documents, entities and their relations need to be identified first in order to derive to an entity-centric description. For existing structured data, interpreting and eventually correcting may be needed. The extracted entities and relations from both sources should be linked to corresponding entities in the LOD cloud. Enriching linked entities is an optional step before storing data in the data storage layer. This phase ensures that the data is converted into a suitable format, such as RDF.

- **Data storage**. In the storage layer, there are several types of storage mechanism that can be used. For example, a set of static RDF files obtained from the data preparation phase, may be served with a Web Server; for an optimized storage and retrieval of triples, triplestores may be chosen.

- **Data publication**. The data must be exposed as part of the LOD in order to actually use it for various purposes. Publishing and exposing this semantic data may serve a broad range of applications such as mashups that combine data from several sources, each of which is accessed through an exposed Web API.

The upper part (highlighted) of the Figure 1.1 is where the work of this thesis can be placed. More specifically, it is the data preparation phase of the overall process that is of particular importance to this work.

[1] `http://lod-cloud.net/state/` (Last checked April 2013)

[2] Although HTML has some structure (such as tags), those structural elements are often considered mainly for presentation purposes.

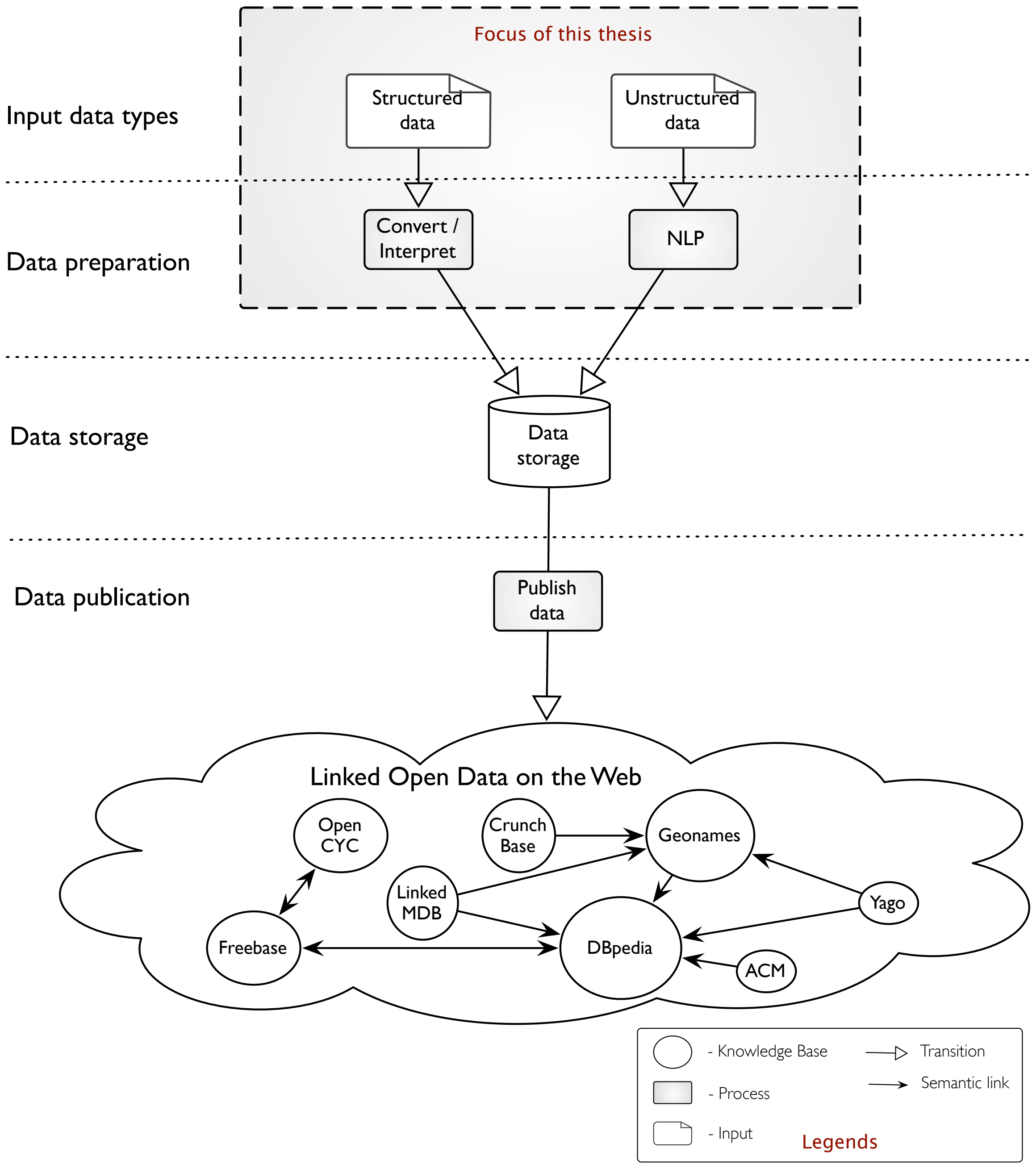


Figure 1.1: Layered Architecture of Publishing Knowledge Bases in Linked Data.

The **main hypothesis** of this thesis is that the results of metadata extraction can be improved through enhancements, corrections and extraction of complementary information from other sources, thereby utilizing the synergy that can be achieved when combining and integrating complimentary information resources. Some examples of issues that may require such improvements are:

- **Data heterogeneity** is an inherent challenge of most data integration efforts. Data is often designed by different people using various formats and structures. Therefore, data sources are often structurally, syntactically and semantically heterogeneous.

- **Disambiguation**, which is still an open area of research, is a process of identifying which sense of a word is used in a particular context when the word has has more than one meaning. For example, Oslo without any context may of course refer to Oslo - the Norwegian capital but it could also refer to Oslo, MN USA - a small town in the US state of Minnesota. In short, the semantic information of interest should be made explicit to avoid ambiguity.

- **Missing information** is a problem long known to the knowledge discovery and data mining community. Most collections inherently include missing records, fields or feature values. Extraction of both missing and complementary information is one of the crucial steps in integrating information in knowledge bases. Particularly extracting missing information from the Web poses new challenges such as the recognition of a noise from the high quality knowledge embedded in the text as well as the scale of input data.

- **Entity linking problem** can be seen as one of the ways to explicitly express relationships denoting a semantic association between entities. It is related to the problem of Named Entity Recognition (NER) in determining the identity of entities mentioned in the text, but linking additionally requires a knowledge base to explicitly link the mention of the entity to the entity that exist in the external knowledge base. Especially, in Linked Data, entities should be interlinked both within and across knowledge bases in the LOD cloud.

- **Semantic enrichment issues** are often tackled to improve recall and completeness in many applications. Semantic enrichment is an additional power enabled by the data model of the Linked Data and it can be utilized to enrich data with more meaning which allows machines to do more fine-grained processing.

1.2 Cultural Heritage Documentation

Cultural heritage is an expression we often associate with the legacy of physical artifacts, a tradition, customs and practices of a society, artistic values and the inherited knowledge from the past, sustained in the present and preserved for the future generations. According to [109], cultural heritage can be characterized into the following three subareas:

- Tangible;
- Intangible;
- Natural.

Tangible cultural heritage refers to concrete physical artifacts bearing a materialistic characteristic and examples of such artifacts are historical buildings, monuments, books, work of art and other objects that are considered worth further preservation. On the other hand, intangible cultural heritage includes non-physical phenomenon of a culture such as traditions, language, folklore and other aspects maintained by the social customs[3]. Finally, natural cultural heritage consist of landscapes, biodiversity (flora and fauna), geodiversity (geological elements) considered culturally important[4].

Traditionally, memory organizations have been in charge of organizing and preserving our cultural heritage. Such organizations include libraries, archives and museums of various kinds (history, geological and zoology museums), botanical gardens etc. Therefore, the information that is managed by these memory institutions is an important global documentation of the intellectual and artistic endeavor of mankind.

In this work, the focus (as an initial input data) is on the bibliographic information that is typically managed by libraries. More specifically, the information domain of interest is the documentation of entities and relationships of intellectual artistic endeavor described in bibliographic databases.

[3]http://en.wikipedia.org/wiki/Intangible_cultural_heritage (Last checked May 2013)

[4]For example, United Nations Educational, Scientific and Cultural Organization (UNESCO) maintains a list of world's heritage list, http://whc.unesco.org/en/list (Last checked May 2013)

1.3 Research Questions

The main objective of this thesis is to develop a generic framework that is capable of extracting entities from cultural heritage metadata along with the complementary information from natural language documents to populate a knowledge base. Furthermore, the framework includes methods for automatically verifying and linking entities to other knowledge bases. These challenges are manifested in the following research questions:

RQ1. *Can we identify entities and relationships described implicitly in structured data?*

The structured data needs to be carefully processed in order to extract described entities, their attributes and relationships for semantic integration in knowledge bases. Basically, this question deals with the interpretation of a correct set of entities that is described in metadata including the type of entity. A lack of proper identifiers and relying on identification based on descriptive text, additionally poses a significant challenge.

RQ2. *How to extract complementary information about entities and relationships?*

A semantic integration in the knowledge bases requires an acceptable level of completeness of information about entities and minimizing the amount ambiguous and missing information. A local collection is rarely sufficient and other sources of information such as Web documents often need to be exploited to reconcile fully fledged description of an entity. As a huge repository of static and dynamic documents as well as social media content, the Web is a potentially large amount of high-quality knowledge. Another interesting aspect of exploiting Web as a source of complementary information is that we can explore the synergy that can be achieved when extracting knowledge from structured and unstructured sources.

RQ3. *What methods can be used to verify extracted knowledge?*

Verification of extracted information is one of the key processes to address the quality issues of knowledge bases. Not only the reusability but also the success of published data depends on several factors and one of the crucial factors is to which extent the fact can be verified. An entity-oriented interpretation of metadata and extraction of complementary information from the Web to remove ambiguity, is often seen as one part of the larger and general task of semantic integration.

RQ4. *How to perform linking of extracted entities to entities in other semantic knowledge bases?*

Linking entities is one the main ingredients integrating the knowledge base in the world of LOD. The main challenge in exploring this research question is: given a local entity, how to effectively discover its corresponding entity or entities in other knowledge bases. The result of this step is either creation of an *"same as"* relationship for the local entity that will link to equivalent entities in other knowledge bases or the reuse of already established identifier.

1.4 Research Methods

Descriptions of research methods differ substantially by research area. In computer science research there are several research methods. When we talk about computer science research method, usually initial questions that comes to our mind are "what kind of data is used in the evaluation", "how the data was collected", "how to interpret results" etc. These questions help to better understand and assess the claims of the research, the quality of the research, reported results and how to interpret those results. Furthermore, significant difference in research methods becomes the source of problems for comparing and evaluating the results.

There are two well-known paradigms that characterize research in information systems: (i) the behavioral (or natural) and (ii) design science [103]. The theoretical focus of behavioral science paradigm is an attempt to develop and explain principles, laws and rules that predict phenomenon both organizational and human, surrounding the design and implementation of a particular research. In an organizational setting, the purpose of an organization is to improve the effectiveness and efficiency of an organization. To achieve this goal, theories of behavioral science increase awareness of researchers about the interaction between people, technology and an organization. Furthermore, the decisions made in a development methodology impact and are impacted by theories of behavioral paradigm.

The design science research, on the other hand, comes from the engineering sciences and is essentially considered as problem solving paradigm via a systematic form of designing [84, 196]. The design science paradigm seeks to create innovations through artifacts which define the hypothesis through design, implementation, and use of information systems. In design science, the research is addressed via building and evaluating new and innovative artifacts that meet specific identified needs [103]. Many will argue that the recent technological advances are mainly the result of creative design science achieved through new and innovative artifacts. Internet and the Web, personal computing, databases are all examples of phenomenal innovations of the past several decades and sometimes had unintended impacts [102, 165].

This thesis approaches the research questions posed via a design science method as it is the most appropriate and requires artifacts to prove the claims made in this work through experimental evaluations. For each of the questions, the following steps are taken:

- **State the observation** - identify a problem and search for existing solutions. This step is accomplished through an extensive literature review.
- **Formulate a hypothesis** - which is a possible tentative outcome in a form of improvement over existing approaches or proposing new techniques. A literature review reveals the advantages and disadvantages of existing methods. Then, formulate a hypothesis step helps to understanding potential improvements or proposing a new approach.
- **Perform initial implementation** - bring the hypothesis into a reality by creating a software prototype.
- **Evaluation** - comparison of implemented solution with the state-of-the-art approaches. The evaluation step is an important way of analyzing approach. In this work, we use both well-known real-world datasets where possible and synthetic ones. In particular, the Clueweb2009 dataset which is used by sev-

eral evaluation initiatives such as TREC[5] and TAC[6]. The synthetic dataset includes manually assessed relevance judgments by experts as well as a sample of entities. Furthermore, evaluation metrics are included in order to assess efficiency, effectiveness, performance, and other criteria.

- **Share** - share the results with the community in a well-known research venue (a publication).

1.5 Contributions

The major contributions of this work are:

- An approach for exploiting bibliographic metadata to support fine-grained extraction of entities, their attributes and relationships. This metadata extraction approach proposes additionally a technique for enhancement of the original metadata. Furthermore, the approach includes semantic enrichment of existing metadata by using external knowledge bases and services. Additionally, we propose evaluation metrics that can be used to assess the quality of the extraction.
- An approach to interpret metadata beyond the typical realm of the collection managed by the memory institutions. We consider FRBR as an underlying conceptual domain model and demonstrate that such a network of entities and relationships can be extracted out of product descriptions on the Web.
- Extraction of new and complementary entity information from unstructured sources. The proposed method is an extension of existing pattern-based techniques. In particular, we incorporated several additional components: flexible definitions of patterns with a similarity function to compute the semantic distance between patterns, provenance-aware confidence score which consists of normalized sum of relevance score, page spam-score and PageRank, identifying the alternative labels of entities, and finally the large-scale aspect to deal with Web-scale knowledge base population.
- Verification of extraction results with evidence based methods in order to populate a knowledge base. Two verification methods were proposed: clas-

[5]Text REtrieval Conference, http://trec.nist.gov (Last checked April 2013)
[6]Text Analysis Conference, http://www.nist.gov/tac/ (Last checked April 2013)

sification and linking. With the classification method, we train a generic classifier with a range of features. For the linking, we perform automatic context-driven discovery of corresponding LOD entity.

- Present a two-step linking technique that establishes connection between equivalent entities. In this approach, given the local entity we discover corresponding entities across the data sources in the LOD cloud. To tackle the problem of large data volume, we first reduce the search space with a blocking mechanism. In the second step, the attribute-aware linking between the local entity and the LOD entity is achieved.

1.6 Publications

The contributions of this work has appeared in the following peer-reviewed publications:

P1. Naimdjon Takhirov, Trond Aalberg, Maja Žumer. *An XML-based Representational Document Format for FRBR*. In Proceedings of the first International Symposium on Web Intelligent Systems and Services (WISS'2010, Hong-Kong, People's Republic of China), volume 6724 of Lecture Notes in Computer Science. Springer-Verlag, Berlin-Heidelberg 2011.

P2. Fabien Duchateau, Naimdjon Takhirov, Trond Aalberg. *FRBRPedia: a Tool for FRBRizing Web Products and Linking FRBR Entities to DBpedia*. In Proceedings of 11th ACM/IEEE Joint Conference on Digital Libraries (JCDL'2011, Ottawa, ON, Canada), ACM, New York, USA 2011.

P3. Naimdjon Takhirov. *Semantic Reuse of Existing Metadata, A model-based perspective*. IEEE TCDL bulletin, 8(1) February 2012.

P4. Naimdjon Takhirov, Fabien Duchateau, Trond Aalberg. *Supporting FRBRization of Web Product Descriptions*. In Proceedings of the 15th International Conference on Theory and Practice of Digital Libraries (TPDL'2011, Berlin, Germany), volume 6966 of Lecture Notes in Computer Science. Springer-Verlag, Berlin-Heidelberg, 2011.

P5. Naimdjon Takhirov, Fabien Duchateau, Trond Aalberg. *Linking FRBR Entities to LOD through Semantic Matching*. In Proceedings of the International Conference on Theory and Practice of Digital Libraries (TPDL'2011, Berlin, Germany), volume 6966 of Lecture Notes in Computer Science. Springer-Verlag, Berlin-Heidelberg, 2011.

P6. Naimdjon Takhirov, Fabien Duchateau, Trond Aalberg, Maja Žumer. *FRBR-ML: FRBR-based Framework for Semantic Interoperability*. Journal of Semantic Web, IoS Press, 2012.

P7. Naimdjon Takhirov, Fabien Duchateau, Trond Aalberg. *An Evidence based Verification Approach to Extract Entities and Relations for Knowledge Base Population*. In Proceedings of the 11th International Semantic Web Conference (ISWC'2012 main research track, Boston, MA, USA), volume 7649 of Lecture Notes in Computer Science. Springer-Verlag, Berlin-Heidelberg 2012.

P8. Naimdjon Takhirov, Fabien Duchateau, Trond Aalberg, Ingeborg Sølvberg. *An Integrated Approach for Large-scale Relation Extraction from the Web*. In Proceedings of the fifteenth International Asia-Pacific Web Conference (APWeb'2013, Sydney, Australia), volume 7808 of Lecture Notes in Computer Science. Springer-Verlag, 2013.

P9. Naimdjon Takhirov, Fabien Duchateau, Trond Aalberg, Ingeborg Sølvberg. *A Tool for Extracting Semantic Relations from the World Wide Web*. (under submission).

1.7 Additional Publications

Additionally, several publications have been produced during the course of the work on this thesis but are not included in this thesis due to the level of their pertinency to the core topic of the thesis.

P10. Naimdjon Takhirov. *Supporting Creative Work in Educational Digital Libraries*. In Proceedings of the ACM/IEE Joint Conference on Digital Libraries (JCDL'2011, Ottawa, ON, Canada), ACM, 2011.

P11. Naimdjon Takhirov, Fabien Duchateau. *A Cloud-based and Social Authoring Tool for Video*. In Proceedings of the ACM Conference on Document Engineering (DocEng 2011, Mountain View, CA, USA), ACM, New York, USA 2011.

P12. Krisztian Balog, Naimdjon Takhirov, Heri Ramampiaro, Kjetil Nørvåg. *Multi-step Classification Approaches to Cumulative Citation Recommendation*. In Proceedings of the 10th International Conference in the RIAO series: Open research Areas in Information Retrieval (OAIR'2013, Lisbon, Portugal), ACM, New York, USA, 2013.

P13. Naimdjon Takhirov, Ingeborg Sølvberg, Trond Aalberg. *Organizing Learning Objects for Personalized eLearning Services*. In Proceedings of the European Conference on Digital Libraries (ECDL'2009, Corfu, Greece), volume 5714 of Lecture Notes in Computer Science. Springer-Verlag, Berlin-Heidelberg, 2009.

P14. Naimdjon Takhirov, Ingeborg Sølvberg. *Adaptive Personalized eLearning on top of Existing LMS*. In Proceedings of the Joint Conference on Digital Libraries (JCDL'2009, Austin, TX, USA), ACM, New York, USA, 2009.

1.8 Structure of the Thesis

The remainder of the thesis is structured as follows.

Part I Introduction, Background and Preliminaries.

- **Chapter 2** introduces knowledge bases and outlines traditional applications which can benefit from the use of knowledge bases. This chapter further discusses the various forms of knowledge bases along with typical content stored in knowledge bases as well as the building blocks of knowledge bases: entities, their attributes, and relationships.

  The second part of this chapter deals with the knowledge extraction. It presents the existing open-domain and ontology-based paradigms for knowledge extraction and lists some of the traditional sources for knowledge extraction.

  The last part of this chapter is about metrics. First, quality measures used for evaluating various parts of the thesis are discussed and finally similarity metrics that are employed are presented.

Part II State of the Art and Overview.

- **Chapter 3** starts with reviewing existing knowledge extraction methods. It then briefly surveys current extraction approaches that use natural language documents. This chapter further reviews related fields such as automatic ontology construction and other knowledge base population methods. This chapter additionally presents the entity matching which is exploited in various parts of the approach.
- The topic of **Chapter 4** is to explore the use of metadata as a source of knowledge extraction. It starts by surveying the current state of bibliographic catalogs and the use of FRBR model as an underlying conceptual model for the bibliographic universe. This chapter continues with the discussion of the ontologies and vocabularies for the FRBR model. The last part reviews the existing approaches to interpreting metadata using sound conceptual models.
- **Chapter 5** serves as an overview of the framework this thesis proposes. It provides a generic outline over the different parts involved for the knowledge population task set forth for this work.

Part III Interpreting Metadata.

- **Chapter 6** presents – FRBR-ML – an approach to extract entities, attributes and their relationships with a particular focus on representation, semantics and metrics. This chapter further describes experimental evaluation that has been performed on a real world dataset.
- **Chapter 7** continues exploring conceptual model for interpreting metadata and evaluates the use FRBR model for product descriptions on the Web. To accomplish this task, the approach makes use of various Web services exposed via Web APIs and the evaluation is performed using the real-world Amazon dataset.

Part IV Extracting Entities and Relations.

- Part IV in general and **Chapter 8** in particular focus on extracting entities and relations from natural language documents. This chapter introduces the

SPIDER prototype that tackles the issue of extraction and discusses the results of experimental evaluation performed at Web-scale collection.

- **Chapter 9** further develops the ideas of the SPIDER prototype and proposes a verification approach through the KIEV prototype. This chapter further presents the experiments performed to evaluate the practical application of the evidence-based verification approach.

Part V Linking.

- **Chapter 10** presents an attribute-based approach to link the entities extracted both from the metadata and natural language documents. This chapter additionally evaluates the effectiveness of the proposed approach through an experimental evaluation and the performed experiments consider linking of FRBR work entities to their corresponding DBpedia entities.
- To support the approach presented in the previous chapter, **Chapter 11** demonstrates an application of linking through FRBRpedia prototype and its use in the FRBR domain.

Part VI Conclusion.

- **Chapter 12** concludes the work presented in this thesis with a summary of major contributions. Finally, this chapter outlines some of the future perspectives that can be explored.

2 Background and Preliminaries

"Knowledge is power".

- Francis Bacon.

2.1 Context

There has been an increasing interest in knowledge bases in which descriptions are entity-centric in the database, IR, NLP and Semantic Web communities. For example, search for information is often conducted around entities and more than half of online queries submitted to search engines, involve search for specific entities instead of documents [116]. Various evaluation initiatives, designated conference tracks and numerous workshops with specially focused theme around knowledge bases, have been taking place recently to address challenges associated with the research and development in knowledge-enhanced information access.

The overall purpose for providing fine-grained descriptions at entity level is driven by the need to utilize individual item by the consumers rather than having to download entire documents and process each document to retrieve entities. This enables a greater reusability of data since there seldom is limitation on the use of published data and consumers are able to take advantage of the data and develop new services on top of that data. Each entity is individually addressable unambiguously via a URI, therefore providing contextual information about that entity becomes easy.

In addition to the datasets that are being shared online, much of the knowledge on the Web is encoded in natural language documents which are primarily targeted for human consumptions. This enormous amount of content is potentially a source of high quality structured knowledge when knowledge extraction techniques are applied to retrieve machine processable structures from free texts.

The rest of this chapter will present background information about knowledge bases, and sources for extracting content for knowledge bases. Additionally, the basic idea of knowledge extraction from text and structured sources will be discussed. Finally, we present the quality measures used for the evaluation later in the experiments as well as the similarity metrics employed for entity matching.

2.2 Knowledge Bases

Even though knowledge bases are a popular field of study , there is no single precise globally accepted and shared definition. Because of its generality, the term is used interchangeably as well as to refer to related, but different things. Large companies have for decades developed and maintained repositories in which employees are encouraged to contribute with knowledge, expertise and experience [90]. This codified knowledge becomes explicit, readily available across the organization and applied in new situations [6]. Other similar kind of knowledge bases include wikis, frequently asked questions (FAQs), tutorials etc. These kind of knowledge bases are created and maintained mainly for human consumption and primarily contain natural language text.

As more and more data is stored in a digital form, the need for imposing semantics on the data has significantly increased. The main goal of introducing semantics on the data is to enable automatic and intelligent processing of this data by machines. A machine processable knowledge base is therefore a special type of database for organizing knowledge [111, 130]. Researchers in the field of Artificial Intelligence (AI) and especially its subfield Knowledge Representation and Reasoning have for a long time been interested in equipping machines with human-level intelligence by representing knowledge in a way that enables inferencing over that knowledge. AI research has not reached the goal of passing human-level intelligence to machines yet but developments indicate that the state of the art is advancing at a rapid pace. On the other hand, Database (DB), Information Retrieval (IR) and Natural Language Processing (NLP) research fields have had a profound progress in the field of intelligent information access recently by focusing on sub-areas: entity recognition, question answering, semantic and entity search, recommender systems. In many aspects, in all of these sub-areas, organizing knowledge and knowledge itself play a crucial role [14, 71, 106, 141, 155, 159, 179].

Building a machine processable knowledge base is not a new topic. Douglas Lenat started the Cyc project [121] in the 1980s to assemble a comprehensive everyday common sense knowledge database. The main objective was to enable various AI applications to perform human-like inferencing. The recent wave of attention to the Semantic Web and Linked Data has revitalized the interest in building high quality semantic knowledge bases. However, much of the human knowledge is either manually encoded in legacy databases in a structured form or is available in text documents. Therefore, automated methods need to be developed in order to exploit this wealth of information and making the knowledge explicit for easier sharing, and reuse in broad range of semantically enhanced applications.

2.2.1 Knowledge Base-enhanced Applications

Knowledge bases include ontological descriptions of concepts and relations and as such are exploited for the purpose of knowledge sharing. The encoded knowledge in knowledge bases is explicitly stated using concept definitions as well as axioms and constraints. They serve variety of purposes and their use can be highly diverse. Even though knowledge bases are exploited and used in numerous ways, the following type of applications can generally characterize their usefulness:

- **Organizing data** is important functionality in the large collections of documents. A knowledge base is often used to organize and structure the information and plays a key role of enabler of semantic services such as exploratory interfaces which facilitate both simple and faceted browsing the vast array of information. An additional service that can be supported is serendipitous and intelligent information access [79].

- **Semantic search** seeks to improve the search results by understanding the intent expressed in the query. There are basically two types of semantic search:

  – Searching and discovering information in the Web of Data is sometimes also referred to as semantic search [30, 159, 168]. This type of semantic search is applied to structured dataset, often RDF data to overcome the complexity of SPARQL queries and user's who prefer simpler means of specifying keyword queries. The Semantic Search Challenge[1] was organized to exactly address the issue of searching for entities in the RDF data. For any query, participating systems were required to return a ranked list of entities identified by their URIs. The dataset used in the challenge was a sample of Linked Data crawled from publicly available sources (excluding blank nodes).

– Ontology-based search denotes the use of ontology to improve the accuracy of search results by capturing the explicit meaning of a keyword query [190] or providing a way to guide a user through the search space [176]. Ontology-based search exploits the knowledge encoded in the ontology during search [36]. These improvements aid with word sense disambiguation [157], keyphrase extraction [94], query expansion [26], query formulation [183], and multi-lingual information retrieval [96]. An evaluation of the use of ontologies and their fitness to a variety of search tasks is presented in [177].

Semantic search has recently received a renewed interest as the major search engines are now developing or have already developed products to support some aspects of semantic search. The knowledge graph from Google[2] was publicly introduced in 2012 and the primary objective was to enhance search results. The source of content for the knowledge graph comes from a variety of sources (CIA World Factbook, Wikipedia, Freebase, etc.) and at the time of this writing includes some 570 million entities and 17 billion facts about them. Microsoft Bing has recently announced a similar product called Snapshot/Satori[3].

- **Semantic interoperability** is another area where knowledge bases can be used to integrate and link information from heterogeneous sources. For example, by creating *owl:sameAs* links between entities "The Lord of The Rings" (in an American library data source) and "Ringenes Herre" (in a Norwegian

[1]http://semsearch.yahoo.com/ (Last checked April 2013)
[2]http://www.google.com/insidesearch/features/search/knowledge.html (Last checked April 2013)
[3]http://www.bing.com/blogs/site_blogs/b/search/archive/2013/03/21/satorii.aspx (Last checked April 2013)

library data source) it is known that even with different languages these two entities refer to the same real-world entity. Integrating the two data sources enables therefore a semantic interoperability since entities are interlinked and can be shared in an unambiguous manner. Finally, the integration of data sources enables additional features such as semantic enrichment and data cleaning.

2.2.2 Knowledge Base Spectrum

The simplest form of a knowledge base with a least expressive structure is a *catalog* including various forms of controlled vocabularies (Figure 2.1, adapted from [140]). A controlled vocabulary typically includes a mechanism by which terms are identified unambiguously and are subsequently used for indexing and retrieval. As shown in Figure 2.1, *glossaries* are the next in the spectrum of definitions. Glossaries are list of terms with their human-readable definitions.

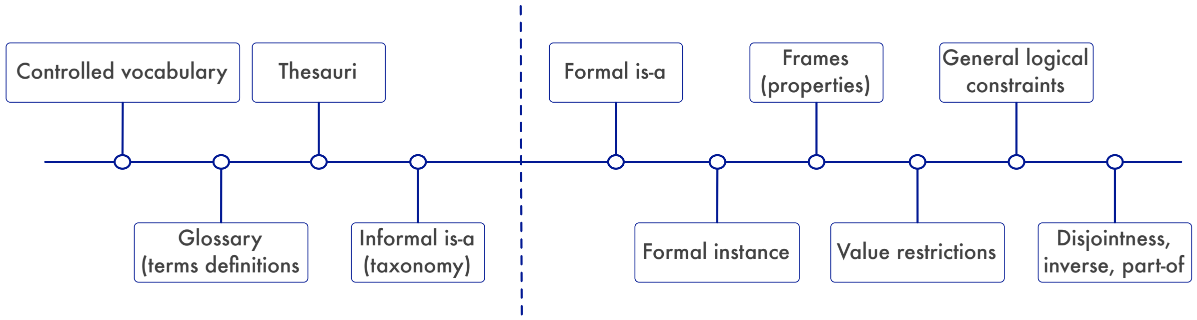


Figure 2.1: A Knowledge Base Spectrum.

By adding more structure and relations to terms, one can derive to more semantics – similar to *thesaurus* like structures. In thesauruses, relationships (such as synonym relationships) are used to express the range dependencies and connections between the terms.

Terms organization in a tree-like structure has been practiced for a long time with the main goal to support better navigation (e.g. from general concept to more specific). This type of organization of terms is clearly informal and sometimes this organization is called *taxonomies* or *informal is_a*. A strict subclass relationship is not held in the structures in such hierarchies and an instance of a more specific class is often also an instance of the more general class but that is not always enforced.

The kind of knowledge bases in the spectrum discussed up to now are primarily for human use which is illustrated with a line that divides the spectrum into two. In [140], the right part of the spectrum is called *ontologies*. The strict *is_a* subclass hierarchies enable more intelligent processing. These formal structures support inheritance with *formal instance* relationships, i.e. if A is a superclass of B, and C is a subclass of B, then C is a subclass of A too and an object of instance of C is also necessarily an instance of A.

Support for properties or attributes are enabled by the *frames* which are the next point in the spectrum. A frame is an AI data-structure for representing a "stereotyped situation" and is said to represent a network of nodes and relations [148]. For example, iPad has the property *price* with value 500$. The usefulness of properties are more visible when they are inherited down the subclass hierarchy. For example, a general category Apple product may have a *price* property which will be inherited by all products.

The next point in the spectrum is the exploitation of *value restrictions*. It can be thought of a constraint on a value of given class. For example, a *price* property may only be allowed to have numbers. Continuing in the spectrum, requirements for expressiveness grow and the next point is the ability to state arbitrary *logical statements*. An example of a logical constraint is that for example a property *deathDate* must be after *birthDate*. Finally, the most expressive knowledge bases include first-order logic constraints and more detailed relationships such as disjoint classes, disjoint coverings, inverse relationships, part-whole relationships.

2.2.3 The Content of Knowledge Bases

In a machine processable knowledge base, entities and their properties are primarily stored in a form of statements. The notion of a statement can reflect several things and a clear definition of the term will depend on the type of knowledge base that is of concern. For instance, for a commonsense knowledge base like Cyc [121] the following will be considered as a statement:

```
Barack_Obama is_a UnitedStatesPresident
```

This statement asserts the fact that *Barack_Obama* belongs to a group of people who have been United States Presidents. The form of this statement is called *triple*, because it contains an atomic data item in the form of *<subject> <property> <object>*. This kind of expression is also used in the Resource Description Framework (RDF).

The power of semantic knowledge bases is additionally supported by the fact that they include interfaces with which the content stored in the knowledge base is exposed through common querying interfaces such as SPARQL[4] endpoints as well as updating and fetching data from a knowledge base in the REST style[5]. Consequently, this feature enables low-barrier methods for consumption of content of the knowledge base.

2.2.4 Ontologies

A typical implementation of an ontology[6] is a formal representation of a domain of knowledge. Ontologies are at the heart of many Semantic Web applications to provide a shared conceptualization of domain. The most well-known definition commonly cited in the Semantic Web and Knowledge Representation communities is the one by Tom Gruber [93], i.e.:

> *"An ontology is an explicit and formal specification of a conceptualization of a domain of interest".*

An ontology consists of classes, properties, instances and axioms. For example, the Figure 2.2 depicts a typical superconcept-subconcept relationship, also called *is_a* hierarchy. The relationship links the general concept and more concrete one. Any PhD student is also a student and for the student concept a more general concept would be person. In this example the concept PhD student is what we call class and all instances of PhD students are real world persons.

In addition to providing a shared vocabulary and describing formally a domain, the power of ontologies lies in enabling the reasoning mechanism [45]. The formal semantics of ontologies provide basis for deriving logical conclusions – also called inferencing mechanism.

[4]http://www.w3.org/TR/rdf-sparql-query/ (Last checked April 2013)
[5]http://en.wikipedia.org/wiki/Representational_state_transfer (Last checked April 2013)
[6]In this thesis, ontologies are used as schema + instances.

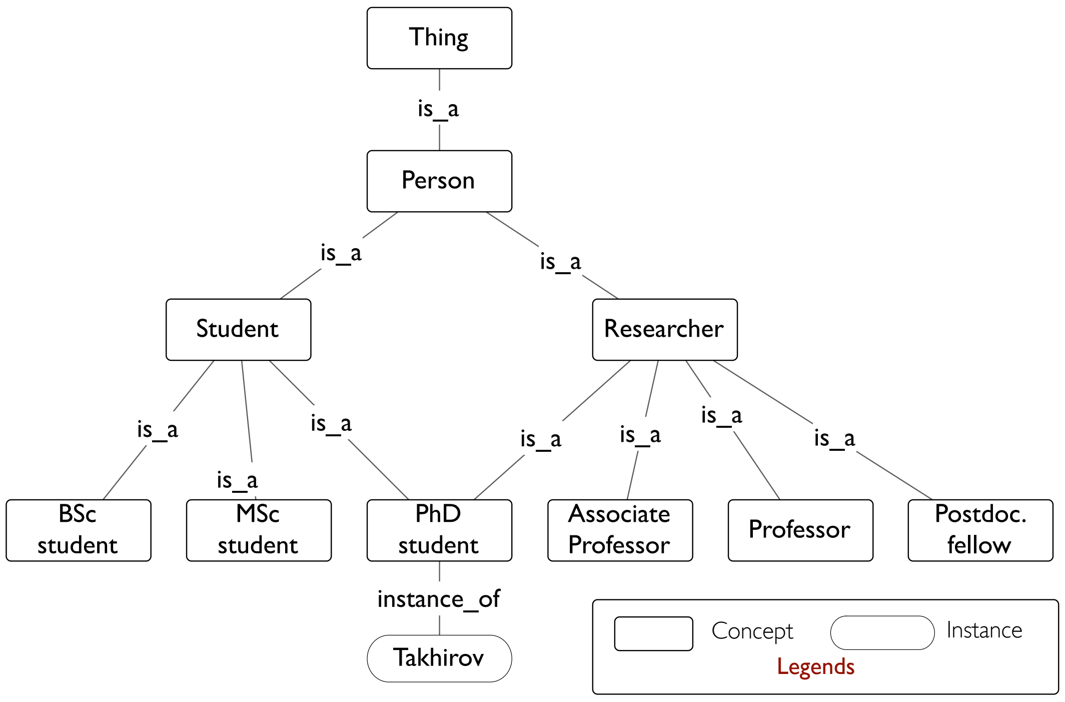


Figure 2.2: An Example of an is-a Hierarchy.

2.2.5 Entities

A common understanding of knowledge bases on the Semantic Web is that they utilize the flexible method RDF provides for decomposing knowledge into smaller pieces, called triples. This kind of organizing knowledge is supported through a directed labeled graph. Using the graph terminology, the nodes (subject and object) represent *entities* and the edges (predicate/property) that connect them is a fact or a relationships between those entities. In general, entities are often associated with the sense of *being* or *existing* in a specific domain and therefore there must be a mechanism that enables their unique identification. An entity can essentially be seen as an abstraction from the complexities of a domain. Abstraction is a common way of dealing with complexity and it has been widely used in computer science to reduce the complexity. It provides the ability to hide details of properties of an entity type.

In data modeling, abstraction is used to impose categorical structure to the data [128]. For example, through the use of abstraction we can define the generic concept *car* for a set of vehicles of various models (Mercedes, BMW, Audi, etc.). In the database domain, the abstraction is used in two ways:

- **Generalization** provides a generic type to a set of entities. Generalization is often distinguished from *classification* in which instances are assigned to an entity type. For example, in Figure 2.2 *Takhirov* is assigned to a class *PhD student* and it is an example of generalization by classification. On the other hand, type level generalization is similar to *is_a* hierarchy often used in AI (e.g. Person is generalization of Student). The benefit of generalization is that it provides a common view over individual entities by leveraging similarities and abstracting away their differences [191].

- **Aggregation** is the abstraction by which an entity is constructed based on its characteristics. For example, name, gender, address and age are characteristics of a person and thus serve as constituent properties. Aggregation implies that the object to be aggregated has its constituent properties as integral part of its structure. Often aggregated properties are inherited by specialization. The name, age and address of a person are also characteristics of a student. The concept of aggregation is similar to the concept *part_of* of AI in which properties are also seen as objects and hence any *part_of* expresses an object being an aggregate of another object (name is *part_of* person).

The RDF Schema (RDFS) has a similar mechanism to define abstractions. Specifically, RDFS specification includes the definition of the two important classes *rdfs:Resource* and *rdfs:Class*. This definition reflects the distinction between instances and types, as well as properties to define type hierarchies (*rdfs:subClassOf* and *rdfs:subPropertyOf*).

In ontologies, entities are represented by individual instances of specific types of objects. For example, "J.R.R Tolkien" represents an instance of a writer thereby may have an *instanceOf* relation to the concept "writer" in the knowledge base. In triple based RDF ontologies, the *subject* typically is an entity in most cases when considering instances of specific types.

Every entity in the knowledge base must have a unique identifier (the subject in a triple). This is required to avoid ambiguity. Many real-world entities may have multiple names by which they are referred to. For example, "Samuel Clemens" is the real name of "Mark Twain", "USA" is the same entity as the "United States of America", "Tolkien" may be used as shorthand version of "J.R.R. Tolkien". As human beings, we can understand that the mentions of these entities are interchangeable, but in order for machines to operate with the same level of understanding, these alternative labels must be mapped to the same entity. Identifiers for entities therefore serve this purpose. In the world of knowledge bases this is called a *canonical entity* which means that all labels are mapped through a domain relation (e.g. *means* or *sameAs* relation).

2.2.6 Attributes

Entities are identified with identifiers in knowledge bases, but without additional information it is not easy to learn more about them. Attributes of entities, also called properties[7], usually reflect a set of values with some distinguishing characteristics like an ISBN number and an author for books that makes entities differentiable from each other. The constituent properties of an entity discussed above, are what many would consider as attributes. Attributes and their values usually correspond to how we interpret real-world entities and their properties, and therefore description of each entity typically contains several attributes.

Some attributes may only allow a set of homogeneous values. For example, an *age* attribute of a person entity may only contain unsigned integers, a *name* attribute will only consist of alphabetic strings. An additional constraint may be set on the values of the attributes to restrict them from accepting diverse types of values (e.g. an *age* may only be between 0 and 150). This homogeneous sets are usually referred to as *domains* in data modeling [191].

In RDFS, there is a comparable notion of *rdfs:domain* and *rdfs:range* to define properties and specify details of the properties when the definition of those properties are given. Domain of a property specifies to which type this property belongs to, while range is used to restrict values that can be accepted.

Support for attributes is an essential feature of object-oriented paradigm as well. In general, all attributes of a general type are also inherited by objects, which are specialization of a general with some exceptions by explicitly disallowing inheritance. If a student is a specialization of a person, then it will inherit all properties of person (e.g. student inheriting age property of person).

[7]In this work, the terms properties and attributes are used interchangeable where it applies.

2.2.7 Relationships

The term *relationship* denotes an association between things, or entities. In open domain knowledge bases relationships are expressed with free-text as we have seen in the examples earlier. Ontological definitions impose a so-called *type signature* where a particular type of relation may only be hold between two specific types of entities, i.e.:

```
isMotherOf(Woman,Person)
```

This example signifies that the relation *isMotherOf* only exists between a *Woman* and a *Person* entities. Using the turtle syntax [20], a refined version of this example can be exemplified as shown in Figure 2.3:

```
@PREFIX ex: <http://example.org/#>

ex:Person rdf:type rdfs:Class .
ex:Woman rdf:type rdfs:Class .
ex:Woman rdfs:subClassOf ex:Person .
ex:isMotherOf rdf:type rdf:Property .
ex:isMotherOf rdfs:domain ex:Woman .
ex:isMotherOf rdfs:range ex:Person .
ex:Hillary_Clinton rdf:type ex:Woman .
ex:Hillary_Clinton ex:isMotherOf ex:Chelsea_Clinton .
```

Figure 2.3: An Example of Specifying Domain and Range in Turtle Format.

In a triple-based modeling, an important distinction between relationship and predicate is often made and that is the latter is used with a more general meaning. A triple consists of subject, predicate and object (or literal) and the predicate in most cases will define the relation if subject and object are instances of entities.

The basic relationship support in the Semantic Web languages, such as RDF and OWL, is achieved through properties being binary associations, namely a relationship is expressed between two entities. For example:

```
Barack_Obama awarded Nobel_Peace_Prize
```

However, some relationships may additionally require attributes in order to be properly expressed. In many cases it is useful to additionally include temporal attribute to the relationship itself. In the example above, since the Nobel peace prize is awarded annually, it is useful to express what year the award was handed to Obama, which is clearly a part of the relationship *awarded*. In the Semantic Web world this kind of relationship is known as *n-ary* relationship[8]. Since RDF only includes support for binary relationships, n-ary relationships can be expressed in the following ways:

- A new entity type for a relation which includes the required attributes:

  ```
  Barack_Obama  awarded   Nobel_Peace_Prize2009
  Nobel_Peace_Prize isPrize Nobel_Peace_Prize2009
  2009  year   Nobel__Prize2009
  ```

  The advantage of this approach is readability but there is a drawback of limitation on the attributes and this method cannot include additional attributes to the relationship.

- Another approach is to use identifiers for the primary entities of an n-ary relationship. For the example above, the primary entities are Obama and Nobel peace prize:

  ```
  fact_#1: Barack_Obama  awarded   Nobel_Peace_Prize
  fact_#2: fact_#1 inYear 2009
  ```

  This way of expressing n-ary relationships allows any number of additional refinement of the relationship between the primary entities. It is a modeling approach adopted in the Yago [180].

Ontology-based knowledge bases typically have upper limit on the number of relationship types (constraints on the types) because of the predefined set of types of relationships, while open domain does not have such limitation. The former is often described as precision-oriented while the latter is recall oriented.

2.2.8 Linked Open Data

Connecting related data across the data sources is the most fundamental idea of Linked Open Data (LOD). The interrelated collection of datasets available on the Web where the data is linked is called LOD cloud [28]. Over the last few years, LOD has

[8] http://www.w3.org/TR/swbp-n-aryRelations/ (Last checked April 2013)

grown from a rather specific and isolated field to a domain of widespread interest. LOD is based on Semantic Web technologies, and can be seen as means of supporting the vision of Semantic Web – a vision in which all data is globally accessible and interconnected, thus making the data reusable, sharable and more valuable. From 3 billion triples in 2007 the LOD cloud has grown to over 31 billion triples in just 4 years, which clearly shows a significant increase in the adoption.

The basic idea is to obtain data that is defined and linked such that this data can be used in semantic aware services by enabling more effective discovery, integration, and reuse across various applications. The LOD cloud[9] refers to interconnected data sources, such as DBpedia [12, 142], Freebase [32, 33] or OpenCyc [121, 133], which can be seen as the foundation of the LOD vision. LOD is based on the standards and formats that have been developed for the Semantic Web. There are set of guiding principles used for publishing and connecting/relating data on the Web [28]. These principles are formulated as follows:

- Use Uniform Resource Identifiers (URI) as names for things; everything in the LOD universe must be identifiable and the use of URI is a straightforward way to represent this information.
- Use HyperText Transfer Protocols (HTTP) URIs so that people can look up those names; this principle enables others interested in the data to reuse with already established, powerful and evolving set of standards.
- Provide useful information when someone looks up a URI using the standards such as RDF and SPARQL.
- Include links to other URIs (within this information) so that they can discover more things; the true power of Linked Data is achieved when things are linked in the LOD cloud. A lack of links between entities is considered by many to be a bad practice of Linked Data.

The above principles have been described as low-barrier and this level of simplicity is reflected in the rapid adoption of the Linked Data. The participation in the LOD movement does not actually require any formal registration or specific precondition to be part of the project. A short extract from the LOD cloud diagram is shown in the Figure 2.4. Each circle is a knowledge base or dataset that is made available as part of Linked Data. The different colors define the nature of the dataset (e.g. government,

[9] http://linkeddata.org (Last checked January2013)

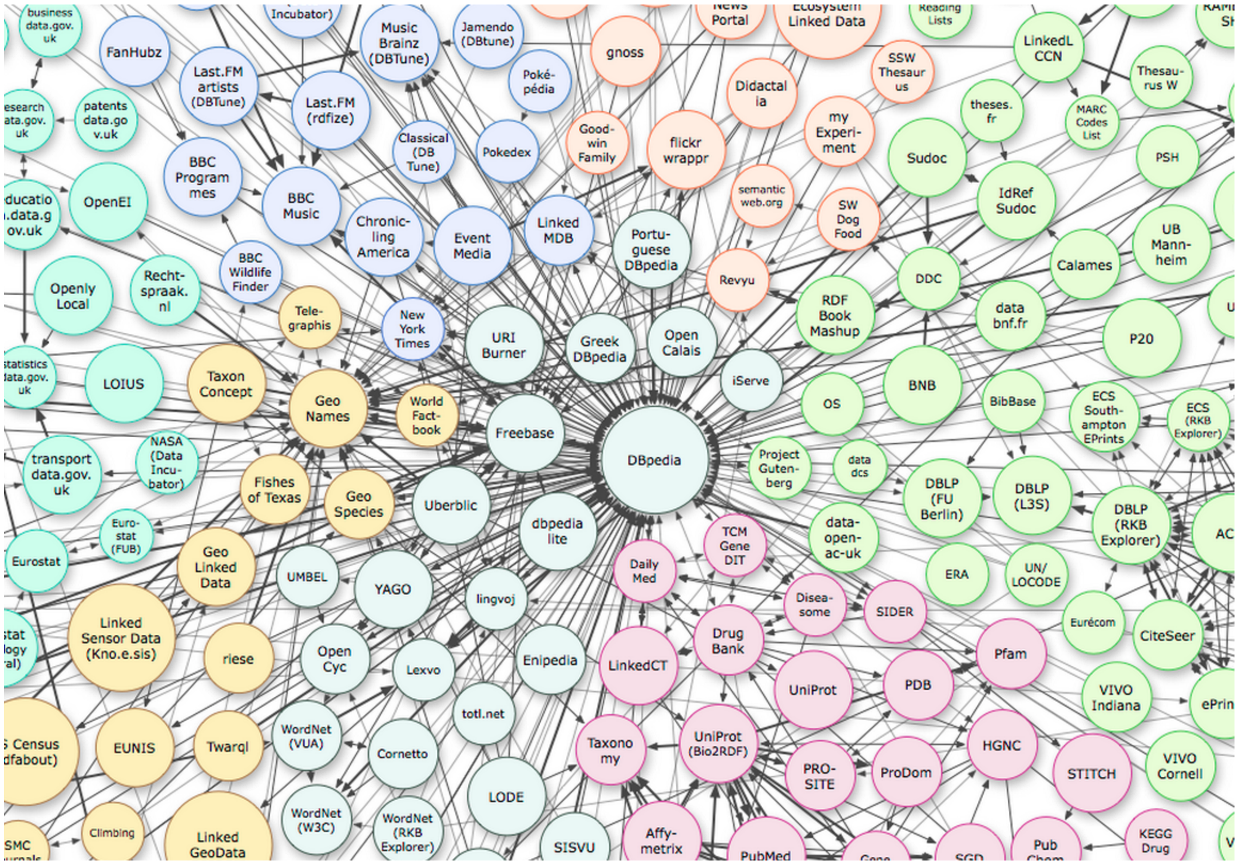


Figure 2.4: An Excerpt of the LOD Cloud Diagram as of September 19, 2011.

life science, geographic, user-generated content etc.). The links between the datasets are shown with arrows. To some extent, the degree of *importance* of a knowledge base is illustrated with the number of referring connections. For instance, DBpedia, Freebase and Geonames are examples of knowledge bases with many other datasets linking to them.

Linking one dataset to another entails creating relations (such as *owl:sameAs*) between the datasets. These links are essentially established between the entities in the different datasets. The motivations behind this practice are many and currently such relations can be used for discovery of complementary information, semantically enriching entities and verification.

A set of information and pragmatic metrics have been proposed by the Web inventor Berners-Lee, called a 5-star deployment scheme[10]. The stars are "awarded" for open

[10]http://5stardata.info/ (Last checked March 2013)

data and it follows the path:

- At the lowest level, 1-star is given to data providers to make their data available online under open license;
- Making the data structured (e.g. CSV, XML) in addition provides one more star to the data;
- The 3 star rating is achieved when a non-proprietary format is used;
- Using URIs to identify entities is awarded 4 stars;
- And finally, making the data discoverable by providing links to other datasets provides a rewarding and highest 5 stars.

As can be seen, providing relationship information is important and by providing links to other entities within and across knowledge bases makes the data discoverable and has the high potential of increasing the value of data. Relationship description or expression is predominantly achieved using *links*. A link may not only describe relationship in a local context such as a local collection, but also may refer to external collections or data sources and this capability is enabled by the use of (dereferencable) URIs.

2.3 Knowledge Extraction

Knowledge Extraction (KE) is the subfield of Information Extraction (IE) which is concerned with the extraction of knowledge from various structured and unstructured sources. One of the factors that enabled the relatively quick inception and success of LOD is the development of automated methods of extracting knowledge from diverse sources.

Availability of online sources of information in the form of natural language and semi-structured and structured texts has led to the interest in technological framework for automatically processing this text to transform it into a task-relevant information [11]. In response to the opportunities presented by online text, DARPA[11] initiated and supported the Message Understanding Conference (MUC) in the late 1980s

[11]Defense Advanced Research Projects Agency - a US Department of Defense agency responsible for the development of technologies for use by the US military, http://www.darpa.mil/ (Last checked April 2013)

by focusing on certain types of extraction tasks [92]. This is considered by many to be the start of the IE field. The benefits of MUCs were that a corpus of training texts were provided along with the specification of the actual IE task. An important feature was that the ground-truth was carefully produced by humans as well as the specification of the output format of each specific task. Therefore, an evaluation could be performed automatically. Participating parties used the training data for training their systems and at the conference itself they would run their system against the unseen data. MUCs provided a platform for discussing and advancing the state of the art in IE and developing IE tools.

With IE tools in place and the availability of large amount of online information, KE emerged as a subfield of IE. The differentiating feature of KE is that it is traditionally backed by a semantic schema (like an ontology) and hence the extracted knowledge is machine-processable and is represented in a manner that enables inferencing. KE methods have been used for variety of tasks and many projects have been directed towards extracting knowledge from relational databases, XML and CSV. Regardless of the source of data, the KE systems commonly perform extraction in a form of triples. The main sources of data to be used by a KE method for knowledge bases are discussed below.

There are two types of paradigms of knowledge extraction:

- **Ontology-based** in which ontology is used as vocabulary and defines the types of concepts used in the knowledge base;
- **Open domain** knowledge extraction systems do not have a vocabulary and the relationship types in the knowledge base are not pre-specified.

2.3.1 Ontology-based Knowledge Extraction

In ontology-based KE, ontologies are applied as a vocabulary of entities and relationships [137, 199]. Hence, there is a certain number of types of entities and relationships included in the knowledge base. During extraction only the candidate relations included in the vocabulary will be considered. Extraction of assertions, although the number of assertions is not high, achieves a high precision. This feature is achieved thanks to the use of ontologies which effectively discards incorrect assertions.

Most of the ontology-based KE approaches use a single ontology. A meta knowledge base may be used to aggregate or integrate several knowledge bases, thereby use multiple different ontologies. In this case, with more than one ontology the domain is expanded and the extraction of many more assertions is made possible.

Using ontologies for constructing knowledge bases is becoming increasingly popular. On the one hand this is due to the Linked Data movement. The prevalent projects of this kind are DBpedia, Freebase, Yago. On the other hand, there is a huge commercial interest that has been growing rapidly lately. Projects such as Google Knowledge Graph and EntityCube, Evi all demonstrate how knowledge bases can be effectively exploited for a variety of purposes.

These ontology-enhanced knowledge bases contain millions and millions of canonical entities and relational facts about these entities. Examples of such facts are:

```
The_Lord_of_the_Rings  type   wordnet_novel
Barack_Obama  hasWonPrize   Nobel_Peace_Prize
USA  sameAs   United_States_of_America
politician  subclassOf   person
Jens_Stoltenberg  bornIn  Oslo
```

2.3.2 Open Domain Knowledge Extraction

In open domain knowledge bases, the types of entities and relations are not specified. In a sense, this feature of open domain knowledge bases is often said to contain phrase-like relations. There is no dictionary for entities; the extraction of entities are driven by *nouns* and therefore every noun is seen as a possible candidate entity. Every verbal phrase between the candidate entities therefore represents a *pattern for relations*. Open domain knowledge bases are populated with the open information extraction paradigm [71]. This feature enables generation of a potentially large number of assertions, and is considered by advocates of this paradigm as a way to address the coverage limitation of ontology-driven KE. Examples of statements from open domain knowledge bases are:

```
"Vitamin D"  "assists with"   "calcium absorption"
"Folic acid"  "also plays an important role in"   "hair loss"
"Iraq war"  "has risen above"   "500 billion dollars"
"Benzoyl peroxide"  "helps kill"   "skin bacteria"
```

2.3.3 Data Sources for Knowledge Extraction

Many will argue that the amount of semantic data is the key in realizing many of the visions of the Semantic Web at large-scale. The challenge is the knowledge acquisition bottleneck that is met by the traditional "top-down" model of designing ontology. To address this issue, automated methods have been developed to use wide range of data sources. In general, these sources can be divided into the following categories:

- Structured;
- Semi-structured;
- Unstructured.

IE methods help with obtaining structured data from unstructured sources expressed in natural language [17, 54]. Thus this domain is inherently related to the field of natural language processing. The main tasks concerned are the identification and extraction of instances of a particular types of entities and relations from a natural language text and their transformation into a structured representation such as database or ontologies [91]. While IR systems focus on retrieval of relevant documents from the collection, an IE system strives to retrieve or extract relevant information (instances of entities and relations) from a document.

Structured Data Sources

The advantage of exposing existing structured data using a shared vocabulary is that it enables its applicability in a broader semantic context. There is a vast amount of structured data – also called relational data – (e.g. stored in relational databases) that can be used to extract entities to be integrated into a knowledge base. The need to make this structured data reusable as semantic data has recently increased and therefore it has become widely accepted practice to publish existing structured data and expose the published collection as knowledge bases. Consequently, much research has been focused on mapping this relational data to RDF. However, often solutions are domain specific with tools that work in a particular environment [174].

To address this issue, the World Wide Web Consortium (W3C) initiated a working group RDB2RDF to study and propose standardized languages for mapping relational data into RDF and OWL[12]. More specifically, this working group recently came up with an interesting recommendation for "a direct mapping of relational data to RDF"[13]. An example of such direct mapping is given in Table 2.3.

ID	Name	AffiliationState
3	Abraham Lincoln	25
4	Thomas Jefferson	35
5	Barack H. Obama	25

Table 2.1: Table President.

ID	Name
25	Illinois
30	New York
35	Virginia

Table 2.2: Table State.

Table 2.3: A Simplified Relational Data Describing Presidents and States of Primary Affiliation by Presidents. The Column AffiliationState of the Table President References the Primary Key Column ID of the State Table.

The table *President* contains a list of American presidents with three column definitions: the primary key column *ID*, the *Name* and the column *AffiliationState*. Since the states of primary affiliation may be common to several presidents, this information is stored in a separate table and the column *AffiliationState* references the primary key column *ID* of the table *State*. Given these two table definitions, a direct mapping to RDF would typically take the following form:

```
<President/ID=3>    <rdf:type>    <President>
<President/ID=4>    <rdf:type>     <President>
<President/ID=3>    <President#Name>    "Abraham Lincoln"
<President/ID=4>    <President#Name>    "Thomas Jefferson"
<President/ID=5>    <President#Name>    "Barack H. Obama"
<President/ID=3>    <President#AffiliationState>    <State/ID=25>
<President/ID=4>    <President#AffiliationState>    <State/ID=35>
<President/ID=5>    <President#AffiliationState>    <State/ID=25>
<State/ID=25>       <State#Name>        "Illinois"
<State/ID=30>       <State#Name>        "New York"
<State/ID=35>       <State#Name>        "Virginia"
```

This type of transformation of representing relational data as RDF graph is common and many databases have been published as linked data in this manner. However,

[12]http://www.w3.org/2001/sw/rdb2rdf/ (Last checked May 2013)
[13]http://www.w3.org/TR/rdb-direct-mapping/ (Last checked May 2013)

this is merely a syntactic approach and the semantics of this data are not always necessarily reusable in the context Semantic Web which is a prerequisite for a typical usage. Simply converting into more Semantic Web friendly format such as RDF does not automatically enable semantic-aware services, since converted RDF data needs to be reinterpreted as well as transformed.

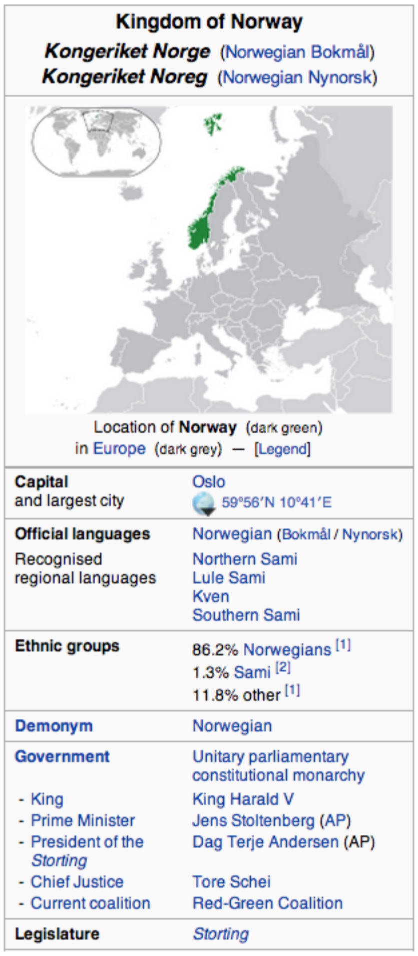
Recently, the initiatives such as schema.org, microformats[14] and GoodRelations[15] have boosted the amount of structured markup data on the Web. Several studies have shown that over 30% of all Web pages in existence[16] now contain structured data [145, 151] and this data can be directly extracted as RDF graphs (e.g. using *Anything To Triples*[17]).

Unstructured Data Sources

Much of human knowledge is expressed in free-text stored in natural language documents. Thankfully, the Web has enabled the sharing of the digital information with minimal effort and the progress in IE enables the analysis and extraction of useful, valuable information from the text, thereby turning it into a machine processable knowledge. Using unstructured text as a source to derive structured data for a knowledge base is still an open area of active research.

Exploiting unstructured text as a source of relational facts often relies on Natural Language Processing (NLP) techniques. A substantial amount of work in NLP is directed towards the development of parsers. These parsers attempt to capture the meaning of sentences with the goal of building a tree or directed graph as output which reflects the various levels of linguistic information including the part-of-speech (POS), the presence of phrases, grammatical structures and semantic roles. The output produced by a parser is the structure and annotations which are useful for determining relationships between entities in a sentence. In this context, one of the major challenges associated with using unstructured data sources is the task of Named Entity Recognition (NER). Traditionally, NER techniques are focused around the detection of common entities such as people, organization or geographic location. However, the recognition of domain specific entities poses a particular challenge because the NER tools usually require training examples for the types of entities to be recognized.

[14]http://microformats.org (Last checked January 2013)
[15]http://www.heppnetz.de/projects/goodrelations/ (Last checked December 2012)
[16]These statistics take into account Web pages up until February 2012
[17]http://any23.apache.org/ (Last checked May 2013)

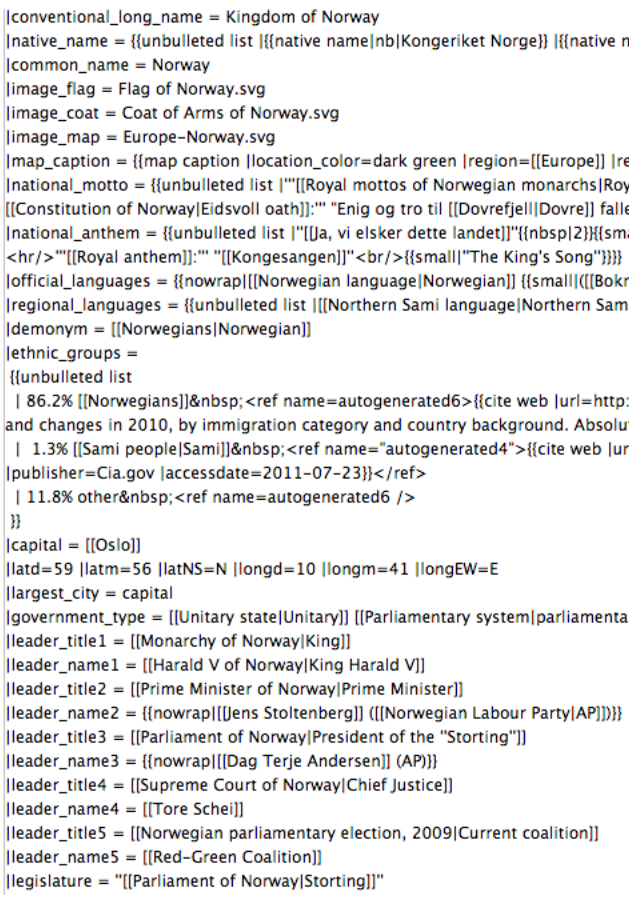


Kingdom of Norway
Kongeriket Norge (Norwegian Bokmål)
Kongeriket Noreg (Norwegian Nynorsk)

Location of **Norway** (dark green)
in Europe (dark grey) — [Legend]

Capital and largest city	Oslo 59°56′N 10°41′E
Official languages	Norwegian (Bokmål / Nynorsk)
Recognised regional languages	Northern Sami Lule Sami Kven Southern Sami
Ethnic groups	86.2% Norwegians [1] 1.3% Sami [2] 11.8% other [1]
Demonym	Norwegian
Government	Unitary parliamentary constitutional monarchy
- King	King Harald V
- Prime Minister	Jens Stoltenberg (AP)
- President of the *Storting*	Dag Terje Andersen (AP)
- Chief Justice	Tore Schei
- Current coalition	Red-Green Coalition
Legislature	*Storting*

(a) Wikipedia Infobox for Article about Norway



```
|conventional_long_name = Kingdom of Norway
|native_name = {{unbulleted list |{{native name|nb|Kongeriket Norge}} |{{native n
|common_name = Norway
|image_flag = Flag of Norway.svg
|image_coat = Coat of Arms of Norway.svg
|image_map = Europe-Norway.svg
|map_caption = {{map caption |location_color=dark green |region=[[Europe]] |re
|national_motto = {{unbulleted list |'''[[Royal mottos of Norwegian monarchs|Roy
[[Constitution of Norway|Eidsvoll oath]]:''' "Enig og tro til [[Dovrefjell|Dovre]] falle
|national_anthem = {{unbulleted list |''[[Ja, vi elsker dette landet]]''{{nbsp|2}}{{sma
<hr/>'''[[Royal anthem]]:''' ''[[Kongesangen]]''<br/>{{small|''The King's Song''}}}}
|official_languages = {{nowrap|[[Norwegian language|Norwegian]] {{small|([[Bokr
|regional_languages = {{unbulleted list |[[Northern Sami language|Northern Sam
|demonym = [[Norwegians|Norwegian]]
|ethnic_groups =
 {{unbulleted list
  | 86.2% [[Norwegians]]&nbsp;<ref name=autogenerated6>{{cite web |url=http:
and changes in 2010, by immigration category and country background. Absolu
  |  1.3% [[Sami people|Sami]]&nbsp;<ref name="autogenerated4">{{cite web |ur
|publisher=Cia.gov |accessdate=2011-07-23}}</ref>
  | 11.8% other&nbsp;<ref name=autogenerated6 />
 }}
|capital = [[Oslo]]
|latd=59 |latm=56 |latNS=N |longd=10 |longm=41 |longEW=E
|largest_city = capital
|government_type = [[Unitary state|Unitary]] [[Parliamentary system|parliamenta
|leader_title1 = [[Monarchy of Norway|King]]
|leader_name1 = [[Harald V of Norway|King Harald V]]
|leader_title2 = [[Prime Minister of Norway|Prime Minister]]
|leader_name2 = {{nowrap|[[Jens Stoltenberg]] ([[Norwegian Labour Party|AP]])}}
|leader_title3 = [[Parliament of Norway|President of the ''Storting'']]
|leader_name3 = {{nowrap|[[Dag Terje Andersen]] (AP)}}
|leader_title4 = [[Supreme Court of Norway|Chief Justice]]
|leader_name4 = [[Tore Schei]]
|leader_title5 = [[Norwegian parliamentary election, 2009|Current coalition]]
|leader_name5 = [[Red-Green Coalition]]
|legislature = ''[[Parliament of Norway|Storting]]''
```

(b) Corresponding Source in Edit Mode

Figure 2.5: An Example of an Infobox.

Semi-structured Data Sources

Semi-structured data is another popular source of extracting semantic information for knowledge bases. Semi-structured data is often characterized by the being a *middleman* – neither raw unstructured nor typed structured data. Probably the most well-known source of this kind is Wikipedia infoboxes as shown in the Figure 2.5. A number of projects use infoboxes as a source of extraction of factual information. For such semi-structured data source as Wikipedia infobox, a solution to extract the information can be as straightforward as using a pattern-matching technique. In fact, prevalent projects (e.g. DBpedia, Yago) specifically make use of this method.

2.4 Metadata as Knowledge

Metadata is considered by many to be one of the primary ingredients in the science of information organization and management. The term is often associated with cultural institutions, and in particular with libraries, as they have for a long time used metadata to bring the order in their collections and employ metadata as a surrogate for books on the shelves.

The bibliographic metadata managed by libraries is recognized as important global documentation of the intellectual and artistic endeavor of mankind that could be reused and integrated with other sources in advanced new services that enable users to learn about, discover, annotate and discuss our cultural heritage [10]. Cultural institutions have for decades created metadata records describing various cultural items expressed as text, music or other forms.

The use of this metadata has traditionally been limited to library services such as a simple bibliographic search, but increasingly there are activities towards exploiting this resource in innovative ways, beyond the library domain [65, 114]. It is well recognized that in the future, in order to enable the global interoperability of bibliographic metadata, cultural institutions should participate in a shared standardization effort focused around Semantic Web standards. However, this is a complex issue because the huge amount of bibliographic metadata is created using different standards and ad-hoc schemas. Syntactic solutions, which are achieved by transforming to formats that are compatible with the tools and services used for semantic aware services, have been proposed recently, but additionally, there is a need to properly integrate this metadata (using various other complementary sources) in the LOD.

The notion of *properly integrating* metadata means not only representing the knowledge contained in the metadata using semantic technologies, but it also means increase the value of resulting collection by improving the transformation results. For example, results can be improved by removing ambiguities and completing missing information using other data sources.

2.5 Metrics

This section provides a brief description of the metrics used for evaluation in terms of quality and similarity metrics.

2.5.1 Quality Measures

In order to adapt the IR metrics to our tasks, it has to be adapted in a sensible way. For example, for a fact extraction task, it would typically operate on statement or fact level, which has been extracted. The metrics would then consider facts as documents while result set would be the collection of extracted facts. For the matching task, we operate on correspondences rather than documents or facts. To be generic, correspondences and facts are referred to as *elements* in various tasks performed during the course of this thesis. A ground truth is required to be constructed which includes a set of elements where each element is labeled as correct or incorrect. The metrics are based on the contingency values shown in Table 2.4.

	Correct elements	**Incorrect elements**
Computed elements judged as correct	True Positive (TP)	False Positive (FP)
Computed elements judged as incorrect	True Negative (TN)	False Negative (FN)

Table 2.4: Contingency Table.

Based on the table data recall is calculated as:

$$recall = \frac{TP}{TP + FN}$$

which basically indicates the number of correct elements that have been discovered out of the total number of elements. The precision on the other hand is number of correct discovered elements divided by the total number of discovered elements:

$$precision = \frac{TP}{TP + FP}$$

In many applications, both recall and precision are important and therefore an F-measure is used as a balanced measure between precision and recall:

$$F - measure_{\beta} = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall}$$

The β parameter in the above equation is often used as an indicator of the importance of precision or recall. The default value is 1 which equally weights both recall and precision. The higher value the β parameter takes, the more weight is given to recall, while lower values would more emphasize precision.

2.5.2 Similarity Metrics

Traditionally, the matching task is performed by comparing the various properties of the candidate pairs of correspondences in two different data sources, such as *label*, *data type*, *categorical information*, etc. String-based similarity methods take advantage of the typical characteristics of a string, such as sequence of letters.

The output of similarity metrics is a numeric value that indicates the level of similarity between comparing candidates. Usually, a similarity metric produces a value between 0 and 1. High numeric value is generated as an indication of highly similar values, i.e. the value 1 is exact match. Low value means less similar values, i.e. the value 0 means totally dissimilar values.

There exist many similarity metrics and although they are main components in most entity matching approaches, the goal is not to describe all of them. Therefore, the most important categories are briefly presented below.

- **Structural** similarity metrics exploit the structural properties of elements instead of their attributes (labels, identifiers). This kind of analysis can be performed both internally and externally. The methods based on using internal structure of elements take into account the properties of elements including their attributes, relationships, datatypes and various constraints. Internal structures are useful to reduce the number of candidate pairs. The external structures on the other hand, study the external characteristics of an entity. The intuition behind the external method is that if two elements from different data sources are similar, then their neighboring elements might too be similar. Using external structure method is advantageous if two schemas have conceptually very similar structure, but it is not well suited to matching schemas created with different design perspectives.

- **Instance-based** similarity metrics perform comparison on instance level rather than the class (entity type). Instance-based methods are further divided into two types: (i) instances of a common class and (ii) instances of difference classes. The common class instance comparisons are performed using well-known metrics, such as Jaccard similarity metric [50]. The Haussdorff distance[18] is an example of a metric that is often used for computing the distance between two sets via a disimilarity function for the classes that do not share the same set of instances.

- **Model-based** metrics tend to have model-theoretic semantics which is used to support the matching results. Description logic based techniques as well as propositional satisfiability (SAT) techniques represent a model-based method. Description logics techniques, i.e. subsumption test, are often used to create the relationships between classes. In the SAT based techniques, the matching problem is typically translated into a propositional formula. The formula is then dispatched to a satisfiability solver and, if a satisfying model is found, it can be used as a example to show that the system satisfies the required property.

- **Terminological** methods mainly deal with comparing string. In this work, we use the labels, categorical information and other attributes of entities to compute terminological similarity. In order to compute the similarities which are character string based, the properties used for the comparison need to be normalized. Some examples of normalizations are tokenization (phrase splitting), character replacement (numbers, blank spaces, special characters, e.g. exclamation mark, colon, etc), label expansion.

Each similarity metric has its advantages due to the nature of task it tries to tackle and many also include disadvantages. Since, we consider terminological similarity metrics in various parts of this thesis, we will be focusing on describing some of the metrics used in this work: Scaled Levenshtein Jaro Winkler, and Monge Elkan.

The Levenshtein distance is cost-based edit-distance like metric. It assigns a unit cost to all edit operations (single character edits such as insertion, deletion and substitution) between two strings. The distance is zero if the comparing strings are equal. For example the Levenshtein distance between the strings “Sofia” and “Sophie” is 3:

- substitute “p” with “f”
- delete “h”
- substitute “e” with “a”.

[18]urlhttp://en.wikipedia.org/wiki/Hausdorff_distance (Last checked March 2013)

The value is usually normalized to be between [0,1] hence a the normalized version of Levenshtein distance is called scaled Levenshtein.

Jaro Winkler distance is a measure widely used in the database domain for the record linkage and various other data integration tasks. It is mainly designed for short strings such as person's first and last names and it is based on the number and the order of common characters in two strings. The first part of this distance is based on Jaro metric. Given two strings s_1 and s_2, the Jaro metric is computed as:

$$jaro(s_1, s_2) = \frac{1}{3} \times \left(\frac{|s'_1|}{|s_1|} + \frac{|s'_2|}{|s_2|} + \frac{|s'_1| - T_{s1,s_2}}{|s'_1|} \right)$$

where $|s'_1|$ is the number of common characters in s_1, $|s'_2|$ is the number of common characters in s_2 and T_{s1,s_2} is half the number of transpositions (permutation to change characters). Winkler further developed the metric to include the common prefix between strings:

$$winkler(s_1, s_2) = jaro(s_1, s_2) + \left(cp \times (1 - jaro(s_1, s_2))\right)$$

where c is the length of the common prefix (usually up to 4 characters maximum) and p is the scaling factor with the default value of 0.1. As an example consider strings s_1="Sofia" and s_2="Sophie", the Jaro-Winkler distance is:

$|s_1| = 5$
$|s_2| = 6$
$T_{s_1,s_2} = 0$

$$jaro(\text{“}Sofia\text{''}, \text{“}Sophie\text{''}) = \frac{1}{3} \times \left(\frac{3}{5} + \frac{3}{6} + \frac{3-0}{3} \right) = 0.7$$

The Jaro-Winkler score is therefore:

$$winkler(\text{“}Sofia\text{''}, \text{“}Sophie\text{''}) = 0.7 + (2 \times 0.1 \times (1 - 0.7)) = 0.76$$

As can be seen, the Jaro-Winkler metric slightly improved the Jaro score because of the common prefix.

Another metric that is used is Monge Elkan [150]. This metric is an affine version of the previously proposed Smith-Waterman function. Given, two strings s_1 and s_2, this metric is calculated as:

$$monge(s_1,s_2) = \frac{1}{|s_1|}\sum_{i=1}^{|s_1|}\max_{j=1}^{|s_2|} monge(s_{1i},s_{2j})$$

The metric is recursive and produces a value between [0,1]. Although it has a quadratic time complexity $O(N^2)$, it is useful for matching various forms of abbreviations (e.g. Univ. vs University, MIT vs Massachusets Institute of Technology, Dept. vs Department etc).

Part II

State of the Art and Overview

3 Knowledge Base Population

This chapter first surveys the state of the art in knowledge extraction. Then, automatic ontology construction is briefly reviewed, followed by the Knowledge Base Population (KBP) initiative of Text Analysis Conference (TAC). Finally, entity matching and search will be presented.

Automatic construction of a knowledge base from structured and unstructured sources is an effort called knowledge base population. The basic task, given a collection, is to extract entities and relations between them and integrate those extracted entities and relations in the semantic knowledge base as shown in Figure 3.1.

One of the earliest efforts to construct a general knowledge base is the Cyc project [121]. This knowledge base is created manually by experts and contains a broad selection of common-sense knowledge. Although it is proprietary, a subset of this knowledge base has been made available under the terms of the Apache license[1].

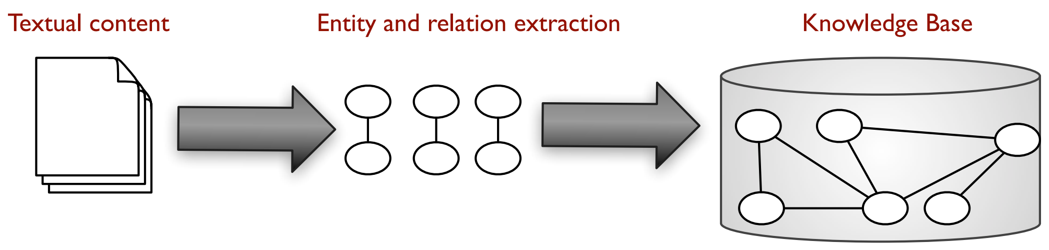


Figure 3.1: A Typical Knowledge Base Population Pipeline from Textual Content.

[1]http://www.opencyc.org (Last checked April 2013)

Another well-known manually constructed KB is WordNet. As a lexical database for the English language, many would argue that the success of WordNet has been remarkable; many other projects exploit it for the variety of purposes because of its high-quality compilation of knowledge by domain experts.

Other forms of manually built knowledge bases include various forms of systems for vocabulary control and thesauri such as OpenThesaurus[2], Library of Congress Subject Headings (LCSH)[3], Medical Subject Headings (MeSH)[4] etc.

On the one hand, a manually created knowledge base by domain experts and by other means in which humans are involved, usually is of a high quality. On the other hand, these knowledge bases are relatively smaller in size when compared with automated approaches. In this work, we focus on automated means of integrating knowledge in knowledge bases. The rest of this chapter will detail various tasks related to maintaining a knowledge base.

3.1 Automatic Ontology Construction

Automatically creating ontologies have for a long time been an attractive research area. Ontologies can be built from diverse sources [41, 130, 136, 179]: databases, structured, unstructured and semi-structured content, dictionaries and taxonomies. Any automatic construction method requires a fair amount of effort and the difficulty level is directly related to how much structure is there in the input content. For example, for dictionaries and taxonomies, which already have a structure, the main effort would be to create mappings to the concepts of the output ontology.

Wikipedia has played a central role and have been used as a backbone for mining high quality knowledge for such research projects. Knowledge bases such as DBpedia, Yago and Freebase all use Wikipedia for extracting facts for integration in their respective ontologies. These knowledge bases are one of the first initiatives to automatically extract structured content from Wikipedia relying on the infoboxes provided by the knowledge-sharing community.

Since, many companies and organizations have added their own knowledge bases to the LOD cloud, from generic ontologies such as Yago and Freebase to specialized bases

[2]http://www.openthesaurus.de (Last checked April 2013)
[3]http://id.loc.gov/authorities/subjects.html (Last checked April 2013)
[4]http://www.nlm.nih.gov/mesh/ (Last checked April 2013)

such as MusicBrainz or LinkedMDB [99]. The process for converting unstructured or semi-structured data sources into facts is called *Triplification*[5]. For instance, Triplify has been designed to extract triples from Relational databases and expose them on LOD [13] while Catriple builds a store of triples from Wikipedia categories [127]. Similarly to most of these approaches, we generate triples and store them in our knowledge base.

In the Information Retrieval (IR) domain, researchers have studied the discovery of corresponding LOD entities for a given task, such as in the TREC challenge [15]. Due to the large scale application and the uncertainty of the results, a ranking of the most probable entities which correspond to the query (usually with target categories) is computed [172, 192]. The linking to LOD for disambiguation and enrichment has also been studied for any bag of words [147]. In our context, the entities are extracted from textual documents and usually represented with a label. The surrounding context of the label in the sentence is the main information available for discovering the corresponding LOD entity.

3.2 Knowledge Extraction from Text

Knowledge extraction refers to the creation of knowledge from structured and unstructured data by transforming this data into a machine processable format. Major approaches for knowledge extraction are basically divided into three types: supervised, semi-supervised and unsupervised methods. In the supervised methods - also categorized as kernel methods[6] - knowledge extraction is considered as a classification task. From a snippet of a text the supervised system performs parsing and obtains examples with labels reflecting the relation. Labeled examples are used to create a model. This model is then applied to parsed texts to obtain the class relation label [55, 138, 202].

Early pattern-analytic methods such as semi-supervised techniques have been proposed more than a decade ago [7, 37]. The basic idea is the following. Given a handful set of examples the system is bootstrapped to discover or learn patterns for the seed examples. A seed example at this step is said to be a pair of entities,

[5]http://triplify.org/Challenge/ (Last checked April 2013)

[6]A class of algorithms for pattern analysis using high dimensional feature space, .e.g. Support Vector Machine (SVM).

e.g. *"<J.R.R. Tolkien, The Lord of the Rings>"*. Given these two entities, the pattern generation step discovers patterns from the text snippets where input entities are mentioned. In subsequent step, these patterns are used to discover new, previously unknown examples for a given relation. Both relations and entities are specified in ontology upfront prior to the extraction process. For this reason, this method is also referred to as ontology-driven or ontology-based IE [44, 152, 181, 199].

The third paradigm to IE is an unsupervised method [105], also called Open Information Extraction [58, 71]. As the name suggests, this approach to IE does not require a pre-specified vocabulary or an ontology [75], nor does it necessarily need training data or rules. Usually, this approach facilitates domain independent extraction of assertions. This paradigm is often considered to be liberal in a sense that essentially any text between two entities' mentions is considered as a relation. Obviously, this implicitly promotes the recall while accepting a level of noise in the extraction results [126].

3.2.1 Wrapper Induction Systems

The need to automatically extract data that easily fits into a relational model did not only appear in the IE community, but also the database community recognized the necessity of transforming the content into a relational form. Furthermore, the industry called for business intelligence applications that could exploit large collections of HTML documents. *Wrapper generation* field emerged as a result to address this issue. Essentially, wrapper generation systems can be regarded as a more general supervised approach, but it concentrates on Web pages and exploits the structural patterns of an HTML page. Wrapper generation systems are motivated by the fact that some Web pages do not exhibit the rich grammatical structure for NLP systems to be employed. A typical *wrapper generation* system performs extraction from an HTML page based on predefined manually hand-coded templates, called *wrappers*. Therefore, this kind of approach shares the shortcomings of the supervised approach discussed earlier. Additionally, it is in some cases considered as time consuming, and requires some maintenance work to keep the wrappers up to date. To automate the process of wrapper creation, *wrapper induction* systems have been proposed [118]. The main purpose of the wrapper induction is to generate data extraction rules from manually labeled training examples without using linguistic properties. As the name suggests, a wrapper induction approach is based on inductive learning, which is seen as a process of reasoning from a set of examples [117]. Using a set of input examples corresponding to the samples of the input-output behavior of the wrapper to be constructed, the output typically consists of classes of wrappers.

3.2.2 Knowledge Extraction Approaches

Extraction from natural language text is substantially different than extracting from metadata or using wrapper induction systems because the quality of text varies greatly due to the unstructured nature of texts. This section provides a review of existing knowledge extraction approaches.

DIPRE and Snowball

Over the recent years, various works have been proposed to discover relationships for specific domains [198]. For instance, Snowball associates companies to the cities where their headquarters are located [7] while DIPRE focuses on books and authors [37]. One of the earliest semi-supervised system, DIPRE, uses a few pairs of entities for a given type of relationship as initial seeds [37]. It searches the Web for these pairs to extract patterns representing a relationship, and use the patterns to discover new pairs of entities. These new entities are integrated in the loop to generate more patterns, and then find new pairs of entities. Snowball [7] enhances DIPRE in two different directions. First, a verification step is performed so that generated pairs of examples are checked with MITRE named entity tagger. Secondly, the patterns are more flexible because they are represented as vectors of weighted terms, thus enabling the clustering of similar patterns.

One of the main differences between Snowball and DIPRE is that the former adds a verification step. When new pairs of entities are discovered from generated patterns, they are checked by using MITRE named entity tagger. Thus they avoid too generic patterns. The other difference lies in the flexibility of the patterns. In DIPRE they are hard-represented while in Snowball the three flexible parts of the pattern (s_b, s_m, s_a) are vectors of weighted terms, thus enabling to cluster similar patterns. Each cluster of pattern is represented by a centroid pattern. There is also a mechanism to filter the pattern by numbers of tuples supporting it. The similarity between two patterns is computed by adding the cosine measures between each part of the part. A pattern's confidence is estimated by the number of positive tuples that pattern generates. A tuple will have high confidence if generated by multiple high-confidence patterns.

The evaluation is performed via an estimated precision (using a sample of 100 tuples) while experiments have generated 80000 tuples.

OAK

Hasegawa et al. have proposed an unsupervised system for extracting relations [98]. The idea is to identify types of entities in corpora using the OAK system. The basic assumption is that the pairs of entities that occur in a similar context represent the same relation. This relation is discovered through the process of context-based clustering of pairs of entities. The intuition of the system is that a context providing a basis for multiple relations is not expected and a pair of entities would either not be clustered at all or would be clustered to the most frequently expressed relation. Consequently, the approach is based on tagging named entities in the corpora, obtaining the pair of entities' co-occurrence and the context in which they are mentioned, measuring the context similarity, create clusters and finally assigning labels to the cluster of pairs of named entities.

The proposed system does not deal with the issue of pairs entities linked by multiple relationships. In addition, the experiments are restricted to a few types of entities (30 at most).

Espresso

Later experiments include the Espresso [163] system, which is a weakly-supervised relation extraction system that also makes use of generic patterns along with principled reliability measure. The principled reliability measure proposed in Espresso is a measure of pattern and instance reliability, which enables the filtering algorithm. The system is based on the Hearst patterns [100] and learns the surface patterns to extract more instances. To accomplish this task, Espresso takes seed instances for a particular relation, which is why the system is called weakly-supervised. The generic pattern used in Espresso has a broad coverage which results in both higher number of true-positives and false-positives. For example, for the relation "part-of the", the pattern *X of Y* yields both *wheel of the car* (correct) and *house of representatives* (incorrect). Therefore, the main objective addressed is to achieve a balanced recall/precision via the use of what they call *reliable patterns*. The system is presented as efficient in terms of reliability and precision. However, experiments were performed on smaller datasets and it is not known how the system performs at large-scale.

KnowItAll

The KnowItAll system has been designed for large scale datasets such as the Web [72]. This system is able to annotate its own training examples using a few generic patterns. The patterns are composed of specific-domain extraction rules to obtain examples along with a probability to be relevant based on search engine hit counts.

TextRunner brought further perspective in the field of Open Information Extraction, for which the types of relationships are not predefined [17]. A self-supervised learner is in charge of labeling the training data as positive or negative, and a classifier is then trained. Relations are extracted with the classifier while a redundancy-based probabilistic model assigns confidence scores to each new example. The system was further developed into a ReVerb framework [75], which improves the precision-recall curve of the TextRunner.

NELL

ReadTheWeb / NELL [43] is another project that aims at continuously extracting categories (e.g., the type of an entity) and relationships from web documents and improving the extraction step by means of learning techniques. Four components including a classifier and two learners are in charge of deriving the facts with a confidence score. According to the content of the online knowledge base, more iterations provide high confidence scores (almost 100%) for irrelevant relationships. In addition, NELL is mainly dedicated to the discovery of categories (95% of the discovered facts) rather than relationships between entities.

Prospera

A recent work about knowledge extraction reconciles three main issues in terms of precision, recall and performance called Prospera [153]. The Prospera system further develops the work done on the SOFIE [181] project. It utilizes both pattern analysis with n-gram item sets to ensure a good recall and rule-based reasoning to guarantee an acceptable precision. The performance aspect is handled by partitioning and parallelizing the tasks in a MapReduce-based distributed architecture. There are three major phases in Prospera: (i) the pattern gathering phase identifies a pair of entity names with the surface string appearing between those pair of names. This step additionally performs context-based mapping of names to entities in the Yago ontology; (ii) the pattern analysis phase generalizes patterns to obtain n-grams to obtain fact candidates; (iii) to obtain better precision, the final reasoning phase performs MaxSat-based reasoner considering the pre-specified constraints. A restriction of this work deals with the pattern, which only covers the middle text between the two entities. This limitation affects the recall, as shown with the example *"Lord Of The Rings, which Tolkien has written"*.

GATE (ANNIE)

The GATE project[7] (General Architecture for Text Engineering) offers a set of tools for processing text [56]. In addition to being an architecture, it has a development environment, and a framework for building systems for human language processing. As a mature and open source project, GATE has attracted a large audience of both users and developers. In particular, it includes an IE system coined ANNIE (A Nearly-New Information Extraction System) [134, 135] with a range of modules including a tokenizer, a gazetteer, a sentence splitter, a POS tagger, a named entities transducer and a coreference tagger. One of the differentiating feature of ANNIE is that it has a support for multilingual processing through the use of Unicode for text display in the GUI and accepting input in various languages. ANNIE has been reported to achieve a precision and a recall value of around 90% for the NER task [134].

Probase

Probase [200] is a recent Microsoft Research project that is based on a probabilistic model and it focuses on universal text understanding. It views the world in a concept-centric way and the goal is to construct a fine-grained taxonomy of concepts. Probase only considers an *isA* relation and therefore targets a hierarchical tree of concepts like other type hierarchies. This framework is probabilistic in the sense that it assigns a probability score to each pair of concepts to indicate the level of confidence.

The system makes an extensive use of (Hearst [100]) patterns and iteratively extracts concepts until no more new information can be extracted. Probase additionally includes a merging mechanism whereby it merges similar subconcepts or performs single and multiple sense alignments. Probase uses the Bing corpus of 1.6 billion pages to construct its taxonomy of 2.7 million concepts[8].

[7] http://gate.ac.uk (Last checked April 2013)
[8] http://research.microsoft.com/probase/ (Last checked April 2013)

3.3 Text Analysis Conference (TAC)

As a result of growing interest in knowledge bases, TAC has introduced in 2009 a dedicated track called "Knowledge Base Population" (KBP)[9]. The goal of KBP is to automatically identify equivalent entities in the KB, discover slots (attributes) about the entities, and finally extend the KB with any new attributes. This KBP track was organized to address the three key components of KBP, which are *entity linking*, *instance attributes discovery* (also called *slot filling*), and *cold start*.

3.3.1 Entity Linking

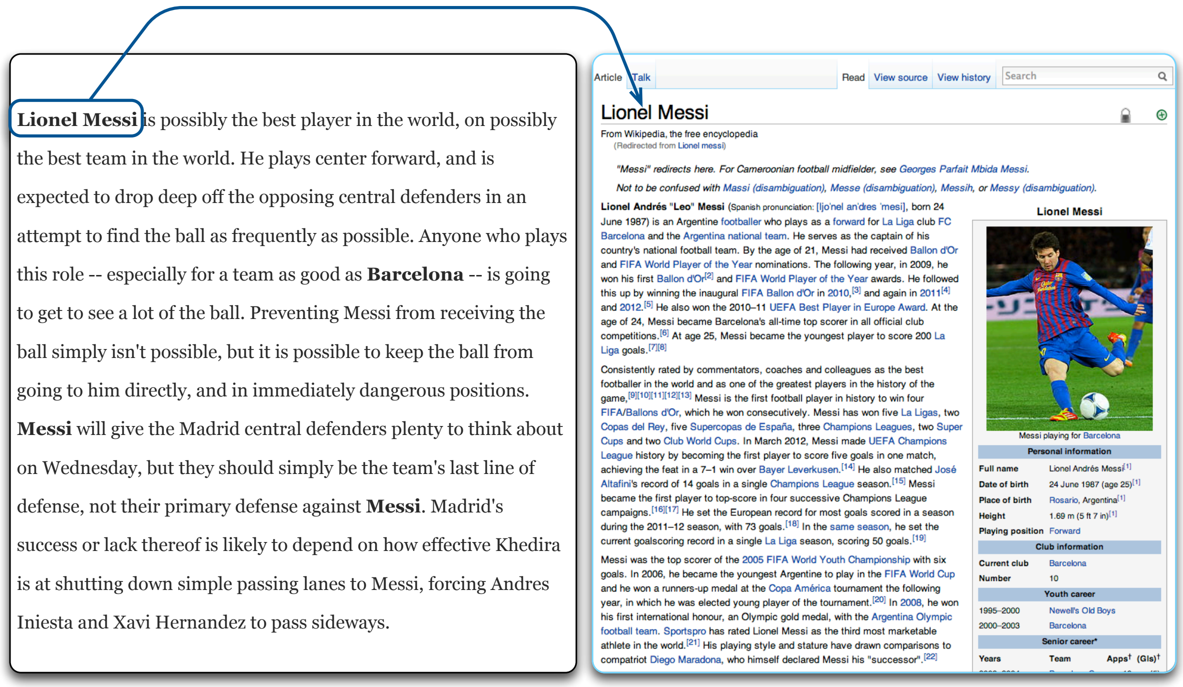
In the context of a general KBP task, we are typically interested in mapping a mention of an entity to an existing entity in the KB. When we use the term linking we typically think about the process of tying something together. Merriam-Webster has two definitions of this term:

- make, form, or suggest a connection with or between (things);
- connect or join physically.

Connecting, making or forming a connection between entities is known as the *"entity linking"* problem. Figure 3.2 shows an example of entity linking. In the news article, the entity Lionel Messi is linked to its corresponding KB entity `http://en.wikipedia.org/wiki/Lionel_Messi`.

A first major step during entity linking is the entity recognition. This is a research field in itself broadly known as *Named Entity Recognition* (NER). The NER task was for the first time introduced in 1995 at the Message Understanding Conference (MUC-6) [92]. It deals with identifying named entities in text of predefined set of named entity types [40, 52, 83, 97, 144]. A *named entity* is said to be a phrase that clearly identifies an item from a set of other items that have similar attributes. Example of named entity types are people names (first-name, last-name pseudonyms), geographic locations (cities, countries), organizations, addresses, birthdates, phone numbers etc. Named entity is a term widely used not only in information extraction but also in other related domains such as information retrieval, natural language processing and question answering. Due to its independent nature, the NER task was explored in

[9]`http://www.nist.gov/tac/2012/KBP/` (Last checked April 2013)



Lionel Messi is possibly the best player in the world, on possibly the best team in the world. He plays center forward, and is expected to drop deep off the opposing central defenders in an attempt to find the ball as frequently as possible. Anyone who plays this role -- especially for a team as good as **Barcelona** -- is going to get to see a lot of the ball. Preventing Messi from receiving the ball simply isn't possible, but it is possible to keep the ball from going to him directly, and in immediately dangerous positions. **Messi** will give the Madrid central defenders plenty to think about on Wednesday, but they should simply be the team's last line of defense, not their primary defense against **Messi**. Madrid's success or lack thereof is likely to depend on how effective Khedira is at shutting down simple passing lanes to Messi, forcing Andres Iniesta and Xavi Hernandez to pass sideways.

Article Talk Read View source View history Search

Lionel Messi

From Wikipedia, the free encyclopedia
(Redirected from Lionel messi)

"Messi" redirects here. For Cameroonian football midfielder, see Georges Parfait Mbida Messi.

Not to be confused with Massi (disambiguation), Messe (disambiguation), Messih, or Messy (disambiguation).

Lionel Andrés "Leo" Messi (Spanish pronunciation: [ljoˈnel anˈdres ˈmesi], born 24 June 1987) is an Argentine footballer who plays as a forward for La Liga club FC Barcelona and the Argentina national team. He serves as the captain of his country's national football team. By the age of 21, Messi had received Ballon d'Or and FIFA World Player of the Year nominations. The following year, in 2009, he won his first Ballon d'Or[2] and FIFA World Player of the Year awards. He followed this up by winning the inaugural FIFA Ballon d'Or in 2010,[3] and again in 2011[4] and 2012.[5] He also won the 2010–11 UEFA Best Player in Europe Award. At the age of 24, Messi became Barcelona's all-time top scorer in all official club competitions.[6] At age 25, Messi became the youngest player to score 200 La Liga goals.[7][8]

Consistently rated by commentators, coaches and colleagues as the best footballer in the world and as one of the greatest players in the history of the game,[9][10][11][12][13] Messi is the first football player in history to win four FIFA/Ballons d'Or, which he won consecutively. Messi has won five La Ligas, two Copas del Rey, five Supercopas de España, three Champions Leagues, two Super Cups and two Club World Cups. In March 2012, Messi made UEFA Champions League history by becoming the first player to score five goals in one match, achieving the feat in a 7–1 win over Bayer Leverkusen.[14] He also matched José Altafini's record of 14 goals in a single Champions League season.[15] Messi became the first player to top-score in four successive Champions League campaigns.[16][17] He set the European record for most goals scored in a season during the 2011–12 season, with 73 goals.[18] In the same season, he set the current goalscoring record in a single La Liga season, scoring 50 goals.[19]

Messi was the top scorer of the 2005 FIFA World Youth Championship with six goals. In 2006, he became the youngest Argentine to play in the FIFA World Cup and he won a runners-up medal at the Copa América tournament the following year, in which he was elected young player of the tournament.[20] In 2008, he won his first international honour, an Olympic gold medal, with the Argentina Olympic football team. Sportspro has rated Lionel Messi as the third most marketable athlete in the world.[21] His playing style and stature have drawn comparisons to compatriot Diego Maradona, who himself declared Messi his "successor".[22]

Lionel Messi

Messi playing for Barcelona

Personal information	
Full name	Lionel Andrés Messi[1]
Date of birth	24 June 1987 (age 25)[1]
Place of birth	Rosario, Argentina[1]
Height	1.69 m (5 ft 7 in)[1]
Playing position	Forward
Club information	
Current club	Barcelona
Number	10
Youth career	
1995–2000	Newell's Old Boys
2000–2003	Barcelona
Senior career*	
Years	Team Apps† (Gls)†

Figure 3.2: An Example of Entity Linking Problem. A News Article about Messi is Correctly Linked to the Corresponding KB Entity (Wikipedia).

many evaluation initiatives (Information Retrieval and Extraction Exercise –IREX, the CoNLL task[10]) for both specific languages and as a multilingual entity tracking (MET) project [143].

<ENAMEX TYPE="PERSON">Steve Jobs</ENAMEX> served as CEO and majority shareholder until <ENAMEX TYPE="ORGANIZATION">Disney </ENAMEX> purchased <ENAMEX TYPE="ORGANIZATION">Pixar</ENAMEX> in <TIMEX TYPE="DATE">2006</TIMEX>.

Figure 3.3: An Example of Annotated Block of Text in the MUC format.

A sentence tagged with NER typically associates the part of the text with the predefined type of entities. For example, Figure 3.3 illustrates the tagged sentence for the unannotated text *“Steve Jobs served as CEO and majority shareholder until Disney purchased Pixar in 2006”*. As can be seen, the annotated text block contains tags such as “ENAMEX” – Entity Name Expression (Person, Or-

[10]http://www.cnts.ua.ac.be/conll2003/ner/ (Last checked April 2013)

ganization, Location) and "TIMEX" – Time Expression.

Major approaches to the NER problem explored the of use linguistic grammar-based techniques [23, 189, 197] as well as statistical models [34, 77, 78, 89, 203]. In general, grammar-based systems have reported better precision, but at the cost of lower recall. The statistical methods on the other hand, require a larger amount of training data. The research in NER is mature and an evidence of that is that it is being extensively used in many commercial applications. Particularly, data mining applications make use of NER for a broad range purposes (e.g. marketing initiatives).

The second step in entity linking process is the *disambiguation*. Human language is inherently ambiguous without additional information. This additional information is needed in order to understand what mean when we say something. For example "apple" without any context can mean the fruit apple, the company "Apple" or a person's name "Apple". The correct sense of an ambiguous word/s may be appropriately chosen based on the context it occurs. This problem is known as the *word sense disambiguation* (WSD) problem [38, 40, 110, 122, 156, 201]. The goal is to assign the most suitable meaning to the ambiguous word in a given context. Even crowdsourced knowledge bases such as Wikipedia have acknowledged the problem of ambiguity with a large number of topics having more than one article. This problem is solved by linking different concepts to the corresponding article; the ambiguous topic is redirected to the disambiguation page in which the user can select appropriate topic of interest. According to [156], WSD relies on various sources of knowledge in order to associate the most appropriate sense to a term with the words in the context. There are many knowledge sources such as machine-readable dictionaries or semantic networks.

Proliferation of Wikipedia as knowledge sharing community brought a renewed interest in the WSD field. Many approaches [40, 89, 97, 144, 147] are now utilizing this KB as an external source of knowledge for solving WSD problems. The main reason that using Wikipedia as an underlying KB yields quite good results is that articles and links in Wikipedia articles are created manually.

However, there are open sets of problems that still remain challenging in entity linking. For example, as the human language evolve, new acronyms will be used (WMD became a widely used acronym for Weapons of Mass Destruction during the UN inspection of Iraq). Many researchers stressed that the problem of polysemy is hard to solve for every domain and for every problem. Another challenging area is the typing. While most efforts are now able to tackle the common types of entities such as person, organization and geographic locations, there are many other types of entities that are not recognized as distinct types.

3.3.2 Attribute Extraction

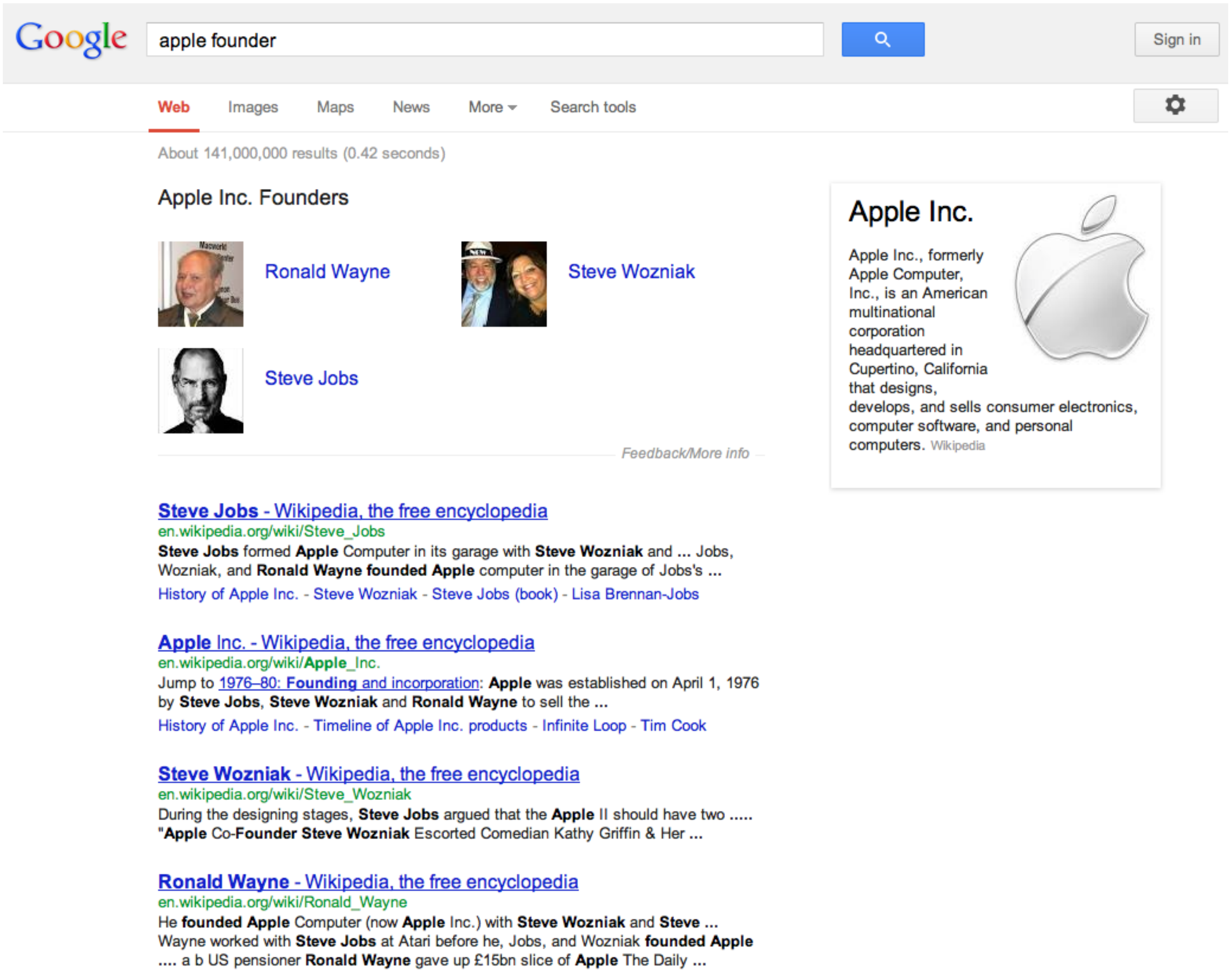
The main objective of attribute extraction (slot filling) task is to discover a range of attributes pertaining to a particular entity type. For example, a person entity may have the following slots: *date_of_birth*, *alternative_names*, *age*, *country_of_birth*, etc. In the context of Text Analysis Conference (TAC) KBP, this task is called "slot filling" and defined as follows:

> *Given a named entity and a pre-defined set of attributes ("slots") for the entity type, augment a KB node for that entity by extracting all new learnable slot values for the entity as found in a large corpus of documents. The reference KB is derived from English Wikipedia, while source documents come from English and Spanish. A diagnostic task, Slot Filler Validation, will be to determine whether a candidate filler in a document is a correct slot-filler for a given entity.*

Attribute extraction is being acknowledged important because it enables many other semantic applications beyond being part of the KBP task. A typical application area is exploiting a search query during retrieval of an entity. For example, Figure 3.4 shows the result for the query "apple founder" – an expressed information need of a user interested in the history and founders of the company Apple. In the top of the results page, there are three entities listed as the direct answer to the query. This illustrates that Google Knowledge Graph[11] exploits attributes of an entity (e.g. *org:founded_by*) for their integrated semantic search. The example shows clearly that exploring textual documents is not required in order to find out who the founders of Apple actually were.

Attribute extraction research has taken variety of directions from extracting using html structure (wrapper induction) [60, 117], class-driven [115, 162] and instance driven [9] methods. One of the challenges is the open nature of the problem: documents usually span various genres: news corpora, archives, Web documents, technical content, etc., which may conceivably introduce noises when extracting in domain independent mode.

[11]http://www.google.com/insidesearch/features/search/knowledge.html (Last checked April 2013)



About 141,000,000 results (0.42 seconds)

Apple Inc. Founders

Ronald Wayne
Steve Wozniak
Steve Jobs

Feedback/More info

Apple Inc.

Apple Inc., formerly Apple Computer, Inc., is an American multinational corporation headquartered in Cupertino, California that designs, develops, and sells consumer electronics, computer software, and personal computers. Wikipedia

Steve Jobs - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Steve_Jobs
Steve Jobs formed **Apple** Computer in its garage with **Steve Wozniak** and ... Jobs, Wozniak, and **Ronald Wayne founded Apple** computer in the garage of Jobs's ...
History of Apple Inc. - Steve Wozniak - Steve Jobs (book) - Lisa Brennan-Jobs

Apple Inc. - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Apple**_Inc.
Jump to 1976–80: **Founding** and incorporation: **Apple** was established on April 1, 1976 by **Steve Jobs**, **Steve Wozniak** and **Ronald Wayne** to sell the ...
History of Apple Inc. - Timeline of Apple Inc. products - Infinite Loop - Tim Cook

Steve Wozniak - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Steve_Wozniak
During the designing stages, **Steve Jobs** argued that the **Apple** II should have two .....
"**Apple** Co-**Founder Steve Wozniak** Escorted Comedian Kathy Griffin & Her ...

Ronald Wayne - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Ronald_Wayne
He **founded Apple** Computer (now **Apple** Inc.) with **Steve Wozniak** and **Steve** ...
Wayne worked with **Steve Jobs** at Atari before he, Jobs, and Wozniak **founded Apple**
.... a b US pensioner **Ronald Wayne** gave up £15bn slice of **Apple** The Daily ...

Figure 3.4: Results for the Query "apple founder" in Google.

3.3.3 Cold Start

The cold start can be seen as an integration of the Entity Linking and Slot Filling tasks to produce a new knowledge base independent of a reference KB. The cold start task is defined as follows: given a schema with an empty knowledge base, populate the KB by mining a large text collection. More specifically, the input components of this task are:

- a KB schema;
- a document collection;

- a set of named entity mentions.

The cold start is designed to evaluate the ability of a system to construct a KB from a text collection and the other two tasks basically depend on this first task. The KB created by a software system addressing this task is evaluated as a single connected resource and the knowledge should be consistent and accurately represent the content of the given input collection. The task is called "cold start" to convey two differentiating features of the evaluation: (i) the KB is created at the start of the task and (ii) the KB is initially empty, i.e. unpopulated. It is however assumed that there is a schematic information given upfront for relational facts that the KB will consist of. This means that identifying or discovering "new" attributes or slots for a given object type is not a requirement[12]. The latest cold start task, as of this writing, specifies three entity types (person, organization, and geopolitical entity) and as many as forty two relation types. Wikipedia is a great source of knowledge and can very well be used as verification engine for such a task. Therefore, the cold start task is dominated by entities about which Wikipedia does not have articles.

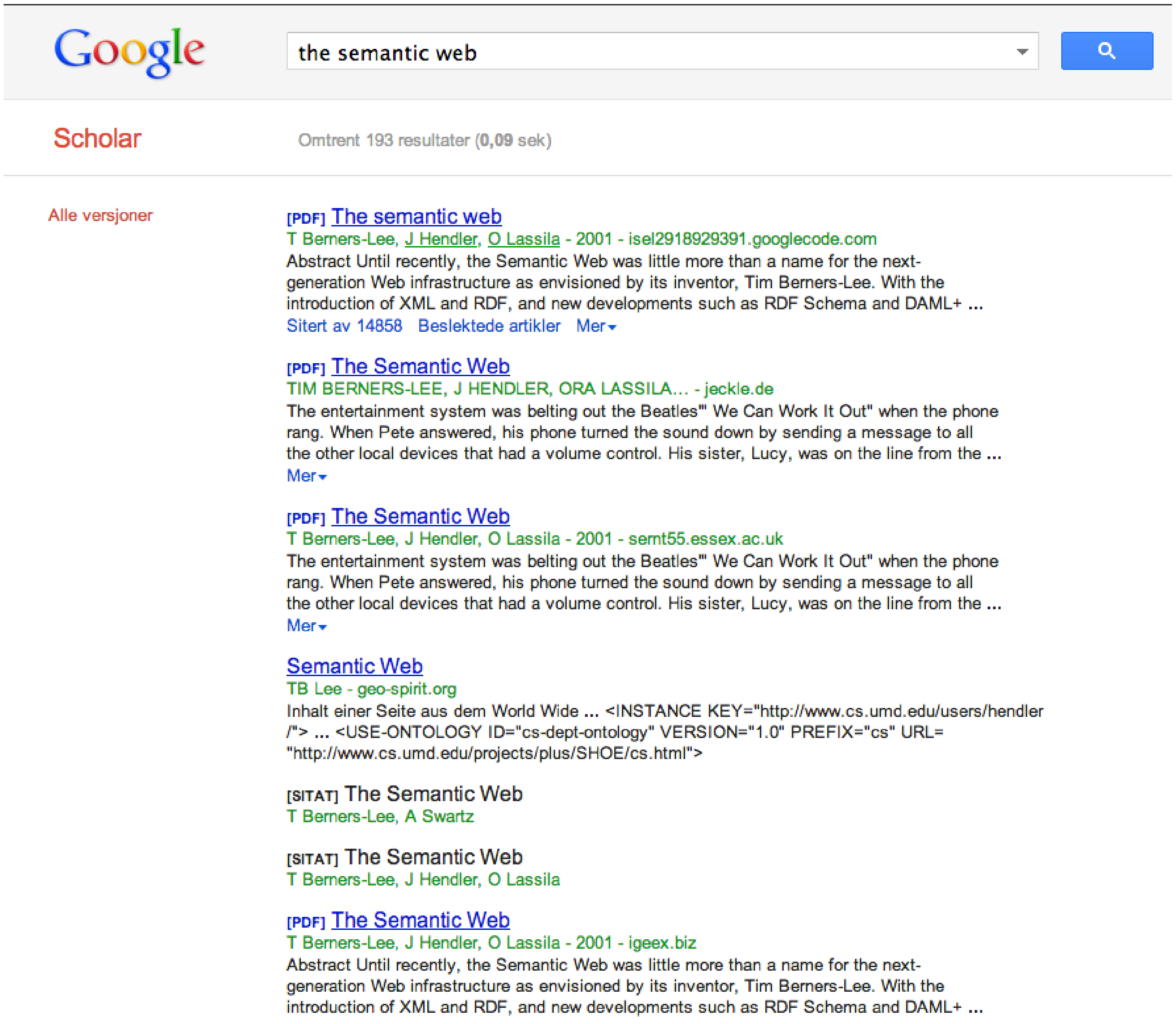
The three tasks described above within the context of TAC KBP provide the research platform for experimenting with the construction and maintenance of Semantic Knowledge Bases. One of the main motivations to maintain a knowledge base is related to the Linked Data movement, which is part of the Semantic Web – an environment in which applications can query the data, draw inferences using vocabularies, etc.

3.4 Entity Matching

Entity matching (also called duplicate identification, record linkage, entity resolution) is one of the important and challenging tasks in data integration and data cleaning [113, 169]. The main goal is to identify equivalent entities (objects, instances) and create correspondences as a result a *match* operation. An output typically consists of a set of corresponding elements, each of which indicates that certain elements of the one schema are related to certain elements of another schema.

Figure 3.5 depicts a result from the Google web search engine containing a list of hits for the query "the semantic web". This figure illustrates that a single real-world entity (in this case "The Semantic Web" article) may have many references and often there

[12]For more information, see the task definition for Cold Start TAC KBP 2012 `http://nist.gov/tac/2012/KBP/task_guidelines/` (Last checked April 2013)



Omtrent 193 resultater (0,09 sek)

[PDF] The semantic web
T Berners-Lee, J Hendler, O Lassila - 2001 - isel2918929391.googlecode.com
Abstract Until recently, the Semantic Web was little more than a name for the next-generation Web infrastructure as envisioned by its inventor, Tim Berners-Lee. With the introduction of XML and RDF, and new developments such as RDF Schema and DAML+ ...
Sitert av 14858 Beslektede artikler Mer

[PDF] The Semantic Web
TIM BERNERS-LEE, J HENDLER, ORA LASSILA... - jeckle.de
The entertainment system was belting out the Beatles" We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other local devices that had a volume control. His sister, Lucy, was on the line from the ...
Mer

[PDF] The Semantic Web
T Berners-Lee, J Hendler, O Lassila - 2001 - semt55.essex.ac.uk
The entertainment system was belting out the Beatles" We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other local devices that had a volume control. His sister, Lucy, was on the line from the ...
Mer

Semantic Web
TB Lee - geo-spirit.org
Inhalt einer Seite aus dem World Wide ... <INSTANCE KEY="http://www.cs.umd.edu/users/hendler/"> ... <USE-ONTOLOGY ID="cs-dept-ontology" VERSION="1.0" PREFIX="cs" URL="http://www.cs.umd.edu/projects/plus/SHOE/cs.html">

[SITAT] The Semantic Web
T Berners-Lee, A Swartz

[SITAT] The Semantic Web
T Berners-Lee, J Hendler, O Lassila

[PDF] The Semantic Web
T Berners-Lee, J Hendler, O Lassila - 2001 - igeex.biz
Abstract Until recently, the Semantic Web was little more than a name for the next-generation Web infrastructure as envisioned by its inventor, Tim Berners-Lee. With the introduction of XML and RDF, and new developments such as RDF Schema and DAML+ ...

Figure 3.5: An Example of Multiple References to the Same Article.

are variations in spellings. This is a problem when entities are highly heterogeneous and there is a great deal of inconsistencies in their representation.

The main objective of any entity matching strategy is to decide whether given entries of two or more entities correspond to the same real-world entity. Entity matching has its roots in the database community as a common task in migration of legacy data from multiple sources into a new one [68].

The problem of entity matching can be formalized as follows. Given the data source

D_1 and D_2 and the sets of entities $A \in D_1$ and $B \in D_2$ a particular entity matching approach must find all corresponding entities $A \times B$. A correspondence therefore denotes an interrelation between the two entities from two different data sources. Entities typically have few attributes. However, the number of instances can potentially be large. Hence, the Cartesian product for discovering correspondences $A \times B$ is often inefficient. To address this issue, entity matching systems normally employ so-called *blocking* process which enables reducing the list of potential candidate matches for a current object. This technique defines a partition of candidates to be matched for a particular entity and these partitions are organized into blocks. Hence, the name "blocking". The match operation for an entity is then restricted to the specific block of candidates. The result of matching is often assigned a similarity score s to a particular correspondence, which is between [0,1] and this score indicates the strength of the correspondence between the two objects. This "smart" way of matching used in the blocking methods, makes use of similarity metrics, which can be lexical, structural, constraint-based or contextual (see Section 2.5.2).

In the context of this thesis, many of the tasks that are tackled remind of the problems of entity matching, especially integrating data from diverse sources in the LOD during the linking and enrichment.

3.5 Entity Search

The ultimate goal of building and maintaining a knowledge base is to make it usable in a variety of applications. One of the primary benefits of semantic knowledge bases is that they can be used in various information access contexts. Often users are interested in precise answers to their information need. What is meant with precise answers is that the user might be interested in discovering and exploring specific types of objects instead of documents. Objects of interest are identifiable things that can be referred to such as music, books, movies, people, organizations, places, events, etc. In fact, a recent research indicates that around 40% of Web search queries target objects [168]. The search for this kind of objects is generally referred to as *entity search*, also called *object search*[13].

Entity search is the task of retrieving entities given a query contrary to the traditional document retrieval [14, 64, 158]. In recent years, a range of services and applications

[13]For simplicity, we use the term entity search

have been directed towards this domain both in academia and industry. The formal definition of the task is given in [168] and it is based on *"answering arbitrary information needs related to particular aspects of objects, expressed in unconstrained natural language and resolved using a collection of structured data"*. The main challenging part of this task is that unlike documents, entities first need to be identified and recognized in the mixed space of structured and unstructured texts such as Web pages. Therefore, this task is of a cross-disciplinary nature in many ways and is at the intersection of research areas of Information Extraction, Natural Language Processing, Machine Learning and Information Retrieval.

The earlier research on entity search was primarily performance-centric and reported experiments were targeting efficiency [18, 48]. However, the main evaluation criteria for this task are undoubtedly related to effectiveness – and most importantly the precision because of the nature of the task. The importance of entity search also attracted the evaluation campaigns; Text REtrieval Conference (TREC) has been organizing dedicated entity track for several years with the most recent one in 2011 [15]. The main task in 2011 was to provide a list of related entities, given some criteria, which is a further development of the traditional ad-hoc search task.

Recently, NIST initiated a new *knowledge base acceleration* track at TREC – TREC KBA[14]– introduced in 2012 [81]. The initiative is running in 2013 as well. The main task of KBA track is given a set of input topics (entities), the system should filter a stream of time-ordered documents according to their pertinence for automatically editing Wikipedia articles describing those entities. This basically means, given a Wikipedia article "Boris_Berezovsky_(businessman)" filter the stream of documents for editing (by discovering their level of relevance). The 2013 version of the track, additionally introduced a new task called *Streaming Slot Filling*, which is concerned with detecting changes in the slot value of a given entity.

A general trend seems to have recognized the importance of entity search and therefore the above mentioned evaluation campaigns indicate a move beyond a typical ad-hoc search task and maintaining the knowledge base with an up-to-date information at any given time.

[14]`http://trec-kba.org` (Last checked April 2013)

3.6 Summary

This chapter provided an overview of the state of the art in knowledge extraction and related fields. We started with describing the automatic ontology construction and quickly described some of the approaches. Next, we surveyed knowledge extraction from natural language documents, the reviewed its history and some of the systems proposed. An inherent part of the approach presented in this thesis is to extract knowledge from text and therefore it is important to note the similarities and differences with respect to patterns and their definitions.

The approach is similar in the sense that it makes use of patterns and can run iteratively. The major difference is that the patterns are defined in a flexible way thus enabling merging of similar patterns. To the best of our knowledge, none of the state of the art approaches deal with the issue of identifying the alternative labels of an entity which is addressed in this work.

Additionally, we reviewed the initiatives at TAC as well as entity matching which is employed in various parts of this work. Finally, entity search was described as a successful application and use of semantic data.

4 Metadata as Knowledge Source

Metadata is used as an initial source for extraction of knowledge for the knowledge base population vision explored in this thesis. One of the requirements is to make explicit the entities and relationships that can be extracted from bibliographic records, which consequently will enable semantic aware integration and reuse as well as new services based on the resulting knowledge.

Even though metadata ca be a rich source of knowledge, it is often not trivial to reuse metadata in new contexts. Metadata of the bibliographic world is managed as distinct records, and the major issue is that entities that implicitly are described in a record such as authors are usually identified *by descriptions only*. Adapting to new semantic models is therefore a complex challenge that on the one hand requires solutions for mining existing bibliographic information to discover the structure of entities and relationships represented by the metadata. On the other hand there is a need for solutions for explicit representation of these structures in ways that meet the requirements of the environment where this information is created, maintained and used.

4.1 Bibliographic Catalogs

The structure of catalogs maintained by libraries consists of a set of bibliographic records. In fact, creating and organizing bibliographic records is what we usually call *cataloging*. The descriptive elements that constitute a bibliographic record are based on a common set of rules. The underlying principle in cataloging is that an item (e.g.

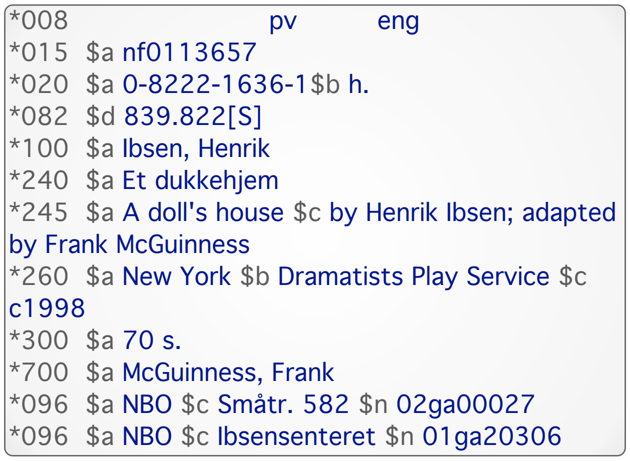
a book) should be described using the information that is found on the item and the bibliographic record should include the access points that are needed by users for searching. In the description, one distinguishes between main and added entries for titles, persons and corporate bodies.

The main entry is the primary access point the record should be organized under - often the author - and added entries are other access points that users should be able to find the record under. This framework has the origin in card catalog where one had to decide what catalog cards to create for a particular publication. Although computer catalogs are now the norm, the current cataloging rules originate from the card catalog era. The entries for persons (and corporate bodies) are authority controlled, to ensure that names are used consistently throughout the catalog using separate authority files.

MARC

Exchange of bibliographic metadata is an important service in the library domain and the MARC format is a key bibliographic standard for the flow of information between libraries [124]. The main part of library holdings consists of mass produced publications and the benefits of reuse are many. Libraries may create or import and adapt records from others. Some agencies are solely producers of bibliographic data and systems, and do not offer any end-user services. National bibliographies are typically created under the auspices of national libraries and attempt to list everything that is published by a country, either based on the aggregation of bibliographic data or by creating bibliographic metadata, which is a resource for others to reuse. The MARC format is one of the key bibliographic standards for this information flow [125]. MARC is an acronym for **MA**chine **R**eadable **C**ataloging and is an encoding format that was developed in the late 60's at the Library of Congress. The standard specifies a simple and generic data structure for bibliographic descriptions as well as other types of records. Information is organized as a set of fields identified by three octets (tags) that can be of two types. Tags starting with two zeros are reserved for control fields (also called reference fields) that can hold data of variable length and the remaining tags are used for datafields that can have subfields identified by a code.

Each MARC record, such as the one depicted in Figure 4.1, typically describes a single publication and each datafield reflects a logical grouping of the data elements that together describe a specific aspect of a publication without dependencies on other records. For instance, publication information is stored using the MARC datafield 260, with three subfields for place of publication ($a), publisher ($b) and publication year ($c). Records are self-contained information units, which means that each record contains all the information needed for a descriptive identification of a publication.



```
*008                    pv        eng
*015  $a nf0113657
*020  $a 0-8222-1636-1$b h.
*082  $d 839.822[S]
*100  $a Ibsen, Henrik
*240  $a Et dukkehjem
*245  $a A doll's house $c by Henrik Ibsen; adapted
by Frank McGuinness
*260  $a New York $b Dramatists Play Service $c
c1998
*300  $a 70 s.
*700  $a McGuinness, Frank
*096  $a NBO $c Småtr. 582 $n 02ga00027
*096  $a NBO $c Ibsensenteret $n 01ga20306
```

Figure 4.1: A MARC Record Describing Henrik Ibsen's "*A doll's house*".

4.2 The FRBR Model

Libraries are now starting to realize that the information model used for current bibliographic records may have to be modernized to be able to adapt to new requirements. Much of the motivation is driven by an increased interest in creating more advanced end user services. These services are mainly directed towards exploring the contents of library catalogs.

The conceptual ER-based model presented in the IFLA[1] report on Functional Requirements for Bibliographic Records (FRBR), is an important foundation for this renewal [31, 188]. The library environment is unfortunately inherently conservative and change resistant because of the huge number of existing collections, systems and practitioners involved worldwide [187]. A major hurdle for adopting new models is the amount of existing legacy data and the many challenges that are related to the reinterpretation of this information in the context of the FRBR model. A library record can be a very rich source of knowledge about many aspects of a publication, but most

[1]International Federation of Library Association and Institutions, http://www.ifla.org (Last checked November 2012)

of this knowledge is unfortunately only released after the record is found, when it is displayed to the end user.

The model identifies the main entities and relationships that are of interest to end users and it is designed to support the following four tasks:

- *find* entities that correspond to user's expressed information need;
- *identify* entities;
- *select* entities;
- acquire *access* to entities.

The FRBR model depicts intellectual products as four interrelated entities: *Item*, *Manifestation*, *Expression* and *Work* (Figure 4.2) *Manifestation* and *item* entities are equivalent to the commonly known concepts of publication and copy respectively. The intellectual contributions found in publications are modeled in FRBR as the *expression* and *work* entities. A *manifestation embodies* one or more expressions whereas each expression *realizes* a single work. An *expression* is the intellectual product that we recognize as unique content in the shape of text, sound, images or other types independent of the specific formatting it has been given in different *manifestations*. The *work* entity is the most abstract and is needed because of the way we refer to and reason about intellectual and artistic creations at a more general level. The play in prose by Henrik Ibsen "Et dukkehjem" (A doll's house) exists in numerous translations where each translation is considered to be a specific *expressions* which *realizes* the same work. The main advantage of the *work* entity is that it enables collocation of intellectually equivalent products and enables the modeling of closely related intellectual products in tree-like structures. The FRBR model additionally includes entities for agents (*person* and *corporate body*) and the relationships they have to the different intellectual and physical products. Shakespeare *created* the *work* Hamlet, and the person responsible for a specific translation is related to a particular expression with *has realized* relationship. The FRBR model defines the entities that occur as subjects of works, and the model describes the attributes that are needed to for each entity and a rich set of relationships that may exist between the entities.

On one hand, FRBR model is considerably different from the data structure that is found in MARC records. On the other hand, the different entities are often implicitly or explicitly described in the bibliographic records. As an example, the MARC datafield 245 (subfield $*a*) in Figure 4.1 is the title of the publication and is normally

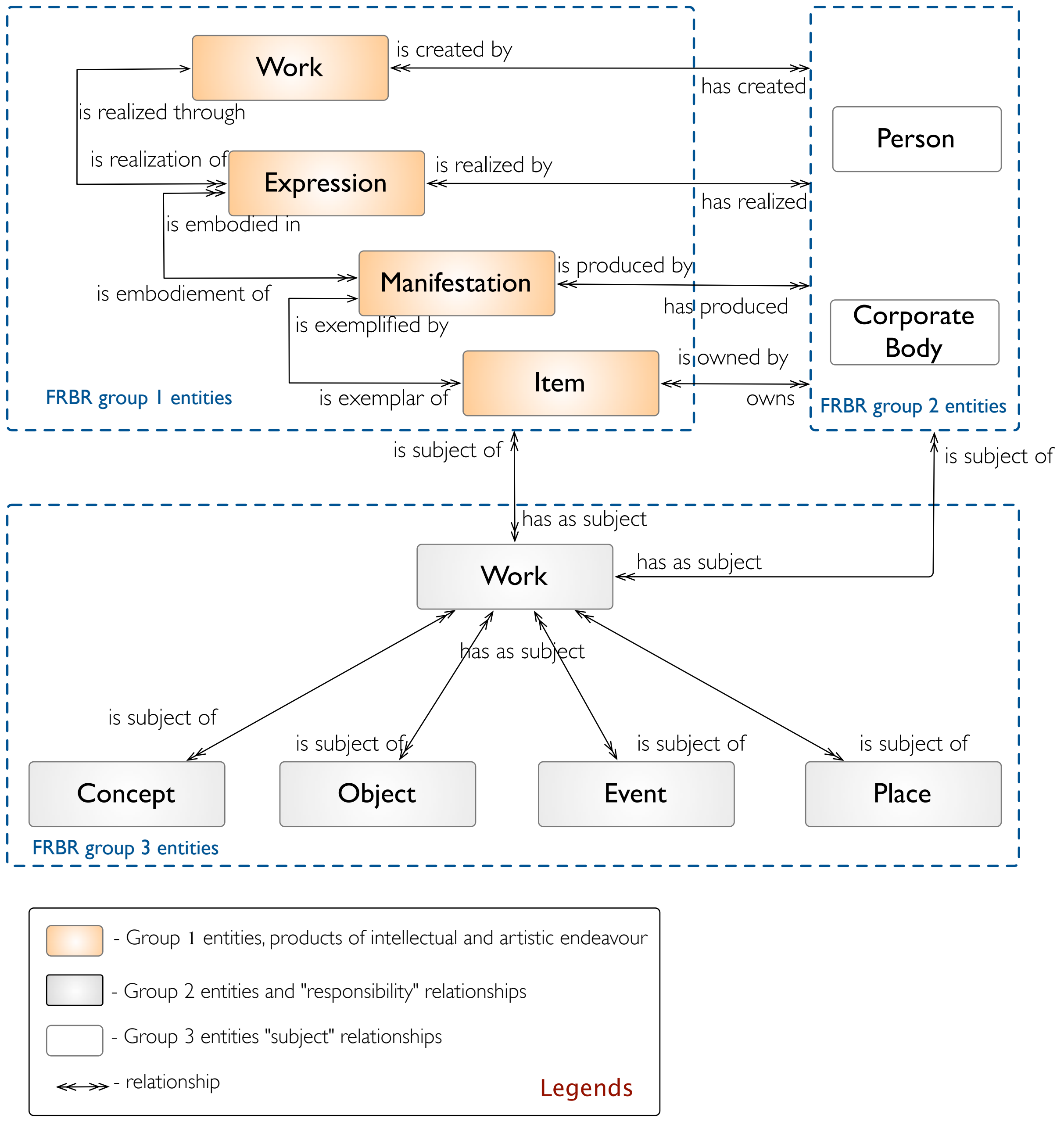


Figure 4.2: The FRBR Model.

considered an attribute of a FRBR *manifestation* entity. The FRBR model is not intended to serve directly as a data model, but there has been a significant interest in the use of the model as a foundation for new types of services and user interfaces.

As a conceptual model, the main contribution of FRBR is a more knowledge-like representation of bibliographic data that enables many types of applications such as exploratory interfaces where users are presented with listings of works for each author and can follow the relationships to learn about and find the versions or editions they prefer.

The FRBR report was published over a decade ago and has so far not been extensively implemented in library systems. However, the model is far from being neglected; the promising LOD vision and the increasing demand for semantic data has revitalized the interest in the model. Some library systems have developed user interfaces that are inspired by the model and using existing MARC-based information such as uniform titles and material codes to group records. Additionally, there is a number of prototypes and production systems available that are at least partially based on the model. Finally, the FRBR model is rather simple to implement if one does not have to consider compatibility with the existing data.

4.3 FRBR Ontologies and Vocabularies

The FRBR model was initially intended as a conceptual framework for bibliographic data, but the report gives a detailed description of entities, relationship and attributes that may be used to define type-vocabularies. One of the first RDF vocabularies based on the model was published in 2005[2]. The need for a more official vocabulary is acknowledged by IFLA and there is ongoing work to establish a vocabulary under the IFLA namespace[3].

The attributes in the FRBR report are listed as generic types and are not intended to dictate any particular implementation. The mapping between MARC 21 and FRBR attributes that was published by the Library of Congress – Network Development and MARC Standards Office [125] shows that there is a difference in the level of granularity. Some FRBR attributes are very generic and map to multiple fields/subfields in MARC 21 while other FRBR attributes are very specific and map to more generic MARC 21 fields. Through the development of a new version of the anglo-american cataloging rules - RDA (Resource Description and Access), the library community may be able to address this problem in the future. RDA uses the concepts defined in FRBR

[2] http://vocab.org/frbr/core.html (Last checked April 2013)
[3] http://metadataregistry.org/schema/show/id/5.html (Last checked January 2013)

as the main terminology and defines all bibliographic data elements in this context. In this way, RDA may be a source for future attribute vocabulary that is consistent with the FRBR model as well as with existing bibliographic data elements.

The FRBRoo ontology is a different approach to the formalization of FRBR as an implementation model [67]. The underlying idea behind this ontology is to merge the FRBR model with the standardized CIDOC Conceptual Reference Model (CRM) that provides definitions and a formal structure for describing the concepts and relationships used in cultural heritage documentation [66]. The merged ontology expresses FRBR with the formalism used in the CRM model and it is intended to facilitate the integration, mediation, and interchange of bibliographic and museum information, as well as elaborate and clarify the more pragmatic or unclear parts of the FRBR model.

MARC records can in principle be expressed directly in RDF as demonstrated in [178], but with limited advantages. Simply transforming a MARC record to a corresponding RDF representation, only makes the data available for tools unable to process native MARC and does not contribute to the machine-interpretation of the implicit meaning in the records. For MARC-based data in strict MARC 21 or UNIMARC the tags and codes represent a certain level of strict typing, but in practice there are parts of the data that have a contextual meaning in the sense that the interpretation depends on the values in other fields. An "added entry" title in a MARC record may mean different things. It can be an alternative title for the cataloged item, the title of a part or the title of a related publication. Indicators may specify the interpretation of fields but in other cases the meaning is only revealed if interpreted in the context of cataloging rules and common patterns in the data. Essentially this means that a direct translation to RDF/XML only makes the data available for other tools, but does not contribute to the machine-interpretation of the meaning.

4.4 The BIBFRAME Model

Recognizing the future needs of the library community, the Library of Congress has recently launched the Bibliographic Framework (BIBFRAME) Transition Initiative with the focus on determine the transition path for bibliographic records to a more Web-based and Linked Data standards [146]. BIBFRAME is envisioned as a foundation for future bibliographic descriptions in the networked Web-based world and the framework is to a large extent motivated by ever increasing user expectations. The BIBFRAME model shares, in part, the entity-relationship centric view of the FRBR model with the main objectives set as follows:

- Focusing on description of conceptual content distinctly from the physical manifestations;
- Identifying entities unambiguously using stable, machine-friendly identifiers;
- Capture and expose relations between and among entities.

Since the inception, BIBFRAME has been embracing the Linked Data and the development of the model is clearly influenced by the Linked Data developments. Specifically, the use of RDF in BIBFRAME allows for further annotations and enrichment. The model of BIBFRAME includes several entity types or in the BIBFRAME vocabulary *classes*:

- **Work** which is an abstract conceptual cataloging item similar to FRBR work entity. This entity can have relationship to authorities such as topic, person, place as well as to entities associated with its creation such as person, organization, meeting.
- **Instance** is a resource reflecting a physical or digital material embodiment of a (BIBFRAME) Work. An *Instance* can be embodiment of one and only *Work*.
- **Authority** is a key authority concept that has defined relationships to *Work* and *Instances*.
- **Annotation** is not a new concept to the bibliographic world but its intended meaning in the context of BIBFRAME is somewhat different. BIBFRAME *Annotation* is about providing assertion-like statements about works, instances, authorities. This feature differentiates the model from the traditional bibliographic practice. This is a statement that can naturally be encoded in the RDF, which is the modeling framework adopted in the model.

The overview, tools and demonstrations along with the progress of the effort is available online at: http://bibframe.org.

4.5 Expressing MARC in RDF

A MARC-based record can in theory be expressed directly in RDF as demonstrated in [178]. However, simply transforming a MARC record to a corresponding RDF/XML representation only makes the transformed data available for tools unable to process native MARC. For MARC-based data in MARC 21 or UNIMARC the tags and codes represent a certain level of strict typing. In practice there are parts of data having contextual meaning in the sense that the interpretation depends on the values of other fields. An "added entry" title in a MARC record may mean different things. It can be an alternative title for the cataloged item, the title of a part or the title of a related publication. Indicators may specify the interpretation of fields but in other cases the meaning is only revealed if interpreted in the context of cataloging rules and common patterns in the data. Essentially this means that a direct translation to RDF/XML only makes the data available for other tools, but does not contribute to the reuse value of the knowledge, simply because it maintains a meaning that does not make sense for other uses.

A mapping between MARC 21 fields and the FRBR attributes is presented in [125]. However, the use of RDF relies on URI-based vocabularies and node identification. Unfortunately, there is no tradition of any of these in common metadata management systems used in cultural heritage domain. Nodes (objects/entities) are identified by description only and will turn into *blank nodes* in an RDF representation. Though cultural institutions for decades have utilized authority files for the description of authors and other actors, they still use descriptions only when referring to persons.

RDF and OWL are not very readable (by humans) when written in XML or as RDF triples. One reason of this is that RDF/XML is often regarded as verbose [108], but the major part of the issue of the readability is the representation of OWL constructs in RDF/XML or RDF triples. Furthermore, the problem of verbosity brings along another issue - that of the size. Once the size of the file is large it requires more computing resources to process. The problem, however, is that every resource must have URIRef, i.e. resources must be identifiable on the Web. In a record-based library world information is usually organized in MARC records, which makes generating globally visible and unique URIRefs a significant challenge. The RDF solution would require every record to have a global identifier since the primary purpose of this framework is to describe Web resources. Unfortunately, the library community is slow adopting new Semantic Web technologies such as RDF/XML, not to mention converting their MARC based records fully into RDF or any other XML-based format. As a conceptual framework for bibliographic records the FRBR model offers a more formal semantic model of the entities and relationships that are of main concern to end users.

4.6 Metadata Formats for FRBR

Many voices in the library community have asked for formats that can be used with FRBR, but a main issue is that new formats need to be compatible with the existing metadata to be of any large-scale use. Solutions for representing FRBR using XML has been developed in the Variations project by the use of Dublin Core application profiles [171] and another solution is developed by the eXtensible Catalog organization[4]. Although they enable the export of MARC to FRBR compatible form, they do not consider backwards compatibility with MARC, which is likely to be a major obstacle for the use of these formats with existing systems.

Using RDF or OWL as representation has the advantage of making the knowledge directly accessible for semantic aware integration and reuse, but as discussed in the previous sections, this requires the use of appropriate vocabularies for all aspects of the model as well identifiers for all entities. RDF and OWL are important technologies for implementing the Semantic Web by enabling information to be exchanged and integrated. The primary exchange syntax for RDF and OWL is RDF/XML and even if XML itself is a human readable format, the resulting serialization of RDF is only designed to be machine readable. Turtle [20] and N3 [24] are non-XML alternative syntax with a more compact and human readable textual form.

A different possibility for disseminating FRBR-based information is to use other frameworks capable of expressing entities and relationships such as the Open Archives Initiative Object Reuse and Exchange (OAI-ORE) [119, 120]. ORE defines a standard for the description and exchange of aggregations of Web resources, such as compound digital objects and multiple media types including text, images, data, and video. Because there are many FRBR manifestations available or described on the web, there is a potential benefit of using ORE in combination with FRBR-based types to cluster manifestations that embody the same expression, and group expressions by work. The aggregations and resource maps of ORE provide a meta-format for making such

[4]http://www.extensiblecatalog.org (Last checked January 2011)

structures accessible for web crawlers.

4.7 Converting MARC to the FRBR model

Different experiments, prototypes and systems have in the last decade attempted to interpret existing MARC records using entities and relationships defined in the FRBR model. One of the earlier experiments was performed by Hegna et. al. using the Norwegian and Finnish national bibliographies and they demonstrated how existing data elements could be used to identify the *work* and *expression* entities for selected authors [101]. An important problem identified in this study is the difficulties in interpreting records with added entries which are often used when the content consists of multiple individual parts (books with two or more novels, essay collections, etc.). Another problem they discuss is the inconsistent use of the key information that is needed for identifying *works* and *expressions* correctly.

One of the algorithms for interpreting MARC records using the concepts introduced in the FRBR model, is the OCLC work-set algorithm [104]. It is developed for records in the MARC 21 format and implements a strategy for selecting work-related information which is used for clustering records that describe the same *work*. The algorithm basically treats all records as describing a single *work*, which partly is a consequence of the MARC 21 format and current cataloging practice that seems to favor the use of descriptive contents notes rather than structured added entries when cataloging publications that have multiple distinct parts – such as books containing multiple novels and essay collections. The FictionFinder[5] prototype, which makes use of the algorithm, demonstrated how the FRBR model can be used to create listings of works by author and additionally supported browsing by genres, characters, settings and literary awards.

Later experiments include the TelPlus project that processed records related to Nobel laureates [132] from different catalogs including both MARC 21 and UNIMARC. The algorithm for clustering works is somewhat comparable to OCLC work-set algorithm but with a different approach to identify and merge equivalent entities based on string matching. The prototype manages different MARC formats and it treats expressions and manifestations as distinct entities. A prototype user interface demonstrating the results was evaluated in terms of usability.

[5]http://www.oclc.org/research/activities/fictionfinder (Last checked January 2011)

The Variations project at the Indiana University Library has a different approach as they attempt to interpret records related to music using a strategy for interpreting added entries as separate entities [171]. Music is often cataloged with a more extensive use of added entries for titles and performers and composers of the various tracks on a CD. The prototype looks up the main title to identify if it is a collective title (generic uniform title) in which case the added entries are interpreted as the main content instead. The Scherzo prototype[6] can be used to search for and explore the catalog by composer, performer, instrumentation and work.

The FRBRizer approach [3] was developed to extract FRBR entities and relationships and the main principle is based on the assertion that a proper interpretation of bibliographic records is best solved by distinguishing between the extraction of the entities and relationships in each record. To create a complete interpretation of a MARC record, we need to interpret all fields and infer what entities the record describes and how they are related. Although a majority of bibliographic records describe a simple structure that consists of a single work *realized through* a single expression *embodied in* a manifestation – which is somewhat straightforward to extract – there are many records having complex structure to introduce significant noise in the result if they are misinterpreted. Additionally, a simple interpretation will often ignore many of the works that are of main interest to end users. The FRBRizer tool we are using to extract FRBR entities and relationships from bibliographic records was initially developed for an experimental conversion of the Norwegian BIBSYS database [2] and has later been further developed to support more advanced interpretations.

All MARC records have a generic structure of datafields and subfields and the rules for the identification of entities can be defined using selection statements with conditions. Establishing the relationships that exist between the entities is a matter of using conditions based on structural aspects of the record, different indications that can be found in specific fields as well as general knowledge about the cataloging practice. Finding the attributes of an entity can be based on a mapping table between entities and MARC fields. This approach can be generalized into a parameterized function/template for converting the MARC record into a set of interrelated entities which we have implemented using XSLT. Rules for all the entities that possibly can be found are described in a specific format which is used to generate an XSLT-based conversion program that contains a template for pair of selection statement and condition. The solution is generic in the sense that rules can be adapted to any MARC

[6] http://webapp1.dlib.indiana.edu/scherzo/ (Last checked January 2011)

formats and we are able to interpret all possible occurrences of entities including works and persons that appear in subject entries, added entries, contents notes and series statements etc. This tool uses MARC records in XML as input and produces a set of FRBR-based records with relationships expressed as links.

MARC records are in no way easy to interpret as FRBR entities and relationships, and there are many sources of errors and misinterpretations that may cause false positives, missing entities, and false or incorrect relationships. A major problem with attempts to create a full FRBR-based representation of a record is that there will be dangling or untyped entities if it is impossible to automatically interpret a record. This typically occurs often for added entries when there is not enough information to interpret the meaning of the entry. Additionally there are major problems with records having *n-m* relationships between entities. A MARC record is sometimes only a list of titles and names with little or no information about how entities are related.

Different solutions for interpreting MARC as FRBR are important contributions because they can be used to migrate MARC data to other formats or can be used to correct or enhance MARC records and in this way enable the future implementation of the full FRBR model in current catalogs [35]. An important aspect of interpreting or converting MARC as FRBR that to our knowledge has not been addressed, is the assessment of the result. Projects and experiments have explored the possibilities for interpreting records and have looked at different solutions for clustering or merging equivalent entities, but to compare and evaluate different strategies and approaches, we need systematic ways to determine the level of quality that can be achieved by different strategies and techniques.

A major benefit of extracting entities and relations described in the metadata is the ability to support new and innovative services, for example based on Linked Data and other semantic technologies. One such service is the semantic search [95]. The kind of queries users are typically interested can be articulated in the following questions:

- What works of "Ibsen" have been translated into English?
- Which films are based on the Stieg Larsson's crime novels?
- What is the relation between "Bored of the Rings" and "Lord of the Rings"?

These are simple examples, but it is not an easy task for a novice user to obtain adequate answers in current Web search engines. The framework proposed in this thesis is to bring the results interpretation of metadata further, namely to integrate the entities and relations described in the metadata into a Knowledge Base that is capable of delivering advanced semantic services.

4.8 Identifiers for Bibliographic Entities

The interpretation of types in Semantic Web data depends on the use of URIs and the need to refer to the same entity using the same URI. Bibliographic information usually contain the identifiers that are available for a publication such as ISBN, ISSN and ISMN. Unfortunately these only cover the manifestation entities and there is even a large body of such entities that we do not have identifiers for. For other entities such as works, expressions and persons, there is an additional lack of specifically-designated fields for identifiers in the MARC formats [82] and there is no tradition for creating and using such identifiers in the library community. Recent standards such as ISWC, ISAN and ISTC can potentially be used for the identification of works and expressions [167] if they are accessible when creating bibliographic information. Libraries mainly use a combination of key attributes for identification such as name and dates for persons, but have a strong tradition in the use of authority files to ensure that such entities are consistently identified within a collection. The VIAF project is an ongoing initiative to create a global authority registry[7] and may be able to provide globally unique identifiers for persons and corporate bodies.

Even if identifier schemes are available, associating the proper identifier to the proper entity in legacy data is a challenge. Several works have previously explored the area of identification of entities and proposed algorithms for duplicate detection techniques [42, 76, 113]. The simple and often usual approach is to construct a key based on the descriptive attributes (such as title, sub-title, author) of entities and use this key for comparison. The comparison is performed based on a decision tree/table or set of rules. The main issue with these techniques is data inconsistency. MARC records referring to the same global entity may have variations in describing various attributes of the entity, and the descriptions in legacy data may not be consistent with descriptions that can be found for the entities that have been assigned globally unique identifiers in other systems.

[7] http://www.oclc.org/research/activities/viaf/ (Last checked January 2011)

4.9 Summary

This chapter discussed and explored metadata as a source of knowledge. Particularly, bibliographic information is the type of metadata explored in this thesis and a gentle introduction to bibliographic catalogs has been presented. The information model of catalogs needs to be modernized and the FRBR model was discussed as a visionary model for such modernization. More specifically, several ontologies that are based on this conceptual model were discussed followed by the BIBFRAME model – a transition initiative by Library of Congress. In this thesis, we explore the use of FRBR as an underlying conceptual model and therefore we have discussed attempts to express existing bibliographic information with the data model of the Semantic Web and converting the library records to the FRBR model. Finally, an important aspect of representing data in the Semantic Web, identification of entities and the use of URI has been presented.

5 A Generic Framework: An Overview

5.1 Introduction

The generic framework presented in this thesis can be used in the data preparation phase of publishing data, i.e. populating and developing knowledge bases. Even though metadata is used as the initial input, unstructured text arguably contains much more information and as such is a highly useful source of complementary information resources. Therefore, the framework uses the existing metadata as the initial source and text documents to complement and supplement the metadata extraction. Furthermore, the supplementary part can run in autonomous mode.

The entity-oriented basis envisioned in the framework enables solutions for migrating and integrating information in environments where data is required to be sharable, exchangeable, extensible and reusable. The use of entities is one of the most important elements in a graph-like organization of entities – a data storage model used by many Semantic Web applications. In such data models, nodes represent entities and edges denote relations between entities. By approaching the problem with an entity-oriented view, the domain of interest will be explicitly described which is otherwise hidden in the descriptive information recorded in the legacy systems.

The visualization in the Figure 5.1 provides a brief summary and this chapter gives an overview with the focus on the different steps within the framework. The framework considers two sources used for extracting relational facts, followed by a linking step:

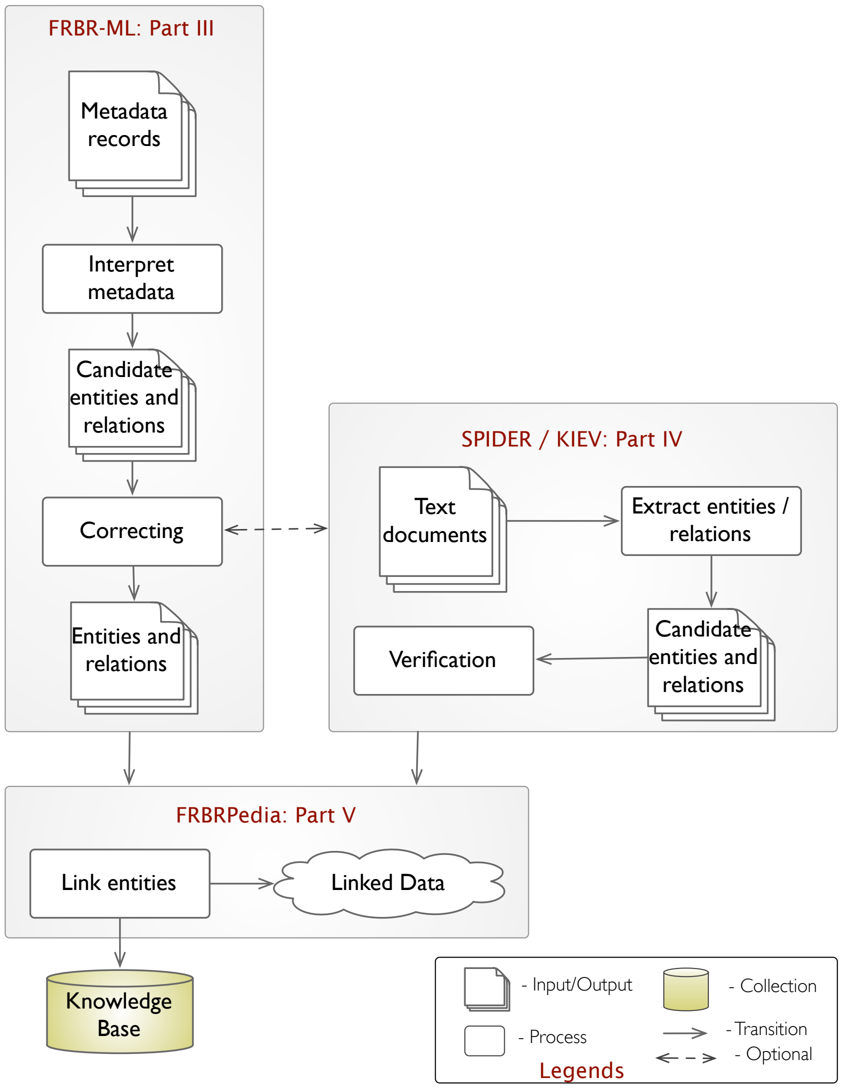


Figure 5.1: Overview of the Generic Framework proposed in this Thesis.

- **Using metadata**
  - **Interpreting metadata**; as described in Chapter 4, metadata formats include descriptive information about entities and relations. By interpreting metadata, the knowledge about entities, their attributes and relations are made explicit for machine interpretation.
  - **Correction**; the output of interpretation consists of distinct set of entities and relations. Correction is an important subsequent step that exploits the use of enhancement techniques to improve the results.
- **Using text**
  - **Extracting Entities and Relations**; natural language documents are another source of potentially valuable information and the extraction step enables discovering entities and relations from such sources. This step serves not only to supplement and complement the metadata extraction to discover missing and/or ambiguous information, but it enables continuous and independent knowledge extraction from text.
  - **Verification**; in order to verify and enrich the extracted entities, the verification step is proposed based on machine learning classification and linking techniques.
- **Linking**; the final step is linking which discovers corresponding LOD entities across the different data sources on the LOD.

5.2 Using metadata

Metadata, especially from the cultural heritage domain, is often regarded as a valuable source of semantic information. The various parts of the approach for interpreting and extracting this semantic information from existing metadata are discussed below.

5.2.1 Interpreting Metadata

The main challenge of interpreting existing metadata, which was once created using different standard or ad hoc schemas is to transform this data into reusable semantic data. This is a complex problem; on one hand, this is a syntactic problem that can be solved by transforming to formats that are compatible with the tools and services used for semantic aware services. On the other hand, this is a semantic problem. Simply transforming from one format to another does not automatically enable semantic interoperability and existing metadata often needs to be reinterpreted.

The basic interpretation of metadata involves identifying and recognizing entities as well as extracting those entities and relations among them using sound conceptual models such as FRBR. Each metadata record can be seen as self-contained universe of entities, attributes and relationships [2] without any dependencies to other records. A generic approach to interpretation will typically include the following steps:

- Identifying entities;
- Discovering relations between entities;
- Extracting attributes of entities;
- Merging the output.

Identifying entities

The initial step in interpreting metadata record is to recognize and extract the correct set of entities described in each record including the type of entity. This process typically consists of using specific metadata elements (fields) that are indicative of unique entities. In the context of MARC records, this will for instance include using titles and names for entities of type *work*. Thanks to logical grouping of data in records encoded with MARC, certain fields indicate specific FRBR entity types along with what kind of role a particular entity plays. For example, an entity may be of type *person* while having either *creator* role or be the *subject* of the *work* described in the record.

Figure 5.2 illustrates a MARC record (tags and codes are colored gray while values are in blue). From this example, the field 100 indicates that “Conrad, Joseph” is an entity of type person with the creator role while the presence of field 240 signifies the title of the original work entity.

```
041  $aeng
100  $aConrad, Joseph,$d1857-1924.
240  $aHeart of darkness
245  $aMørkets hjerte $cJoseph Conrad ; oversatt av
Bjørg Hawthorn ; med forord av Jakob Lothe
$helektronisk ressurs
250  $aNy utg.
260  $aOslo. $bKagge$c2010
300   $a140 s.
```

Figure 5.2: A Simple Example of a MARC Record.

Entities can be identified using a set of rules, which are applied for each type of entity. The rule will typically be applied if an entity of specific type is present in the record. Using the example in Figure 5.2, a rule can indicate that the abstract intellectual work entity is different from the actual publication described in the record because of the presence of fields 240 and 245 and the difference in values. At the lower level implementation aspect, this can be solved by accepting a set of records in an XML format and applying a series of XSLT transformations.

Discovering relationships

Relationships between entities can be established either based on implicit or explicit information. The implicit information is the one based on the roles entities have in the record. Back to our running example, the presence of fields 100 and 240 indicates the relationship “isCreatedBy” from type person to entity type work. The use of relator codes, indicators and field linking as the explicit information can also be used to discover relations if present in the records. However, such practice greatly varies depending on the origin of the record which makes it difficult to use generic rules that could be applied across the data sources.

The basic assumption in establishing relationships is that there is an explicit definition of what possible type of relationships two entities can have. A set of conditions can be set for when the relationship is identified together with the target entity [1].

Extracting attributes

One of the major challenges of interpreting metadata is to clearly capture attributes that pertain to specific entity. In the context of interpreting MARC records using FRBR model, this becomes even more complicated problem because mappings of attributes of entities defined in the model and the fields and subfields of MARC record greatly depends on the contents of the fields and subfields. Therefore, a dictionary-based lookup may be a possible solution. Since some of the attributes are common for more than one entity described in the record (e.g. titles of work and expression), this attribute can be assigned to both entities.

Merging

A collection of metadata records contains a set of self-contained records. The previously described steps are normally performed for each record which will typically result in a set of most probably duplicate entities. The merging step ensures that the duplicates are merged into a single entity and the final set of interrelated entities is obtained. An important point here is when to decide whether or not two given entities are the same. This issue arises due to the lack of proper identifiers for some of the entities. Therefore, extracted entities must be compared using a common mechanism for identity / key generation (e.g. using the same set of attributes – for work entity title and the creator). This can be achieved with simple string matching technique or using advanced algorithms that use more complex rules for comparison.

Another important aspect is to preserve all information about entities to be merged. Even though given entities are equivalent, the different records descriptions may have been made at different levels of granularity which will result in variations in the set of attributes and relations.

5.2.2 Correction

The output of the interpretation is a set of interrelated entities, their attributes and relations. A common understanding is that the result of an interpretation of entities may include errors due to the metadata encoding practice. In fact, several studies [4, 22, 82, 195] indicate that there is a potential for improving the results of metadata interpretation significantly. In general, the typical problems that may arise are:

- Insufficient or erroneously identified entities and relationships;
- Ambiguous entities the type of which is difficult to infer;
- “blank” entities whose relationship to other entities are hard to extract;

Correcting the results can of course be performed manually if the collection size is manageable. However, automatic methods are often needed for larger collections and we are looking for corrections that can be performed in a generic manner.

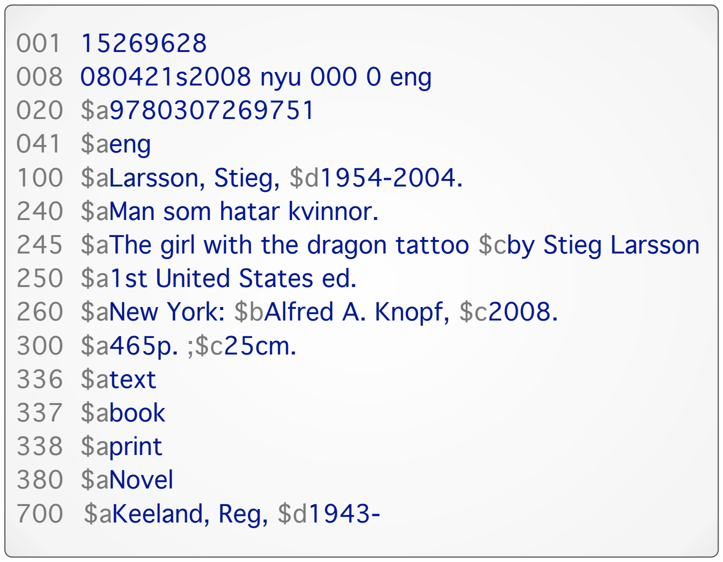
5.3 Using Text

Natural language documents encoded as text are potentially the source of high-quality knowledge [59, 164, 198] and in this thesis they are used as an complementary source to extract entities and their relations.

5.3.1 Extracting Entities and Relations

As illustrated in the Figure 5.1 (marked PART IV) textual documents are used as a complementary source for extracting knowledge.

That is to extract entities and relationships from natural language documents to complement and supplement the results of metadata extraction. The essence of this process is to transform unstructured text expressed in natural language into a structured format that can be exploited by means of machines.
Figure 5.3 depicts an example illustrating a rather ambiguous usage of MARC record in which it is difficult



```
001  15269628
008  080421s2008 nyu 000 0 eng
020  $a9780307269751
041  $aeng
100  $aLarsson, Stieg, $d1954-2004.
240  $aMan som hatar kvinnor.
245  $aThe girl with the dragon tattoo $cby Stieg Larsson
250  $a1st United States ed.
260  $aNew York: $bAlfred A. Knopf, $c2008.
300  $a465p. ;$c25cm.
336  $atext
337  $abook
338  $aprint
380  $aNovel
700  $aKeeland, Reg, $d1943-
```

Figure 5.3: An Example of MARC Record Describing Four Novels.

to automatically infer relationships between entities. This record describes an English translation of the Stieg Larsson's original work "Man som hatar kvinnor" (in the "Millenium" series), entitled "The girl with the dragon tattoo" (*expression*). There are multiple issues with this record. First, the information about the entity in 700 is missing and it is impossible to automatically interpret how "Keeland, Reg" is related to other described entities. Second, the 041 indicates English as the language for the text ($\$a$), but the original language of the work is not known. Finally, "Man som hatar kvinnor" is part of "Millenium" crime novels series and this relationship information is missing. This missing and ambiguous information may be discovered in other records within the same collection, but this is not always the case and additional sources may be used to extract complementary knowledge related to the various entities.

The prototype SPIDER [186] supports extraction of the missing and ambiguous information described above. Additionally, it can run continuously, and enables Web-scale extraction of high-quality relational facts. The extraction process is performed in two steps:

- **Pattern Generation** enables the detection of candidate patterns by using examples or provided entities and generalizes these candidate patterns to obtain patterns for a given type of relationship.
- **Example Generation** exploits the previously generated patterns in order to discover new examples satisfying the type of relationship.

The discovered relational facts are then incorporated into a knowledge base where generated examples and patterns are managed. These examples can be used to maintain the system continuously running and additionally they can be explored by a user too.

5.3.2 Verification

The idea of extraction of examples has been further developed to include a verification step. A verification of the relevance for these examples is performed with the prototype called KIEV [185] using two evidence-based techniques. The former checks if the extracted entities are related with the correct type of relationship using a machine learning classifier. The latter process links both extracted entities of an example to their corresponding entities on the LOD cloud. Once an example is verified, it can be used as a training example to improve the classifier, but also to reinforce the confidence score of a pattern during the discovery process.

5.4 Linking Entities

The motivation for linking entities in the context of this thesis is two-fold. One the one hand, linking is driven by the idea of developing further the verification approach in which an entity is said to be verified when a direct linking to its corresponding entity in a LOD knowledge base is performed. On the other hand, linking the set of entities extracted establishes a bridge to islands of numerous knowledge bases in the LOD cloud thereby enabling support for a proper semantic integration.

5.5 Summary

This chapter provided a high-level overview of the generic framework proposed in this thesis. The two data sources used to extract entities and relations are metadata and natural language documents. The two extraction pipelines include an enhancement step (correction for metadata and verification for textual documents). Finally, enhanced results are linked to LOD and integrated in the semantic knowledge base.

Part III

Interpreting Metadata

6 Entity and Relationship Extraction from Metadata

6.1 Introduction

This chapter starts with presenting the initial step of the framework this thesis proposes. As a reminder from Chapter 5 and specifically the Figure 5.1, this initial step, called FRBR-ML[1], accepts bibliographic information as input to populate a knowledge base. The main research objective of FRBR-ML is to make explicit the entities and relationships that can be extracted from bibliographic records to populate a knowledge base, which consequently enables semantic aware integration and reuse as well as new services based on this information. Specifically, the approach explores the interpretation of bibliographic information using a conceptual model and consists of the following three parts:

- Entity and relationship extraction which basically interprets the entities and information about them described in the bibliographic information and an XML-based representation of extracted entities (Section 6.3);
- Correction and enhancement of semantics of the results of extraction (Section 6.4);
- Metrics that enable assessment of the extraction results (Section 6.5).

The chapter is organized according to this outline. In Section 6.2, we present two use cases, formalize the problem related to the extraction of entities from MARC records

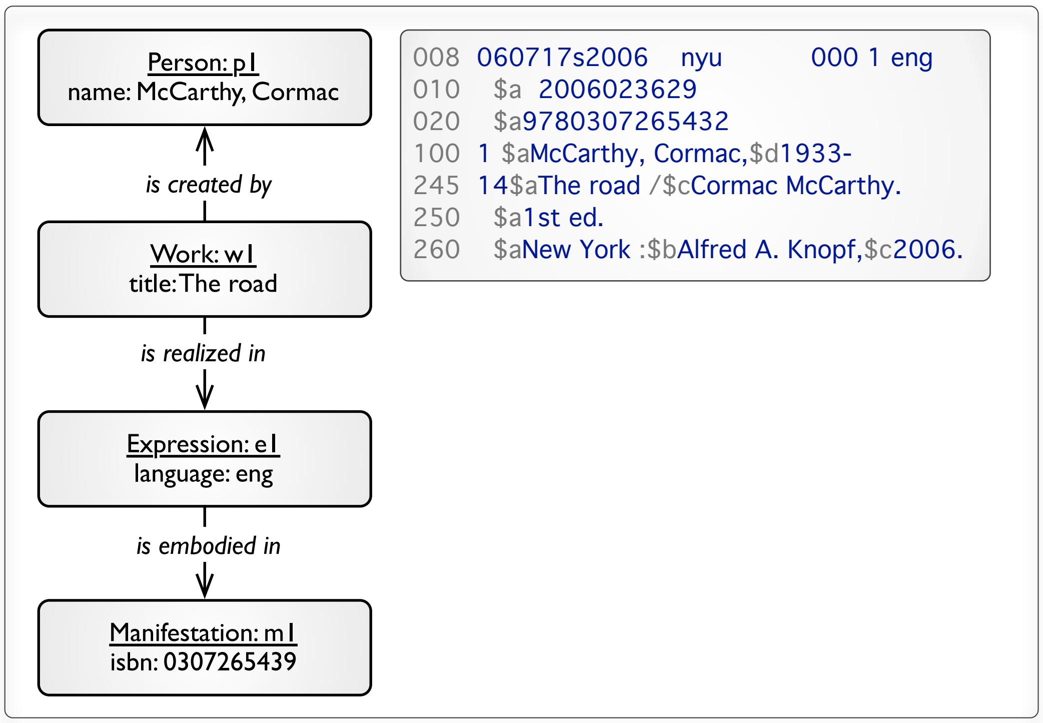
[1]FRBR-ML stands for FRBR in XML.

and provide an overview of FRBR-ML. Next, we detail the different parts of FRBR-ML: representation (Section 6.3), semantic enrichment and correction (Section 6.4), and metrics (Section 6.5). To evaluate our approach, we describe in Section 6.6 experiments performed with the Norwegian national bibliography.

6.2 Overview

6.2.1 Use Cases

As illustrated in Figure 6.1, the simplest example accounting for most of the records is when a *manifestation* is embodying a single *expression* of a *work* [22].



```
008  060717s2006   nyu        000 1 eng
010    $a  2006023629
020    $a9780307265432
100  1 $aMcCarthy, Cormac,$d1933-
245  14$aThe road /$cCormac McCarthy.
250    $a1st ed.
260    $aNew York :$bAlfred A. Knopf,$c2006.
```

Figure 6.1: The Simplest Case: a Manifestation Embodying a Single Expression of a Work Created by a Single Person. The Corresponding MARC Record is Shown on the Right.

In this figure, we see that *a 2006 book* (manifestation), entitled *The Road* (work) has been published in English (expression) by American writer *Cormac McCarthy* (person). The single entities from FRBR groups 1 and 2 are present, thus making the process of recognition and identification easy.

Other categories of records may include several *works* with multiple publishers, translators or these works may be aggregated in various records (e.g. as collection of stories) and/or translated into different languages. Therefore, these category of records are often regarded as the ones that will most benefit from the FRBR model.

Multiple intellectual contributions contained in publications are characterized as distinct intellectual entities [39]. However, not all of these entities are considered equally important: *works* such as cover art, text on the back of the cover etc. are of little interest to most end users. The *expressions* that are of concern to the most users of bibliographic information are those that we consider to be the main content such as the novels in a book containing multiple novels, the essays in an essay collection, the tracks of a music compact disc.

```
100 01$a Sjöwall, Maj, $d 1935-
240 14$a Den vedervärdige mannen från Säffle.$lTyska
245 14$a Das Ekel aus Säffle; $b Verschlossen und
verriegelt : zwei Romane / $cMaj Sjöwall, Per Wahlöö
700 12$aWahlöö, Per, $d1926-1975. $tDen
vedervärdige mannen från Säffle. $lTyska
700 12$aWahlöö, Per, $d1926-1975.
700 01$aSchultz, Eckehard
700 01$aMaass, Hans-Joachim
740 04$aDet slutna rummet
```

Figure 6.2: An Ambiguous MARC Record Describing a Manifestation that Embodies Multiple Expressions of Different Works.

Figure 6.2 describes a such complex case. In this example, German translations of two Swedish novels, we show what kind of entities and relationships are found in existing bibliographic catalogs. These *works*, "Den vedervärdige mannen från Säffle" and "Det slutna rummet", are both originally created collaboratively by two Swedish authors "Per Wahlöö" and "Maj Shöwall", but only the latter is mentioned in the record as the creator (field 100 $a) of the *work*. This may be an unusual and challenging case, but it includes a kind of structure that we believe should be possible to extract from bibliographic records, while keeping the amount of errors in the data at minimum.

As we can see from the above example, added entries (700 fields) are used as additional access points to the bibliographic record to improve the searching for records as well as to present more extensive information about the item described. Such information includes persons and corporate bodies in addition to titles that are relevant as access points. Added entries are recorded using field tags 700, 710, 711, 730 and 740 in MARC 21. With the exception of 740, these titles and names of persons and corporate bodies should – according to the rules – be under authority control. In the context of FRBR, these added entries may reflect quite different aspects of the model, e.g. in this case there are two authors of novel. Other common usages of added entries are to include additional persons such as translators and illustrators.

The type of relationship between a person or corporate body and a resource is indicated by the use of relator codes or terms [139], but the actual use of relator codes greatly varies depending on the local cataloging policy and practice. The titles found as added entries may identify *work* and *expression* entities that are related to the cataloged item in different ways such as the novel upon which a movie is based. In other cases, added entries are used for analytical entries, which can be interpreted as information about the embodiment of additional *expressions* in the *manifestation*.

Finally, the two use cases discussed show that in the simple case it is straightforward to extract the FRBR structure from MARC records, which is the typical in most records. In the second complex use case, the result of interpretation is bound to errors because of missing and ambiguous information in the source metadata.

6.2.2 Formal Model

In this section, we formalize the problem of extracting entities described in MARC records and representing them in a knowledge base powered by FRBR-ML. We have a collection of records $\mathcal{R}$ regardless of representation form. A record $r \in \mathcal{R}$ is composed of a set of properties $\mathcal{P}$, i.e.:

$$\forall r \in \mathcal{R},\ r = <\mathcal{P}>$$

Each property $p \in \mathcal{P}$ is represented by a name and a value. An example of a property is a MARC datafield (245 \$a, The Fellowship of the Ring) where 245 is a datafield tag, \$*a* is a subfield code, together forming a property for the *main title* of a bibliographic record. A subset of $\mathcal{P}$ provides a description of an entity. For instance, a *manifestation* entity found in a record describing “The Fellowship of the Ring” book by Tolkien, may be described by the properties title (245 \$a, The Fellowship of the Ring) and manifestation identifier[2] (020 \$a,0618574948) . A specific property id uniquely identifies the record r.

A record describes one or more publications and therefore can be seen as an abstraction of a set of entities $\mathcal{E}$, such as a *manifestation*, *expression*, *work* and related *persons*. Although entities are present in MARC records, they can only be extracted based on our interpretation of their properties and relationships. In contrast to that, the entities in FRBR are clearly defined by the model. Finally, these entities are related to each other through a set of relationships $\mathcal{L}$, such that:

$$\forall l \in \mathcal{L}, \quad e_1, e_2 \in \mathcal{E}, \quad l : e_1 \times e_2$$

6.2.3 Extraction in FRBR-ML

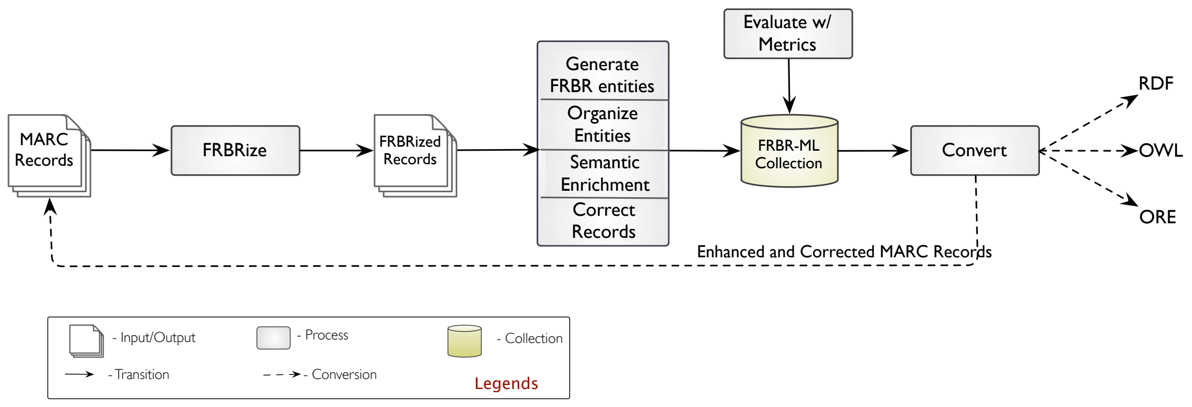


Figure 6.3: Workflow in FRBR-ML.

Although memory institutions are interested in semantic formats, the transition cannot be automatically achieved due to the complexity of adding well-defined semantics to a large body of existing legacy data. Indeed, ontology languages such as OWL/RDF have a great degree of machine readability which enables automatic reasoning, but converting legacy data using the correct semantics of these languages is still an unsolved challenge [61, 173]. Consequently, there is a need for an intermediary format to ensure a seamless transition that enables both legacy data and semantic formats to coexist. Needless to say, the format should enable the exchange of records between different applications or services. Another challenge of the MARC format is that the

[2]In this case an ISBN number of the published book

specific meaning of a datafield often is contextual and not explicitly defined, thus making it difficult to automatically process information. For example, if there is a 740 *Added Entry* field we know that this is a title, but unless the second indicator has the value 2, we do not know in what way this title is related to the work(s) described in the record.

FRBR-ML addresses the issues of interoperability and lack of semantics. Figure 6.3 depicts the workflow of the process of transforming MARC records into FRBR-ML format. The process starts with the conversion of MARC records using an extended version of the FRBRizer tool [2, 5].

This tool performs the conversion of MARC records into a set of FRBRized records using a set of pre-defined rules and a series of XSLT transformations. An example of the output of the conversion is shown in Figure 6.4.

```
<record label="Work" type="C001" id="76d632d21tr0w1">
   <datafield tag="245" ind1="1" ind2="0">
      <subfield code="a">Tolkien</subfield>
   </datafield>
   <relationship type="P2001" label="is realized through"
      href="81d632d21tr0y3"/>
   <relationship type="P2009" label="is created by"
      href="98d632d21tr034"/>
   <relationship type="P2033" label="has as subject (person)"
      href="45d632d21tr012"/>
   <keyvalue>whitemichael#tolkien#</keyvalue>
   ...
</record>
```

Figure 6.4: A Fragment of the Output of the FRBRizer Tool.

The next step is to convert the FRBRized records into our intermediary format. Contrary to the output of the FRBRizer tool, our format aims at reflecting the full structure and semantics conveyed by the FRBR model in a more readable way. Similarly to [166], we strive to maintain the human readability while allowing an easy transformation into a machine interpretable form. In addition, the FRBR-ML format uses the FRBR vocabulary to promote simplicity and understandability, since libraries and memory institutions are increasingly interested in adopting FRBR in the long term. Once our tool has converted the records into structured FRBR entities, it enriches them by querying external resources and it performs correction of properties. The result, stored in the FRBR-ML format, can further be converted either to RDF, OWL, ORE or back to enhanced MARC records. Our intention is to support a two-way interoperability between systems that manage resources in a variety of formats. Finally, to identify interesting properties of our format, we have defined three metrics to measure the loss of information, the amount of redundancies and the percentage of semantic enrichment.

In the next sections, we present in more detail the different parts of FRBR-ML: representation of the format and interoperability (Section 6.3), semantic enrichment (Section 6.4), and design metrics (Section 6.5).

6.3 Representation

FRBR-ML features an entity-oriented representation that is comparable to new knowledge representation frameworks based on RDF. However, for managing this information we will argue that there is a need for a simple and understandable format. The end-user of FRBR-ML would be people who manage cultural heritage and therefore used to managing record-based information. Our approach allows us to bridge the gap between record and resource based representations. Furthermore, using an XML based representation would enable a lower barrier for understandability (Section 6.3.1), but at the same time FRBR-ML can easily store the entities as RDF triples in the knowledge base as well.

The second feature of FRBR-ML is that it represents MARC-based information with a clear structure. By clear structure, we mean that the information is encoded following the FRBR specifications. This organized representation should also minimize the loss of data and redundancies (Section 6.3.2).

The last point is related to exchange of records. The representation should ensure compatibility with both record-oriented and semantic formats (Section 6.3.3).

6.3.1 Entity-oriented Representation

To facilitate understandability, we are adapting the entity-oriented representation to the record-oriented, embedding the FRBR entities in the records with their respective FRBR semantic FRBR type. Instead of representing FRBR entities with records, they are embedded within their respective semantic FRBR type. The naming convention for semantic types in our representation follows the FRBR model, making navigation and browsing simpler.

The FRBRized collection is represented as a set of XML documents. In these documents, each root element labeled "collection" has *work*, *expression*, *manifestation*, *person* and *corporate bodies* as child elements. Each of these elements contains properties represented by datafields and control fields embedded under a semantic element μ. This semantic element is provided by a *map* function that takes as an input a property p:

$$\mu := map(p)$$

If the function returns the same semantic element for several properties, all of them are grouped under this semantic element. This function will be discussed in more details in Section 6.4.1.

In FRBR-ML, entities are linked by relationships. These relationships are not transitive, as defined in the FRBR model. However, all of them are bidirectional. For example, a person who created a work does not mean that (s)he created a manifestation as well. But a person has a relationship *isCreatorOf* to work and the work an inverse relationship *isCreatedBy* to that person.

Figure 6.5 depicts a fragment of a *manifestation* entity. Like any element representing a FRBR entity, the manifestation element can contain a set of attributes, a set of MARC fields and a set of semantic elements. These semantic elements include values of properties of a FRBR manifestation entity as well as semantically enriched information (see Section 6.4.1). A similar representation pattern is used to describe *work*, *expression* and *person* entities, as shown in the complete schema included in the Appendix 13.

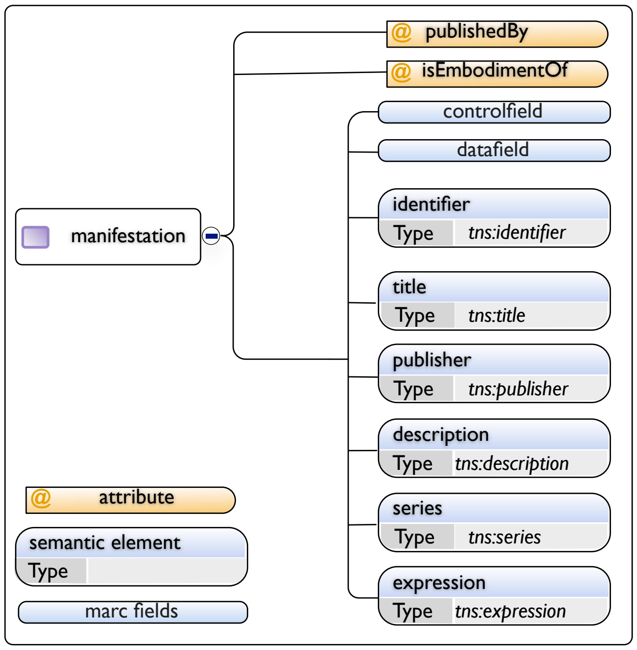


Figure 6.5: A Fragment of Schema for the Manifestation Entity.

Since the same entity may occur in multiple records, there is a need for discovering equivalent entities.

This is mainly intended to solve the redundancy problem in the output by merging entities that are very likely to be the same. As pointed out in the works of others [132], more flexible techniques are needed to determine if two entities represent the same real-world entity and should be merged into one. However, we believe that it is important to have a complete and correct interpretation of the entities and relationships described in the record as a first stage, then followed by matching techniques in combination with verification and correction techniques in order to improve the quality of the results. Our approach is based on comparison of entities by the use of key descriptive information that is unique for each entity within the collection based on selected property values. We generate identifiers for these entities by calculating MD5 hash for their corresponding key values. A post-processing step merges identical entities based on this hash value, which eliminates duplicate entities.

Having finalized the element level discussion, we detail how we organize entities.

6.3.2 Structural Organization

Expressing relationships in a well-defined manner between entities in XML is one of the important tasks to avoid duplication and loss of data. Furthermore, it is a first step to introduce semantics in existing data. In the rest of this section, we study the different possibilities to represent relationships between entities, namely **hierarchical** and **referencing**. Finally, we describe our solution, a **hybrid** method that combines advantages of the aforementioned ones.

Hierarchical Method

As the name implies, this method enables the expression of entities and their respective relationships with hierarchical organization (also called parent-child relationship) [80].

One of the first advantages is an increased readability with implicit semantics between FRBR entities. For instance, a *manifestation* that includes a child element *expression* has an implicit "isEmbodimentOf" relationship to that expression (the inverse of *isEmbodiedIn*). Other advantages are compactness and proximity of data, which enable faster processing. On the other hand, this method suffers from possible infinite loop (e.g., an entity having a relationship to an ancestor entity in the tree). It is also insufficient to represent more than one relationship type under the same parent. Another disadvantage of this approach is data duplication. This issue is present when there are several relationship references to the entity.

Figure 6.6 depicts an example of this hierarchical method in which all FRBR entities are represented as nested elements with the manifestation as the root. Having the manifestation as the topmost node in a hierarchical representation may seem to contradict the FRBR model that has *work* as the most abstract item, but the model does not describe any particular arrangement of the entities.

```
<manifestation id="m1">
   <expression id="e1">
      <work id="w1">
        <person id="p1">
            ...
        </person>
         ...
      </work>
      ...
   </expression>
   ...
</manifestation>
```

Figure 6.6: An Example of a Hierarchical Method.

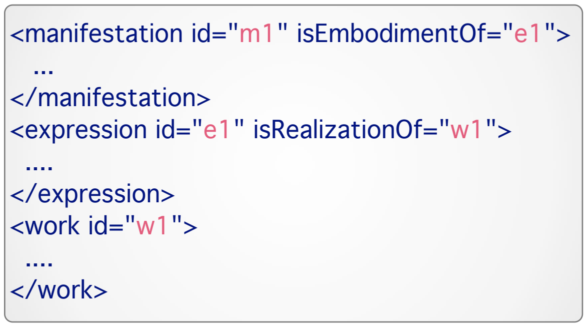
Using the manifestation as root element corresponds to the traditional way of organizing metadata into a record for each of the described items. This arrangement is also tailored the creation of metadata which typically is a process of describing the entities and relationships that makes up the description of a specific manifestation. It is also the arrangement that would lead to the most evenly distribution of entities in records as the maximum number of expressions a manifestation embodies is rather low compared to the inverse case. Other hierarchical arrangements would be better suited for other situations such as using *work* as the topmost level when presenting search results to users who want to explore what is available for a specific author, but such representations can easily be created on the fly when processing a collection of records.

Reference Method

We can also employ the reference method when expressing relationships between entities [112]. This method is based on the usage of ID/IDREF similarly to URI in RDF.

There are different techniques for referencing entities. A first technique is *dynamic typing*, i.e., the type of the relationship is specified by an attribute *type* of a given *relationship* element. It provides a greater flexibility when one needs to add new relationships.

The second technique is to have a set of *strongly* or *statically* typed predefined relationship types which are represented as elements. Although this technique eliminates the readability weakness of the first technique, the problems arise when there is a need to define new types of relationship. No matter which technique is used, the reference method avoids duplication and it provides a better support for updating data. Furthermore, it reduces the size of the XML document. However, related entities are stored in a loosely coupled manner which means it is less efficient in terms of processing.



```
<manifestation id="m1" isEmbodimentOf="e1">
  ...
</manifestation>
<expression id="e1" isRealizationOf="w1">
  ....
</expression>
<work id="w1">
  ....
</work>
```

Figure 6.7: An Example of a Referencing Method.

Indeed, entities are spread across the document and accessing them requires more effort. As illustrated in Figure 6.7, all entities are stored under the root element with this reference method. We notice that the manifestation is linked to an expression entity with an *isEmbodimentOf* reference.

Hybrid Method

The main drawbacks of the hierarchical approach are data duplication and the constraint related to expressing single relationship type between entities. On the other hand, the reference method is less efficient in terms of processing. Our hybrid approach combines the representation of both methods. In one case, an entity can be stored hierarchically under its related entity taking advantage of proximity, readability and efficient processing. But under different conditions, an entity is stored using the reference method to avoid duplication if it appears several times in the collection. For example, a particular embodiment of expression can be represented as either child element (hierarchical) or by attribute (referencing method).

The relationships between entities are represented using the strongly typed method (see Section 6.3.2). This method has been chosen due to the fact that a set of predefined relationship types are specified in the FRBR model.

The process of deciding which representation method to use for a given entity is shown in Algorithm 1. This process takes as an input a set of records $\mathcal{R}$. We initialize a set of constraints λ that should be respected for hierarchical representation (line 3). A constraint is a tuple containing two entity types and a relationship type, indicating that this relationship is allowed. All records are analyzed and for each of them we extract their entities (line 5). For each entity, the decision to represent it hierarchically or referentially is made by the *decide* function (line 7) before adding it to the set of entities $\mathcal{E}$.

Algorithm 1 Hybrid representation decision.

```
1: Input: Set of Records $\mathcal{R}$
2: Output: Set of Entities $\mathcal{E}$
3: function PROCESS($x$)
4:     $\mathcal{E} \leftarrow \emptyset$
5:     $\lambda \leftarrow constraint\_definitions()$
6:     for all $r \in \mathcal{R}$ do
7:         $E = extract\_entities(r)$
8:         for all $e \in E$ do
9:             $decide(e)$
10:            $\mathcal{E} \leftarrow e$
11:        end for
12:    end for
13: end function
14: function DECIDE($e$)
15:    $T = get\_relation\_types(e)$
16:    $add\_to\_stack(e)$
17:    for all $t \in \Delta T$ do
18:        $C = find\_conn(e, t)$
19:        for all $c \in C$ do
20:            $\delta = violates\_constraint(e, c, t, \lambda)$
21:            if not $\delta$ and not $in\_stack(c)$ then
22:                $hierarchical\_rep(c)$
23:                $decide(c)$
24:            else
25:                $referencing(c)$
26:                if not $in\_stack(c)$ then
27:                    $decide(c)$
28:                end if
29:            end if
30:        end for
31:    end for
32: end function
```

In the $decide$ function, we obtain the set of relation types for the given entity e (line 13). To avoid infinite loop, we add this entity to the current stack of entities (line 14). ΔT contains all distinct relation types t of e. The next step analyzes each of these relation types to discover the set of entities C linked to e with relation type t (line 16). The decision to represent an entity hierarchically depends on whether the related entities violate any constraints (line 18) as well as the presence of the entity in the current stack. If these conditions are not met, the entity is represented using the reference method (line 23). The $decide$ function recursively iterates over entities (lines 21 and 25). The set of extracted entities with typed relationships is then stored as a FRBR-ML knowledge base in an XML format.

The hybrid algorithm is flexible due to the set of constraints. By default, our set of constraints includes the basic relationships defined in the FRBR model. However, one may define additional constraints to meet specific requirements.

6.3.3 Exchange of Records

The output of the final transformation is a set of entities described with clear structure as well as typed relationships. These entities are assigned the same identifiers as those generated by the FRBRizer tool. They have relationships to other entities in the same collection by using either referencing or hierarchical method. This output can be converted to other representation formats such as RDF/OWL as well as more domain specific formats such as MARC. These conversions are performed by a series of XSLT transformations.

We begin with describing the transformation to RDF and OWL. The RDF conversion is represented using the vocabulary provided in [62]. This vocabulary is an expression of the concepts and relations described in the FRBR model in RDF. Properties follow naming convention such as *Et dukkehjem has_realization A doll's house*. In this vocabulary, most properties are paired with an inverse, e.g. *embodiment/isEmbodimentOf*. The use of synthetic superclasses for ranges and domains are discarded and we only employ concrete entity types. In other words, we relate *works* directly to *person/-corporate body* rather than to ResponsibleEntities. Furthermore, this vocabulary uses OWL-DL to provide constraints. As the vocabulary lacks support for attributes of the entities, we additionally use the FRBRer model vocabulary [70] to describe attributes. As for OWL, we generate an OWL instance via XSLT transformation for each XML document validated by the schema. The FRBR entities *work, expression, manifestation,* *person* etc. are declared as *owl:Class* and *owl:ObjectProperties* to specify elements and attributes of the entities. While we specify the bidirectional relationships for RDF, we can make use of *owl:inverseOf* construct for OWL transformation.

An extension that we have implemented in the FRBRizer tool deals with the storage of the MARC control field 001 in order to ensure a correct transformation back to MARC. The control field 001 contains the unique control number assigned by the organization creating, using, or distributing the record. All datafields and control fields have an element <*mid*> specifying the original control field 001. During the transformation back to MARC, we can use this information to correctly construct the original MARC record since some records might have been merged. The process of transformation back to MARC starts at the manifestation level. We collect all datafields and control fields that are pertinent to the record, i.e., those with the same <*mid*> value. That is, we traverse the conceptual tree of entities from *manifestation* to *person/corporate body*, *expression*, and *work*. represented as MARCXML.

As for the transformation to ORE, we are concerned with the description of aggregations of entities. For each record related entities identified by the previously mentioned control field 001, we can collect all the related entities which forms an aggregation. For this representation, we use the same RDF vocabulary.

As a summary, our representation ensures compatibility with record-oriented formats, such as MARC 21 but also with strongly semantic formats such as RDF and OWL.

6.4 Semantics

On the semantic aspect, FRBR-ML provides a balanced degree of semantics (Section 6.4.1). Furthermore, FRBR-ML includes a correction process to improve and disambiguate the information found in original records (Section 6.4.2).

6.4.1 Semantic Enrichment

The task of semantic enrichment is crucial in our process. On one hand, all information in a MARC record may not have a clear semantic. For instance, the datafields tagged with values 700-740 are added entries that often can be difficult to interpret the precise meaning of and associate to the correct entity. On the other hand, new formats such as RDF or OWL include a well-defined semantic to enable reasoning or complex querying. Consequently, a conversion from MARC to RDF needs to involve the identification of the precise meaning all information in MARC records. This is a complex problem and in FRBR-ML, we aim to solve problems related to the semantics of attributes, entities and relationships. For the attributes we use a *map* function to look up the correct label of datafields (e.g., *title* label for the datafield 245). More formally, the *map* function uses either a dictionary-based technique or a knowledge-based technique to discover the semantic element of a given property p with name p_n and value p_v:

$$map(p) = \begin{cases} dict(p_n) & \iff \exists\{dict(p_n)\} \\ knowl(p_v) & otherwise \end{cases}$$

For entities and relationships that the interpretation process is unable to interpret the exact meaning of, we primarily use the knowledge based technique to discover the correct type.

Dictionary-based Matching

Based on the MARC specification for the format we are processing, we build a dictionary of corresponding elements between the datafield tag, the subfield value and the semantic element. A fragment of this table is shown in Table 6.1. Discovering the semantic element requires a lookup in the table by decomposing a property name into a datafield tag and a subfield code. If this index pair is not found in the table or if there are multiple entries (which would indicate that there is more than one possible semantics label), it means that obtaining the semantic element for that particular index pair depends on the value of the index pair. Thus, we need a refined technique based on external resources to discover the semantic element for this pair.

Knowledge-based Matching

The lack of semantics is preponderant with many entries found in MARC records. Persons identified have different roles and should be related to works, expressions

Property Name		Semantic
Data Field Tag	**Subfield Code**	**Element**
245	-	Title Info
245	a	Title
100	a	Name (Personal)
...	...	...

Table 6.1: A Fragment of our Dictionary

and manifestations in different ways. If relator codes are missing, we have to identify the appropriate type and target and endpoints of the relationship. Titles in added entries may identify distinct *works* or be related to the *expression* or *manifestation* entities and we need to find out what type of entity they relate to. Some fields may have unidentified or ambiguous semantics and we need to interpret the exact meaning of the value. The problem of discovering the correct type of an entity, relationship or attribute value is very complex. However, we advocate that it is possible to discover the semantic type represented by this entry value, e.g., a writer or a location. To fulfill this goal, we rely on external resources for, mainly semantic knowledge bases such as *DBpedia*, *Freebase* or *OpenCyc*.

Our task is very similar to entity ranking, which consists of discovering a Linked Open Data (LOD) entity's main page. The LOD cloud refers to interconnected knowledge bases, which can be seen as the foundation of the LOD vision[3]. Many *works* have been dedicated entity ranking, such as [172, 192] to name a few. In addition, two yearly challenges have an entity ranking track: *Initiative for the Evaluation of XML Retrieval* [87] and *Text Retrieval Conference* [194]. In our context, we do not have as much information as in entity ranking. Thus, we propose to discover the correct LOD entity by using **aliases**, i.e., alternatives forms of an entity's label. *J._R._R._Tolkien* is the label of the DBpedia entity representing the famous writer of Lord of The Rings, and a few of its aliases are *John_Ronald_Reuel_Tolkien*, *J.R.R_Tolkien* and *Tolkien,_J._R._R.*. These aliases are properties of an entity. For instance, Freebase provides alias for an entity (property *fb:common.topic.alias*) while DBpedia includes redirections (*dbpedia-owl:wikiPageRedirects*). Once the correct entity is discovered, it is possible to obtain its type and use it as semantic element.

More formally, we first normalize the property value p_v, i.e., replacing spaces with underscores, removing extra information in brackets, etc. As a result, we obtain a

[3]http://linkeddata.org, Last accessed January 2013

set of normalized queries $\mathcal{Q}$ for the value p_v. Given a set of knowledge bases $\mathcal{K}$, we send a query $q \in \mathcal{Q}$ against each knowledge base $k \in \mathcal{K}$ to obtain a set of ranked LOD entities. We note t_{qk} the set of result entities returned by the knowledge base k for the query q. The number of results in each set t_{qk} depends on the techniques used to query the knowledge base. Many semantic knowledge bases include various possibilities for retrieving an entity [29]:

- direct access by generating the URI of the entity. In this case, each set t_{qk} contains 0 or 1 entity;
- querying SPARQL endpoints. With this technique, the number of returned entities varies from 0 to the size of the knowledge base;
- querying a search engine or an API, which returns a set t_{qk} with any number of entities as well.

Since the knowledge bases on LOD are interrelated (property *owl:sameAs*), the same entity may appear in different result sets. We first detect which entities are identical thanks to this OWL property. We define φ as the size of the largest result set t_{qk}. To discover the correct entity, the idea is to apply statistics against all result sets. We assume that the rankings of the knowledge bases are somehow coherent and that the correct entity should appear in most rankings at the top. We therefore compute a score for a LOD entity x as follows:

$$\forall t_{qk}, score_x = \sum \min(rank(x, t_{qk}), \varphi)$$

In other words, we sum the different ranks of the entity x in each result set. If the entity does not appear in t_{qk}, the size of the largest result set with value φ is added. Finally, the entity with the smallest score is selected and its type is used as semantic element in our format.

6.4.2 Correction Process

The full potential of MARC records is often disregarded, which resulted in records with ambiguous semantics. Enriching with semantic information is not sufficient since it depends on identifiable entities. This is usually the case when a record does not contain enough information about entities. As we saw in the use case in Section 6.2.1, the translator “Hans-Joachim Maass” is not linked to any expressions that he has contributed to. Consequently, we have two tasks to accomplish: (i) identifying the type of entity to which "Hans-Joachim Maass" is linked and (ii) finding the correct relationship to the entity in the record. To achieve this goal, there are different strategies that can be employed:

- **intra collection search**. The publication may appear several times in different records of the same collection, especially for collection of *works* bundled as a single *manifestation*. In that case, we can analyze such related records to find the correct relationship.
- **inter collection search**. We use external services to perform a search for the entity. This is achieved querying z39.50[4] or SRU/SRW[5] endpoints. With the use of a library catalog supporting one of the above protocols, we can find the lacking information.
- searching the **LOD cloud**. To discover the correct entity in LOD, we can use the same method as in Section 6.4.1. Then we can analyze each pair of property/values of this entity to detect an eventual relationship of the unknown entity.

If the relationship is discovered by applying any of the above methods, then we identify and enrich the entity with the relationship. When a conversion back to MARC is performed, this missing information about the entity can be corrected with regards to the initial MARC record. Indeed, an intra-collection search is more efficient to identify and find the correct relationship to the record since no network connection overhead is involved. Additionally, there is a fair chance of a match in the same collection for entities we are looking for. Searching the LOD cloud could provide good results too in terms of efficiency given the knowledge base is queried locally on the same machine[6].

Recall our example of missing information about translator "Hans-Joachim Maass". This record does not contain relator code that identifies the type of entity to which he is linked. We can find this information from the results of the various applied techniques described above. Additionally, we need to find the relationship to an entity. To accomplish this, we search for each identified entity, i.e.,work "Det slutna rummet", and exclude the other entities from the query to reduce the room for misinterpretation. Once we find the record where the two entities appear, then we can assert the

[4]`http://www.loc.gov/z3950/` (Last checked February 2011)
[5]`http://www.loc.gov/standards/sru/` (Last checked February 2011)
[6]DBpedia or Freebase dumps are freely available for download.

relationship based on the information found in this record. Finally, for each identified entity we build a cache in order to make the process of subsequent identification faster.

6.5 Metrics

To evaluate the semantic enrichment and the representation of our format, we have defined different metrics with respect to completeness (no loss of information), minimality (no redundancies) and extension (enriched information). Applying these metrics against our format enables us to demonstrate the weak points and good properties of our approach.

6.5.1 Completeness

The completeness measure aims at detecting the amount of information that can be lost during transformation [19]. To be complete, the transformed records should contain all properties found in the original records. However, a few properties are more important because they are used to identify entities. Thus, we have defined two completeness measures: a quantitative completeness $quant\_comp$ that measures the amount of present properties after transformation; and a qualitative completeness that measures the amount of present entities after transformation. Both metrics are applied between an original collection $\mathcal{R}$ and a transformed one denoted $\mathcal{R}'$.

The quantitative completeness $quant\_comp$ shown in Formula 6.1 checks all the properties of identical records (i.e., based on their identifiers r_{id} and r'_{id}) using the hash value of the property:

$$\forall r \in \mathcal{R} \text{ and } \forall r' \in \mathcal{R}' \text{ such that } r_{id} = r'_{id},$$

$$quant\_comp(\mathcal{R}, \mathcal{R}') = \frac{\sum \frac{|\mathcal{P}_r \cap \mathcal{P}_{r'}|}{|\mathcal{P}_r|}}{|\mathcal{R}|} \tag{6.1}$$

We define qualitative completeness as an indication of the degree the conversion process is able to interpret all possible entities. We take into account the key properties that identify an entity. For instance, creators are identified by the fields 100 (personal name main entry) and 700 (personal name added entry). Therefore, we define the qualitative completeness formula between the two collections $\mathcal{R}$ and $\mathcal{R}'$ as follows:

$$qual\_comp(\mathcal{R},\mathcal{R}') = \frac{\sum \frac{|\mathcal{E}_r \cap \mathcal{E}_{r'}|}{|\mathcal{E}_r|}}{|\mathcal{R}|} \tag{6.2}$$

However, this metric does not say anything about correctness of the results. Correctness can be evaluated by manual inspection for small collections, but for larger collections we have to use more automatic verification techniques. We discuss this verification aspect in the experiments section.

Both completeness metrics are in the range $[0,1]$, with a 1 value meaning that the transformed records are totally complete in terms of properties and entities.

6.5.2 Redundancy

The minimality metric checks the non-existence of redundant information. In [57], minimality is defined as the percentage of extra information in a generated integrated schema. This definition does not hold in our context, since our FRBR-ML approach includes a process for enriching the original records with semantics. Consequently, we propose to measure the amount of redundant information with a first metric called redundancy. This redundant information is mainly due to the aggregation of data from similar records, e.g., with rules in the FRBRizer or during the correction process.

To detect a redundant property in a transformed collection of records, we define a set $\Delta\mathcal{P}' \subseteq \mathcal{P}'$ which contains all unique properties (according to their name and value). The following constraint is therefore respected for $\Delta\mathcal{P}'$:

$$\forall p_1 \in \mathcal{P}',\ p_1 \in \Delta\mathcal{P}' \iff \nexists\, p_2 \in \Delta\mathcal{P}' \colon \{p_1 = p_2\}$$

The individual redundancy of a record is the ratio between the size of the sets $\Delta\mathcal{P}'$ and $\mathcal{P}'$. The properties are compared using their hash values. To measure the redundancy of a transformed collection $\mathcal{R}'$, we sum the individual redundancies of each record and we normalize the results by the total number of records:

$$redundancy(\mathcal{R}') = \frac{\sum \frac{|\Delta\mathcal{P}'_{r'}|}{|\mathcal{P}'_{r'}|}}{|\mathcal{P}'|}$$

The redundancy metric is in the range $[0, 1]$, with a 0 value meaning that the new set of records does not contain any duplicate information compared to the original ones.

6.5.3 Extension

In database quality or model engineering domains, extra information may not be seen as positive. However, it enables the disambiguation and enrichment of the original data in our context. Thus, our last measure, called **quantitative extension**, computes the percentage of extra information added as a result of our enrichment process. To compute this number, we need the same Δ function which contains all unique elements of a set. Indeed, the metric would be biased if it uses redundant information. The amount of enriched information between an original record r and its transformation r' equals $|\Delta\mathcal{P}_{r'}| - |\Delta\mathcal{P}_r \cap \Delta\mathcal{P}_{r'}|$. We generalize this formula between two collections by comparing identical records:

$$quant\_extension(\mathcal{R}, \mathcal{R}') = \frac{\sum \frac{|\Delta\mathcal{P}_{r'}| - |\Delta\mathcal{P}_r \cap \Delta\mathcal{P}_{r'}|}{|\Delta\mathcal{P}_r|}}{|\mathcal{R}|}$$

The extension metric is in the range $[0, +\infty]$, with a 0 value meaning that the new records have not been enriched at all. Note that we cannot automatically assess the quality of the enrichment process due to the following reasons. For one, the human judgement is required to validate given that the ground truth is lacking. Furthermore, the enrichment process consists of two steps (the addition of a semantic type and the verification/correction of relations) which differently influence the quality of extension. However, we perform a manual evaluation of the quality of enrichment described in the next section.

6.6 Experiments

This section deals with the evaluation of our approach. We begin with the description of the NORBOK dataset used for evaluation purposes. The format included in FRBR-ML is evaluated to check how it fulfills important design criteria such as *completeness*, *redundancy* and *extension*. The next part is dedicated to semantics, i.e. we measure the error rate caused by the enrichment and correction processes. Finally, we detail a complex use case which is commonly found in the library collections.

6.6.1 Dataset and Evaluation Protocol

Experiments were performed on a dataset provided by the Norwegian National Library. More specifically, this dataset is a national bibliography containing **449.063** records grouped into these types of materials:

- books, pamphlets, monographs in series, maps, computerized documents (including e-books), regardless of language;
- audio books in various languages (published in Norway from 1992);
- foreign translations of works by Norwegians (from 1978);
- foreign works about Norway and Norwegian conditions;
- complete coverage from the 1921 publication of Norwegian releases in 1978 for overseas - but a large number of older works are included.

The whole collection is stored in the NORMARC format, a dialect of MARC used in Norway. We have run the enhanced version of the FRBRizer tool to identify the FRBR entities in the records. This means that we have created new conversion rules in FRBRizer to take into account all fields in the initial records. Next, we transform the data into our format and enrich it. During the transformation process described in Sections 6.3.1 and 6.3.2, we have used a set of constraints that includes the basic FRBR relationships. The enrichment process relies on two semantic knowledge bases, *DBpedia* and *Freebase*. These bases have been queried using the three mentioned techniques in Section 6.4.1, i.e., a direct access by building an URI, queries over a search engine[7] or API[8] and with the SPARQL language[9]. Note that our approach is not limited to these bases and that we could have used other sources such as *OpenCyc*. However, DBpedia can be seen as the center of the LOD cloud by containing the largest number of connections to other data sources, and it is strongly connected with Freebase[10]. In the correction process, we have sequentially applied the three proposed

[7] `http://dbpedia.org/lookup` (Last checked February 2011)
[8] `wiki.freebase.com/wiki/Search` (Last checked February 2011)
[9] DBpedia only, we did not query Freebase with MLQ language.
[10] 2.4 million links in November 2008

techniques of Section 6.4.2. When a search in the local database does not provide any results, we perform an inter-collection search by using the *z39.50* protocol. If we are still unsuccessful, the correction tries to discover the missing information on the Linked Open Data cloud, namely the search services provided by DBpedia and Freebase. Based on this experiment protocol, we now detail the interesting results of our approach.

6.6.2 Quantitative Evaluation

In this first experiment, the goal is to detect the good points and weaknesses of our format in terms of design. Thus, we have converted the collection stored in the FRBR-ML format back to MARC. We are able to compare the resulting MARC records to the original ones with different quantitative criteria.

Merging Results

First, we analyze the results of the merging process detailed in Section 6.3.1. Table 6.2 provides a summary of the results for each entity type. We notice that our merging process enables us to remove 15 – 25% of duplicate entities in Group 1 (*work*, *expression*,*manifestation*). Furthermore, the set of Group 2 entities (*person*, *corporate body*) initially contains a fair amount of duplicates (respectively 70% and 80%). As a consequence, the dataset is cleansed, and the data processing (query, search) is accelerated.

Entity type	**# of Entities before merging**	**# of Entities after merging**	**Ratio**
Work	564379	422475	25%
Expression	451057	384954	15%
Manifestation	562838	465211	18%
Person	689957	207536	70%
Corporate Body	189532	38701	80%

Table 6.2: Merging Results.

Quantitative Completeness, Redundancy and Extension

Next, we apply the quantitative metrics defined in Section 6.5: completeness, redundancy and extension. Table 6.3 shows the values achieved for properties, i.e., MARC control fields and datafields. We first observe that our format does not lose much data (both completeness values above 90%). The reason is because we sometimes change the datafield tags during the conversion back to our alternative MARC representation. As an example, 700 fields in the source record may include both the title of a work and the name of the person that is the author. Since we interpret these as separate entities we use 740 fields instead for the title in the work record.

Dealing with the amount of redundancies, it appears that the format tends to duplicate around 25% of the control and datafields. As explained in Section 6.3.2, the hierarchical representation involves redundancies. Our hybrid algorithm may select this representation for relationship types which have not been specified in the set of constraint, thus leading to duplicates. However, we insist on the fact that these duplicates could be easily deleted with a simple script after the conversion back to MARC.

Finally, we check the amount of semantic information which has been added. Namely, 8% of the datafields have been enriched with regards to the initial ones. Note that this amount only includes enriched fields as a result of the correction process (and not the types added as semantic elements). As expected, control fields have not been extended since their main purpose is to provide general information about a record.

To summarize, our format ensures a correct completeness. It does not guarantee a minimum number of redundancies but the duplicate properties can be removed with a post-conversion process. The semantic enrichment is also propagated back to the converted MARC records.

	Property	
	Control Fields	**Data Fields**
Completeness	97%	93%
Redundancy	25%	28%
Extension	0%	8%

Table 6.3: Quantitative Completeness, Redundancy and Extension for the NORBOK Collection.

6.6.3 Qualitative Evaluation

The quantitative metrics do not provide insight about the quality. In this section, we present results on qualitative completeness and qualitative extension.

Qualitative Completeness

We have computed the following results for the NORBOK collection based on the qualitative completeness metric described in Section 6.5.1. Recall that this metric measures the amount of entities we have been able to interpret during the conversion process. In this part of the experiment, we focus on the two most interesting entities, i.e., *work* and *person*. Out of total "818,249" person entities in the original records identified using their key properties, we have been able to interpret "689,957", thus achieving 84% qualitative completeness. For the *work* entity, 88% of the fields that potentially identify works have been processed. These results are affected by the quality of the data and the rules that we have been able to create for this dataset.

Qualitative Extension

In this section, we evaluate the quality of the semantic enrichment process, namely the discovery of a semantic element for an added entry. Recall that the semantic element corresponds to the type of an entity, which is selected by querying different knowledge bases using the techniques describes in Section 6.4.1. It is not possible to manually check the discovered entity for the whole collection. Thus, we have randomly chosen 800 records for evaluation. These records contain 682 added entries for which we search for a semantic element. FRBR-ML computes a score for all entities and the one with the smallest score is selected. In this evaluation, we have ranked these entities and presented the top-3 candidate matches for validation (including a manual search on the knowledge bases for the entry value when needed). This validation step was performed by 8 people from our research group, which means that they have to check all proposed LOD entities and decide whether it corresponds to the given work (based on available information, such as creators, titles, summaries, or types). If none of the proposed entities is correct, participants validated the work by manually searching DBpedia and Freebase. This manual validation forms a ground truth for the 682 records, based on which we are able to compute quality results of our approach.

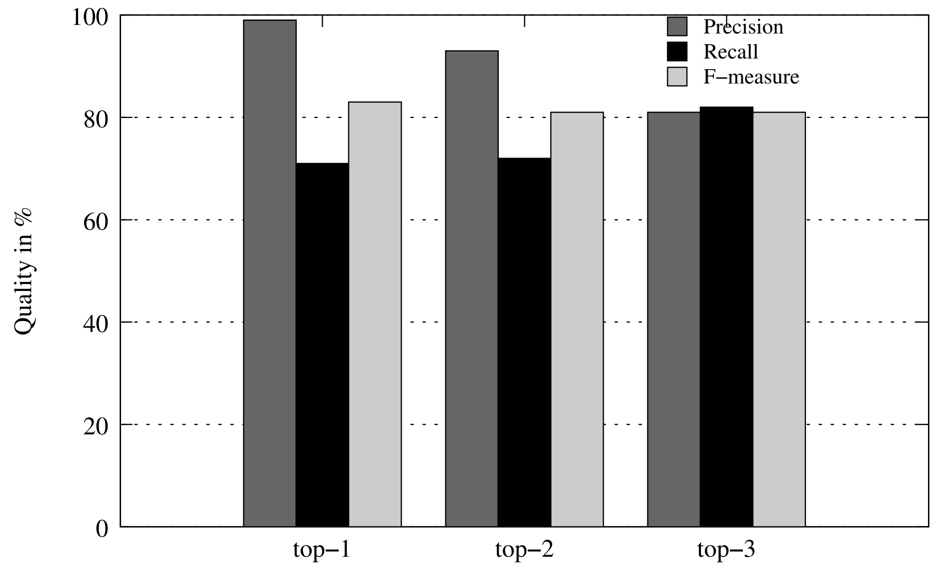
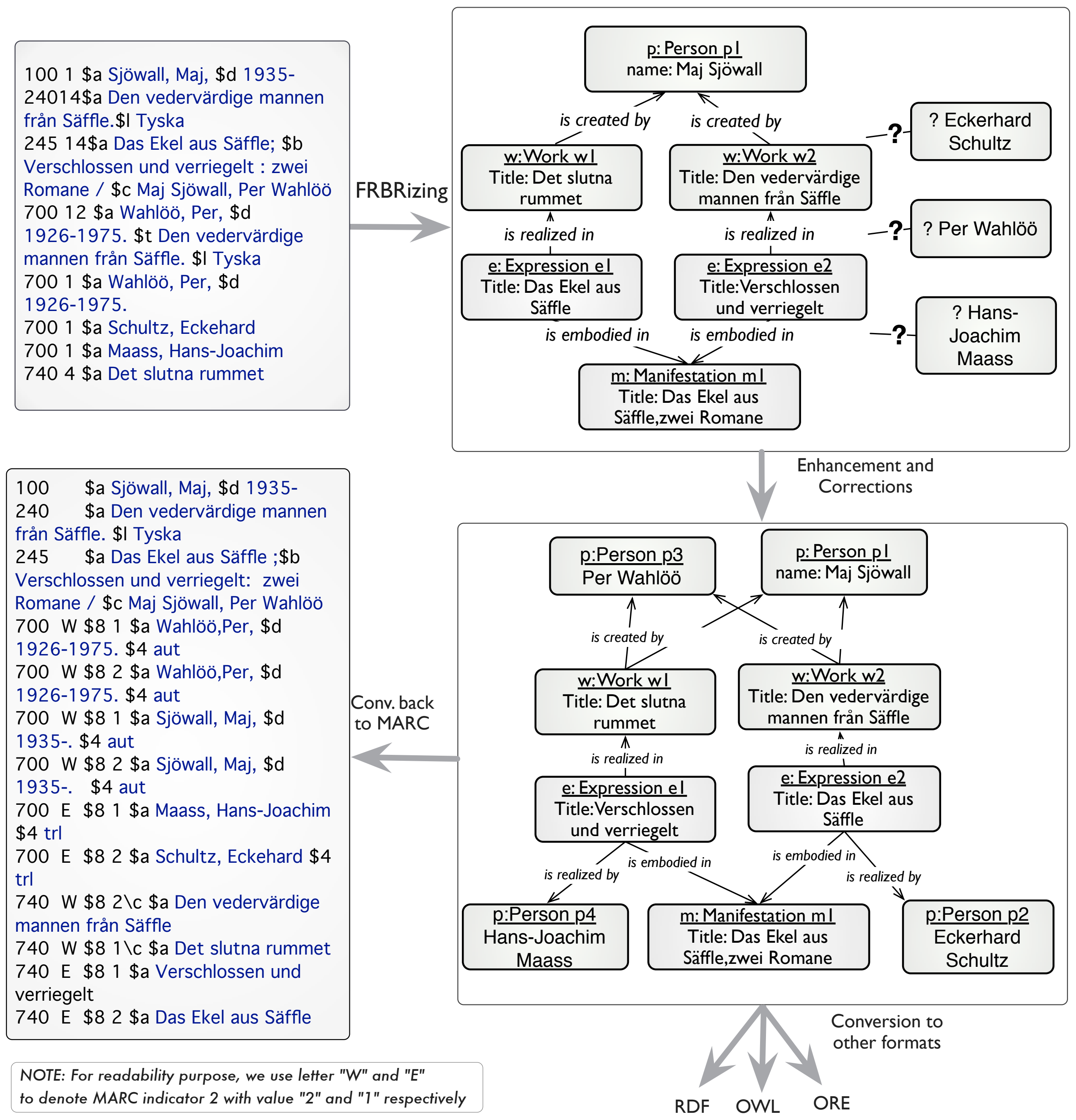


Figure 6.8: Quality Results of the Semantic Process by Top-K.

Almost half of the added entries do not have a corresponding entity in DBpedia or Freebase. Indeed, these added entries may refer to works or persons which are not popular enough to have a corresponding entity in the semantic knowledge bases. The remaining 343 works have at least one corresponding entity. We want to demonstrate that our semantic process identifies in most cases the correct entity at rank 1. Therefore, we compute the quality in terms of precision, recall and f-measure, as discussed in [74]. Applied to our context, *precision* represents the percentage of correctly identified entities among those discovered. On the other hand, *recall* stands for the percentage of entities correctly identified by our approach with respect to the total number of correct entities (based on ground truth). *F-measure* is a trade-off between precision and recall. Figure 6.8 depicts the quality obtained by our approach at top-1, top-2 and top-3. i.e., top-2 means that we consider entities which are ranked first and second by our semantic approach. For instance, top-1 results were obtained from this raw data: 155 true positives (correctly linked), 2 false positives (incorrectly linked), and 64 false negatives (not linked but should have been). Thus, we achieve at top-1 99% precision score (155/157) and recall score of 71% (155/219).

We note that the precision at top-1 is close to 100%, which indicates that our approach does not discover too many incorrect entities. However, we miss some entities during the discovery process (recall equal to 71%). When considering the second and third ranked entities as well, we observe that more correct entities are discovered (recall



```
100 1 $a Sjöwall, Maj, $d 1935-
24014$a Den vedervärdige mannen
från Säffle.$l Tyska
245 14$a Das Ekel aus Säffle; $b
Verschlossen und verriegelt : zwei
Romane / $c Maj Sjöwall, Per Wahlöö
700 12 $a Wahlöö, Per, $d
1926-1975. $t Den vedervärdige
mannen från Säffle. $l Tyska
700 1 $a Wahlöö, Per, $d
1926-1975.
700 1 $a Schultz, Eckehard
700 1 $a Maass, Hans-Joachim
740 4 $a Det slutna rummet
```

```
100      $a Sjöwall, Maj, $d 1935-
240      $a Den vedervärdige mannen
från Säffle. $l Tyska
245      $a Das Ekel aus Säffle ;$b
Verschlossen und verriegelt:  zwei
Romane / $c Maj Sjöwall, Per Wahlöö
700  W $8 1 $a Wahlöö,Per, $d
1926-1975. $4 aut
700  W $8 2 $a Wahlöö,Per, $d
1926-1975. $4 aut
700  W $8 1 $a Sjöwall, Maj, $d
1935-. $4 aut
700  W $8 2 $a Sjöwall, Maj, $d
1935-.   $4 aut
700  E  $8 1 $a Maass, Hans-Joachim
$4 trl
700  E  $8 2 $a Schultz, Eckehard $4
trl
740  W $8 2\c $a Den vedervärdige
mannen från Säffle
740  W $8 1\c $a Det slutna rummet
740  E  $8 1 $a Verschlossen und
verriegelt
740  E  $8 2 $a Das Ekel aus Säffle
```

NOTE: *For readability purpose, we use letter "W" and "E" to denote MARC indicator 2 with value "2" and "1" respectively*

Figure 6.9: An illustration of How the Use Case is Solved.

values reaching 72% and 82%), but to the detriment of precision (decrease to 93% and 81%). As our semantic process aims at enriching records, it should ensure that we do not add too many incorrect semantic elements. In that case, our approach fulfills this goal since the first ranked entity selected by FRBR-ML is in most cases the correct one.

6.6.4 Solving the Complex Use Case

The qualitative extension only evaluates the quality of the semantic elements but it does not deal with the correction of the original ambiguous records. In Section 6.2.1, we presented a simple and a complex use case. The simple use case is tackled relatively easily because only one FRBR entity of each type is present in the record. However, the complex use case requires more effort to be solved. Figure 6.9 depicts the different transformations applied to the original MARC record up to the enriched one obtained by using our approach. The top part of the figure shows the initial MARC record and its FRBRized representation from the FRBRizer tool. We notice that both of them suffer from the same problems, i.e., the translators (Eckerhard Schultz, and Hans- Joachim Maass) and the second creator (Per Wahlöö) are not included in the output of the transformation because it is not clear how they are related to the entities found in the record.

The bottom right part of the figure illustrates the FRBR-ML based representation in which the missing semantic information about persons is enhanced and corrected. For instance, "Hans-Joachim Maass", "Eckerhard Schultz" and "Per Wahlöö" have been identified as *persons* by the knowledge-based matching method (Section 6.4.1). The second problem is tackled using the correction method (Section 6.4.2). For example, "Hans-Joachim Maass" is linked to the German expression "Verschlossen und verriegelt" that he has translated. The local collection lookup did not return any match for the expression title but querying $z3950.libris.kb.se$ with the query ""$find\ @attrset\ bib-1\ @attr\ 1=4\ Verschlossen\ und\ verriegelt$" enabled to discover the correct relationship. On the other hand, the relationship between "Per Wahlöö" and "Det slutna rummet" was found during the intra-collection search since there is a record which has only this work. From this FRBR-ML format, it is possible to convert to RDF, OWL, ORE and back to MARC.

The bottom-left part of the figure shows the results of transformation from the FRBR-ML to MARC. We notice that it is the corrected and enhanced version of the original record. Entities are grouped by the $8 *linking field*. In our example, "$8 1" groups the work "Det slutna rummet", the expression "Verschlossen und verriegelt", the creators

"Per Wahlöö" and "Maj Sjöwall", and the translator "Hans-Joachim Maass". For the second work, we applied "$8 2" as linking field. Indicators[11] W and E were adopted to denote whether the entity is related to work or expression. As an example, "Per Wahlöö" is related to both works since both indicators are W. In addition, the correct *relator codes* "$4 *trl*" for translators and "$4 *aut*" for creators are used to denote their roles.

This new representation is inspired by both UNIMARC and MARC 21. The separation between names and titles comes from UNIMARC format while the use of the $8 linking field is common in MARC 21. Thus, our format has been adapted to fulfill our requirements, it is still compatible with with the ISO MARC standard.

6.7 Summary

Experience and user feedback in the cultural heritage community has shown that the adoption of new semantic technologies is slow, mainly because traditional library catalogs are still employed with records stored in the legacy format. Thus, we have presented in this chapter FRBR-ML – an approach for extracting entity information from bibliographic data to populate a knowledge base. The format included in FRBR-ML can be used as an intermediary format to easily transform from/to MARC, RDF/XML, OWL and ORE. By writing an appropriate converter, one may also convert to other popular formats such as Dublin Core or ONIX.

The enrichment step in the metadata knowledge base population of FRBR-ML consists of different strategies to tackle issues related to the lack of semantics in MARC records and to the identification of basic relationships between entities. We have studied novel techniques for disambiguating obscure entries in original records, thus allowing to correct the initial input data. In addition, we have designed new metrics to check the quantity and quality of the transformation. These metrics evaluate the completeness, the percentage of duplicates and the amount of extra information due to the enrichment process.

The results of this chapters' experiments are promising. The merging process effectively removes duplicate entities, thus substantially reducing the size of the knowledge base. However, the format included in FRBR-ML contains redundant properties,

[11]MARC indicator 2

but these redundancies can be easily removed during transformation to other formats. Additionally, it ensures a very high rate of completeness while allowing to correct and enhance ambiguous records with semantic information. This chapter also demonstrated that this semantic enrichment minimizes the rate of potential incorrect information.

In the future, there are several areas for improvement. The user feedback from librarians is important and should help detect the potential weaknesses and advantages of the approach in real world settings. This feedback mechanism could be integrated into the system when running in continuous mode. Although the approach was presented in the context of MARC-based information and the FRBR conceptual model, the solution is generic that can be deployed for other types of information migration as well.

Another interesting issue to address is discovering complex relationships between entities. To fulfill this goal, a possible solution is to use pattern matching between involved entities. This issue is addressed in Chapter 8 in the context natural language documents.

Although, we have seen several research projects on the use of FRBR as an underlying model recently, these projects seldom explore the full potential of FRBR. In particular, no in-depth studies have been performed on how to actually deploy FRBR-inspired applications to end-user applications such as library catalogs available online. The issues concerned are related to the FRBR *user tasks* which basically deals with the presentation that should be entity-centric rather than simply displaying a groupings of works discovered in other records.

Other sources of structured data are websites that sell various products, such as Amazon[12], Flickr[13], Twitter[14]. It is common for these large Web sites to expose their data for reuse via Web APIs. *Programmable Web*[15] provides a comprehensive list of such APIs. These APIs provide various query interfaces and return results using a number of different formats such as XML, JSON or ATOM. Therefore, an interesting question is: *Can we apply conceptual domain model to product descriptions exposed via Web APIs?* In the next chapter, an experimental study is presented on the example of using FRBR to describe product information.

[12]http://www.amazon.com (Last checked January 2012)
[13]http://www.flickr.com (Last checked January 2012)
[14]http://www.twitter.com (Last checked January 2012)
[15]http://www.programmableweb.com (Last checked January 2012)

7 Exploiting Metadata on the Web

7.1 Introduction

In the metadata extraction approach presented in the previous chapter, we have considered existing legacy metadata as the source of input data. However, the Web still remains the primary source of information for many users. The amount of data available online is far larger than the one stored in library catalogs. However, this Web data is not well structured and not machine-interpretable, although the emergence of the Semantic Web aims at tackling this issue [25]. For instance, representing facts with triples enables computers to understand and use reasoning to answer complex queries. Libraries are also increasingly interested in linking their data to the LOD cloud [131] as it enables semantic reuse of data providing a basis for new and innovative services.

In this chapter, we closely examine application of a semantic model model to descriptions of Web product metadata. As the Web contains a lot of resources that may represent products of creative or artistic endeavor, we present an approach to transform the information about these products into the FRBR model. As the FRBR model focuses on modeling creative works in multiple levels of abstraction (work, expression, manifestation, and item), an example of such hierarchy is depicted in Figure 7.1. In this example, the three-part epic by J.R.R. Tolkien “The Lord of the Rings” is an abstract work encompassing “The Fellowship of the Ring”, “The Two Towers”, and “The Return of the King”. We advocate that such a representation would enable websites (e.g. e-commerce) to better organize and exploit those products.

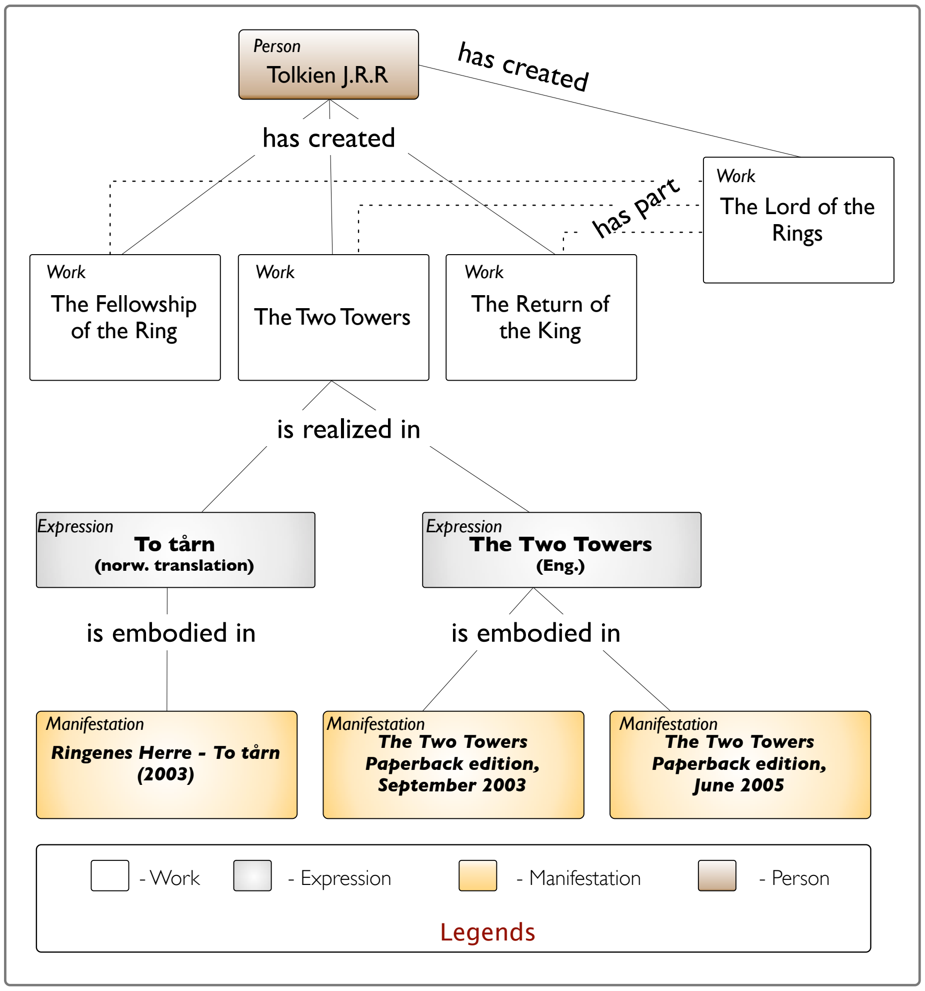


Figure 7.1: A Fragment of Lord of the Rings FRBR Work by J.R.R. Tolkien.

7.2 Overview

The FRBRization workflow is illustrated in Figure 7.2. From the input product descriptions, we first identify the corresponding works (Section 7.3.2), then we generate related manifestations, expressions (Section 7.3.3) and actors (Section 7.3.4). The process of creating relationships between the FRBR entities, presented in Section 7.3.5, produces the FRBR collection.

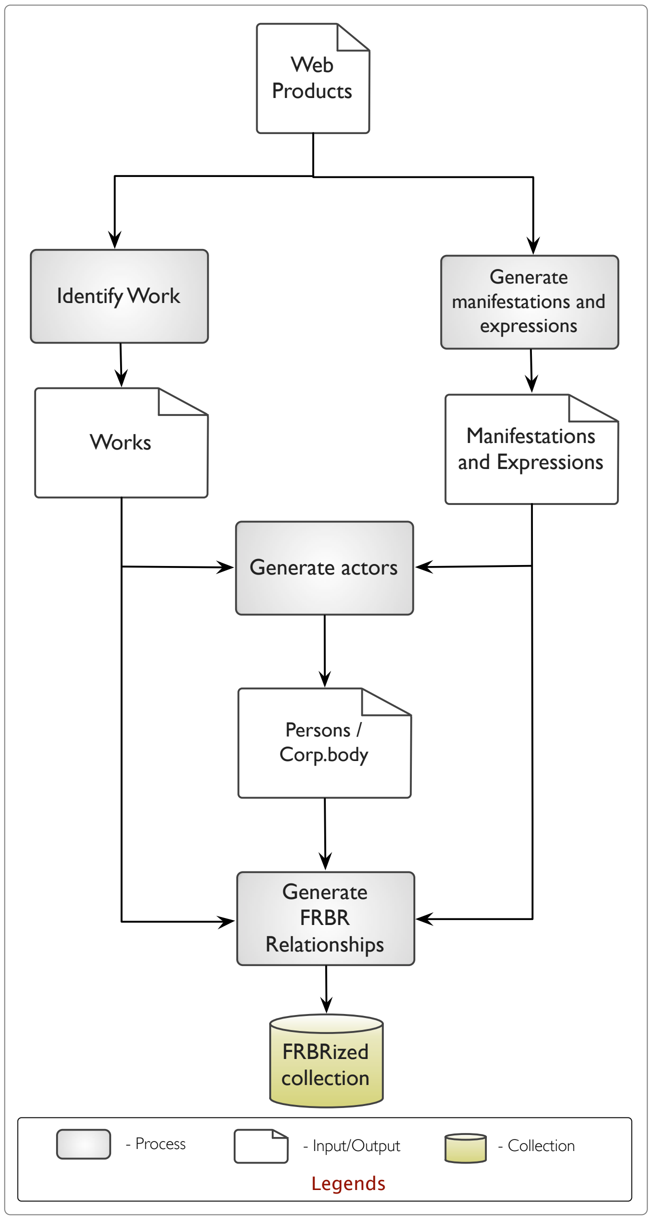


Figure 7.2: The FRBRization Workflow.

7.3 Interpreting Web Product Metadata

One of the main difference between existing approaches described in Chapter 4.7 and our work deals with the input data. Our interpretation process takes as an input descriptions of products found on the Web, specifically products sold by e-commerce websites (e.g. Amazon). Information about products tend to have different properties from their record-based counterparts found in library catalogs. A first difference is that products do not have the same structural pattern as MARC records, and they are stored in a variety of formats. A second one deals with the identification of products, which is unambiguously referenced by URI, thus providing a basis for reuse and exchange [86]. Additionally, e-commerce websites usually provide faceted navigation where the ranked list of results can be filtered on several dimensions. Yet, Web products can be related to FRBR manifestation level, similarly to library catalogs. For example, the 2005 paperback version of the book "The Two Towers" sold for $8.76 at Amazon bookstore is a product which is an item a manifestation that embodies an original English expression of the work "The Two Towers" by Tolkien.

7.3.1 Formal Model

Let $\mathcal{P} = \{p_1, p_2, ...p_m\}$ be a non-empty set of input product descriptions. Each product description p is described with an identifier id_p and a set of attributes $\mathcal{A}$. In other words:

$$\forall p \in \mathcal{P}, \quad p =< id_p, \mathcal{A} >$$

The set of attributes is defined by $\mathcal{A} = \{a_1, a_2, ...a_j\}$. A subset of these attributes $\mathcal{A}' \subseteq \mathcal{A}$ is used to generate FRBR entities by applying the following functions:

$$gen\_work(\mathcal{A}', id_p) \rightarrow w \tag{7.1}$$

$$gen\_manif(w, id_p) \rightarrow \mathcal{M}, \text{with } \mathcal{M} = \{m_1, m_2, ..., m_n\} \tag{7.2}$$

$$gen\_expr(w, \mathcal{M}) \rightarrow \mathcal{E}, \text{with } \mathcal{E} = \{e_1, e_2, ..., e_k\} \tag{7.3}$$

Formula 7.1 identifies a work w given the subset of attributes $\mathcal{A}'$ and the identifier of the product description id_p. With the work w, we generate a set of manifestations $\mathcal{M}$ (Formula 7.2) and a set of expressions $\mathcal{E}$ (Formula 7.3). The work w, each manifestation $m \in \mathcal{M}$ and each expression $e \in \mathcal{E}$ have a set of FRBR attributes $a_w \in \mathcal{A}_w$, $a_m \in \mathcal{A}_m$, and $a_e \in \mathcal{A}_e$ respectively. To generate actors, we need to combine attributes from all previously generated entities. We define $\mathcal{A}_{all}$ as a union of sets computed by

Formula 7.4:

$$\forall m \in \mathcal{M}, \forall e \in \mathcal{E}, \quad \mathcal{A}_{all} = \mathcal{A}_w \cup \mathcal{A}_m \cup \mathcal{A}_e \tag{7.4}$$

This set $\mathcal{A}_{all}$ enables us to generate a set of actors $\mathcal{T}$ by applying the Formula 7.5.

$$gen\_actors(\mathcal{A}_{all}) \rightarrow \mathcal{T}, \text{with } \mathcal{T} = \{t_1, t_2, ..., t_l\} \tag{7.5}$$

The final step in the FRBRization process is to establish FRBR relationships between all entities to obtain the FRBR collection $\mathcal{C}$ as shown below:

$$gen\_rel(w, \mathcal{M}, \mathcal{E}, \mathcal{T}) \rightarrow \mathcal{C} \tag{7.6}$$

7.3.2 Identifying a Work

The FRBR work is an abstract distinct intellectual or artistic creation. This entity is the cornerstone of the FRBR model and any FRBRization process needs to include a method for identifying the work entities.

Web product descriptions are seen at the manifestation level, therefore some attributes of expression and work can be found in the description of the product. For example, the language of the book is an attribute of the expression while the title and author(s) may refer to the original work.

The following techniques can be used to identify a work:

- Creating a work based on title/author and other attributes of the resource if the work has not been created yet; the database of works is then incrementally updated.
- Using an external service to identify a work, e.g. OCLC Classify API [1] for books using ISBN number, ISMN/ISRC music database for music or IMDB API for movies.
- Use z39.50 [2] (or SRW/SRU) protocol to search and fetch the relevant MARC record from publicly available catalogs and then use similar technique to

[1]http://classify.oclc.org (Last checked January 2012)
[2]http://www.loc.gov/z3950/ (Last checked January 2012)

Work-Set algorithm by OCLC [104].

The two latter methods take an identifier as input. On the contrary, the first method requires attributes such as title/author, thus leading to string matching problem. For instance, if the input product description is a translation of a work, we have to make sure that the correct work is discovered.

Having identified the FRBR work, we then need to retrieve other manifestations associated with the work.

7.3.3 Generating Related Manifestations and Expressions

As the FRBR expression is perceived as an obscure abstract entity that is easier to identify, the main challenge is to discover the related manifestations. To achieve this goal, we have identified the following methods:

- Search for author in z39.50 enabled repositories.
- use external service (e.g. xISBN[3] or LibraryThing's ThingISBN[4], Spotify API[5]).

These methods rely on external sources. Note that using the first method implies to employ FRBRization techniques already proposed for MARC records [2, 104, 132].

From the set of related manifestations, we can automatically generate expressions by analyzing attributes pertaining to this entity type. For example, attributes such as language and translator are used to identify expressions. Note that the identifier for an expression is automatically generated at this stage.

The next step deals with actors of this work.

7.3.4 Generating Actors

An actor is a person or corporate body (organization) responsible for the creation or realization of a work. Products available from e-commerce websites usually have information about the responsible for the work, such as author of a book, composer

[3] `http://labs.oclc.org/xisbn/` (Last checked January 2012)
[4] `http://www.librarything.com/api` (Last checked January 2012)
[5] `http://developer.spotify.com/en/libspotify/` (Last checked January 2012)

of a music, director of a movie. Generating an actor can be performed using the following methods:

- Create a local authority file or use existing authority files from external sources
- Search Virtual International Authority File (VIAF) and link actors to VIAF

Contrary to the first method which is a time-consuming and complex task, the second approach includes a Web-based API and the VIAF collection contains data from many national libraries around the world. At the end of this step, we have generated all the FRBR entities required in our FRBRization process.

7.3.5 Generating FRBR Relationships

The final phase of the workflow is to generate the actual FRBR entities and establish relationships. Since we have information about each entity from previous steps, this step creates a collection of entities in a specific output format such as a series of SQL statements that can be used to insert into a relational database, HTML, XML, RDF or a simple text file. This step requires that an entity has a unique identifier and can be unambiguously referenced. For manifestations, we have already identifiers. Work and expression entities can be assigned locally generated identifiers since there is no publicly available global unique identifiers for these entities.

7.4 Experiments

The effectiveness of the approach has been evaluated by the use of a collection of product accessible via a Web API. We have chosen to create our dataset based on search results from Amazon since its database potentially contains a great number of items[6]. Although we focused on books, our approach is generic and it can be applied to other types of products as well.

[6]A blank search on “Books” returns 32,058,092 items.

7.4.1 Protocol

Using Amazon's Product Advertising API[7], we have searched for works by the 80 best selling fiction authors extracted from Wikipedia[8]. Due to the constraints set forth by Amazon on the number of requests that can be sent in one hour, we have a limited number of items to the first page of the results set (10 items per page). We have performed an automated search using the *ItemSearch* operation on *Books* index. We excluded items representing kindle edition. We also filtered out the products not solely offered by Amazon ("MerchantId"=Amazon). Additionally, we performed search on Amazon's *Video* index using previously submitted queries on *Books* index. The attributes made available within these products, among others, are, title, author (director for movies), contributor, ISBN, language, release date. As can be seen from Table 7.1 (column "# of Input Products"), half of the content of the initial set of products were book. Most of these books are published in English language, but the input products include other languages such as Japanese, Chinese or Russian.

Since our focus was on FRBRization of individual works, the relationship between original work and its adaptation to movies have not been drawn. The same remark is true for works including multiple works, expressions, etc. (such as "Murder in the Mews and Other Stories - Hercule Poirot" by Agatha Christie).

In Section 7.3.2 we have proposed three methods to identify the work corresponding for a product. To avoid implementing z39.50 protocol and reduce latency, we used the Classify API by OCLC. Classify API is a web service from the OCLC Office of Research. It enables users to retrieve information about the classification of the submitted work. The next step in the workflow is to generate related manifestations and expressions. Since we chose OCLC Classify API to identify work, we had a greater chance of match in the same database. Therefore, to obtain a list of related manifestations, we again used an OCLC Service - xISBN. The xISBN Web service returns ISBNs and other information associated with an individual intellectual work that is represented in the WorldCat catalog.

The next step involves the identification of actors. We chose to link actors (persons and corporate bodies) to Virtual International Authority File (VIAF). VIAF is a joint project of national libraries of several countries and it is hosted by OCLC. VIAF's long-

[7]Amazon Product Advertising API, Ver 2010-10-01, `http://j.mp/amznProductAPI` (Last checked April 2011)

[8]`http://en.wikipedia.org/wiki/List_of_best-selling_fiction_authors`, as of November 2010

term goal is to include authoritative names from many libraries into a global service that is available via the Web for free. Using VIAF's public API, we submitted queries for each contributor in the dataset. We used an average of Monge Elkan, Jaro Winkler and Levenshtein to calculate the similarity in the top 30 hits.

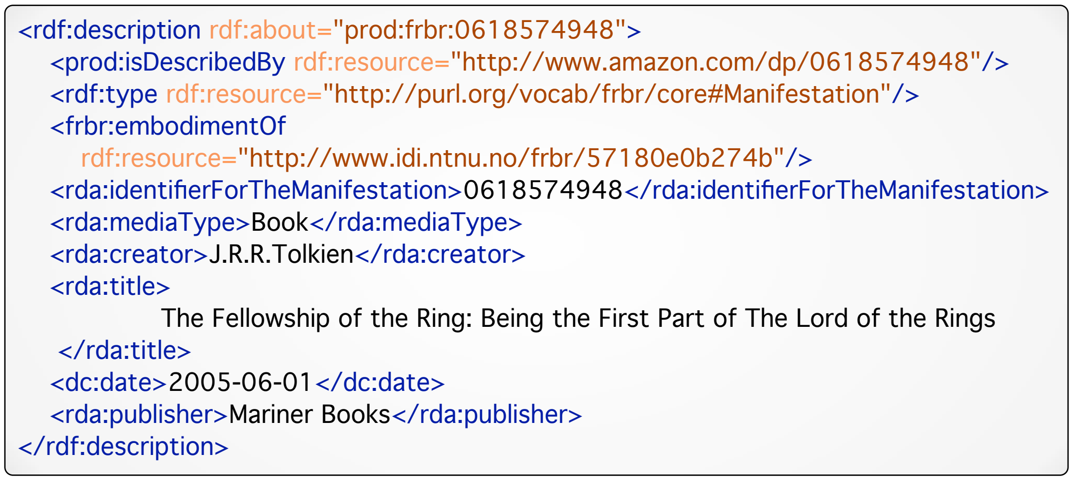
```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:description rdf:about="http://www.idi.ntnu.no/frbr/af18-ce924fa856f6">
   <rdf:type rdf:resource="http://purl.org/vocab/frbr/core#Work"/>
   <rda:title>
          The Fellowship of the Ring: Being the First Part of The Lord of the Rings
    </rda:title>
   <rda:fullerFormOfName>
          Tolkien, J.R.R. (John Ronald Reuel), 1892-1973
   </rda:fullerFormOfName>
   <rda:creator>J.R.R.Tolkien</rda:creator>
   <rda:identifierForThePerson>
           http://viaf.org/viaf/95218067
   </rda:identifierForThePerson>
   <frbr:realization rdf:resource="http://www.idi.ntnu.no/frbr/ 38d3fa23-2ddc"/>
      <frbr:realization rdf:resource="http://www.idi.ntnu.no/frbr/ 15557f7a"/>
      <owl:sameAs>dbpedia:The_Fellowship_of_the_Ring</owl:sameAs>
</rdf:description>
```

Figure 7.3: Work RDF.

```xml
<rdf:description rdf:about="http://www.idi.ntnu.no/frbr/57180e0b274b">
   <rdf:type rdf:resource="http://purl.org/vocab/frbr/core#Expression"/>
   <rda:languageOfExpression>eng</rda:languageOfExpression>
      <frbr:realizationOf rdf:resource="http://www.idi.ntnu.no/frbr/af18"/>
      <frbr:realization rdf:resource="prod:frbr: 0618574948"/>
</rdf:description>
```

Figure 7.4: Expression RDF.

The final phase of the workflow was to generate the relationship between the FRBR entities. This is achieved using the identifiers created for the FRBR entities in the previous steps. The final output is a set of RDF files for each entity type. The XQuery function performing this operation uses all information gathered so far from various sources. Examples of output RDF files are shown in Figures [7.3,7.4,7.5].



```
<rdf:description rdf:about="prod:frbr:0618574948">
   <prod:isDescribedBy rdf:resource="http://www.amazon.com/dp/0618574948"/>
   <rdf:type rdf:resource="http://purl.org/vocab/frbr/core#Manifestation"/>
   <frbr:embodimentOf
      rdf:resource="http://www.idi.ntnu.no/frbr/57180e0b274b"/>
   <rda:identifierForTheManifestation>0618574948</rda:identifierForTheManifestation>
   <rda:mediaType>Book</rda:mediaType>
   <rda:creator>J.R.R.Tolkien</rda:creator>
   <rda:title>
             The Fellowship of the Ring: Being the First Part of The Lord of the Rings
    </rda:title>
   <dc:date>2005-06-01</dc:date>
   <rda:publisher>Mariner Books</rda:publisher>
</rdf:description>
```

Figure 7.5: Manifestation RDF.

7.4.2 Results

Table 7.1 summarizes the results of this experiment. The second column provides the number of Amazon products grouped by product type and the number of actors extracted from these products. In the third column, we show the number of discovered entities during the FRBRization process using Classify API, xISBN and VIAF services. Contrary to what could have been expected, the number of discovered works (739) is less than the number of input products (1216). This occurs because the set of input products contains different products that correspond to the same work (e.g. Norwegian “Tå tårn” and English “The Two Towers” corresponding “The Two Towers” work). We notice that the number of manifestations strongly increased (from 1656 to 28245) because we fetched all manifestations of works. More specifically, we successfully discovered more books and videos while related music and DVDs were more difficult to fetch. The total number of actors we extracted was 2221 while 70% of them were found in VIAF (1569).

The last column describes the number of entities in our FRBRized collection, i.e., after removing unidentified works and actors. The initial set of input products we populated from Amazon contained 1656 items while the number of FRBRized manifestations is **28245**. The FRBRized collection includes works, translations, and movie versions of those works.

Out of total 739 generated works, we have obtained a match for **684** works in Classify.

FRBR Entity Type	**# Input Resources**	**# Discov. Entities**	**# FRBRized Entities**
Work		739	684
Expression			5074
Manifestation	1656	28245	28245
-Book	856	27588	27588
-Video	102	542	542
-DVD	190	113	113
-Music	508	2	2
Actor	2221	1569	2221

Table 7.1: Results of the FRBRization.

The unidentified 55 works were mainly not in English language. Dealing with the actors, the final collection contains **2221** actors since the generated actors based on VIAF were automatically assigned locally generated identifiers.

The following issues were encountered during the experiment:

- Search results from Amazon needed to be cleaned. Indeed, it can contain dirty data such as "The Lord of the Rings: The Return of the King (Widescreen Edition)" and unrelated products. Since we performed search on Amazon database, we could not always limit our list to only works by our initial set of authors. This happens because authors could be mentioned in descriptive text of the resource.
- We could not FRBRize the whole set of product descriptions because the Classify service did not have an entry for the requested product. To solve this issue, we could aggregate the results from similar services (e.g. z39.50).
- VIAF had several identical entries for number of authors (e.g., "Arthur Rankin Jr."). In this case, the system chooses higher ranked item and if the score is identical the item chosen in random manner.

7.4.3 Evaluation

We conducted a manual evaluation of the results. To evaluate the quality of the FRBRization process, we performed a comparison with the DBpedia knowledge base. This knowledge base was chosen because it potentially contains a great amount of general knowledge. In addition, it is strongly connected to other knowledge bases, such as MusicBrainz, VIAF, and Freebase.

The goal is to verify and validate the different attributes of the FRBR works against the attributes of the corresponding DBpedia entity. Our assumption is that the resulting FRBR work is correctly identified if the values of its main attributes are more or less identical with the attribute values of the DBpedia entity. The following attributes were compared to validate FRBR works:

- the title of the work with the label of the LOD entity;
- the type of work with the type of the LOD entity;
- the creator of the work with the appropriate creator attribute of the LOD entity (e.g., author for books, director for movies);
- the dates of creation of both FRBR work and LOD entity.

Note that the attributes of a work and a LOD entity are not identical, so participants validated works at their own discretion using common sense.

Half of the works (343) were randomly selected for a manual validation. This process is time consuming but we believe that this partial validation provides a general overview of the quality of the FRBRization. Four researchers in our group who are very well acquainted with the FRBR model were in charge of this manual validation. To facilitate this task, we have used the FRBRpedia tool [69]. This tool enables to automatically discover the DBpedia entity for a given FRBR work. If a work was not matched to any DBpedia entity by the tool, participants validated the work by either manually searching DBpedia or checking information on the Web. This case accounts for 98 works out of the total 343 works.

This manual evaluation demonstrated that all works contain the correct attribute values according to the DBpedia knowledge base. Thus, our FRBRization approach is effective in terms of quality, although a few entities contain dirty data (extra information). In addition, the 343 works include 18 duplicates that our approach was not able to merge. The main reason for this issue is due to minor differences between two works (such as a different foreword or a new cover). In that case, the FRBR notion of *super work* may be useful. Another significant improvement is to detect and extract aggregate works, i.e., a publication that contain multiple distinct intellectual contributions within a single product description. Currently the individual works within the aggregate work are not separately identified, which in itself poses a great challenge.

7.5 Summary

In this chapter, we have demonstrated that the use of FRBR model as a semantic data model is not only limited to the library domain, but can be applied to product information found on the Web. The main benefit of FRBRizing product information on the Web is that FRBR provides support for knowledge-like representation of the data enabling a broad support for exploratory interfaces where users are presented with a list of works for each author and can navigate relationships to learn about and find other versions or preferred editions of a given work.

The resulting FRBR works have been validated using the DBpedia knowledge base. The FRBRization approach was effective in terms of quality according to DBpedia validation. However, few duplicates could not be merged because the original input data contain dirty information. Nevertheless, we conclude that the approach approach is worth further investigation.

The metadata extraction approaches to populate a knowledge base presented in this chapter (and in Chapter 6) considered metadata as the initial input data. The results of the experiments indicate that, often it is difficult to achieve a complete graphical structure of entities described in the metadata. To tackle issues such as ambiguity and missing information, there is a need to extract information from other sources to complement and supplement metadata extraction, which is the topic of the next chapter.

Part IV

Extracting Entities and Relations

8 Extracting Entities and Relations from the Web

8.1 Introduction

In previous chapters, we have looked at how to extract entity information from legacy metadata and product information found on the Web. The main objective was to derive set of entities and relations, so facts, implicitly described in the metadata, become explicit. However, it was not possible to extract all information and in many cases, we are left with ambiguous entities that is not linked with relations to other entities. As a huge source of unstructured information, the Web is a candidate to extract such information. The challenge of transforming raw text into meaningful relational facts is an interesting problem that has attracted researchers from various domains: Entity Recognition, Entity Matching, Information Retrieval, Artificial Intelligence and Natural Language Processing. Indeed, dealing with textual documents involves the correct identification of named entities.

Recognizing these entities is usually not sufficient due to the strong heterogeneity in the content of the documents. Consequently, identical entities which are represented with different forms or labels should be matched either directly or to a common dictionary to avoid the generation of redundant relationships while ensuring a better confidence for a discovered relationship [113]. The huge amount of potentially large documents to be processed clearly requires an automated approach so that new relationships can be continuously discovered by minimizing human intervention [43, 88].

This chapter presents SPIDER[1] a relation extraction system. It aims at addressing the

[1]**S**emantic and **P**rovenance-based **I**ntegration for **D**etecting and **E**xtracting **R**elationships.

previously mentioned issues by integrating the most relevant techniques to generate trustworthy patterns and relationships. SPIDER is based on a perpetual process of generating patterns and examples either in supervised or unsupervised mode. The main intuition is that generic patterns which are derived from similar sentences and which are discovered in trustworthy documents are useful for detecting relationships in other documents. In summary, our contributions in this chapter are:

- Extracting relationships from a Web-scale source is a major bottleneck. Specifically, SPIDER does not require several days to perform a single iteration.
- Contrary to most knowledge extraction tools, we tackle the problem of uniquely identifying entities both to extend their list of spelling forms and to facilitate the matching to LOD.
- SPIDER includes a flexible pattern definition scheme. This scheme is used to merge similar patterns for efficiency purposes. In addition, we introduce the notion of confidence score that controls the ranking of patterns. The confidence score evolves over time as the system runs.
- Experiments confirm the benefits of our approach w.r.t. similar tools (*ReadTheWeb*, *Prospera*) in terms of quality and performance.

8.2 Overview

8.2.1 Problem Definition

The overall goal of SPIDER is to continuously generate relationships and patterns. Let us first define a relationship. It is a triplet <e_1, τ, e_2> where e_1 and e_2 represent entities and τ stands for a type of relationship. An entity might be represented with different labels in natural language text, and we note $l_e \in \mathcal{L}$ one of the mentions for the entity e. An example of a relationship is *createdBy* between the work entity *The Lord of the Rings* and the person entity *J.R.R. Tolkien*. Note that both the entities and the type of relationship are uniquely identified using an URI. An example denotes a pair of entities (e_1, e_2) which satisfies a type of relationship.

The patterns are extracted from a collection of documents $\mathcal{D} = \{d_1, d_2, ..., d_n\}$. Although not limited to, these documents are webpages in our context. Each document is composed of sentences, which may contain mentions of the two entities. In that case, the sentence is extracted as a candidate pattern. We note $\mathcal{CP}$ the set of candidate patterns given a collection of documents and a set of initial examples. Each candidate pattern is defined as a tuple $cp = \{t_b,\ e_1,\ t_m,\ e_2,\ t_a\}$ with t_b, t_m and t_a respectively standing for the *text before*, the *text in the middle* and the *text after* the entities. A sentence *"Bored of the Rings is a parody of Lord of the Rings"* is transformed to a corresponding candidate pattern *{"", e_1, "is a parody of", e_2, ""}*.

From the set of candidate patterns $\mathcal{CP}$, we derive a set $\mathcal{P}$ of generic patterns by applying a strategy s. A strategy is defined as a sequence of operations $s =< o_1, o_2, ..., o_k >$, each operation aiming at generalizing the candidate patterns. Namely, this generalization implies the detection of frequent terms and the POS-tagging of the other terms of the candidate patterns. Thus, a strategy is a function such that $s(\mathcal{CP}) \rightarrow \mathcal{P}$. All generated patterns are associated to a specific type of relationship τ. For instance, a generic pattern for the *parody* type is illustrated with *"{e_1} is/VBZ a/DT {parody, illusion, spoof} of/IN {e_2}"* [2].

Finally, a pattern and an example both have a confidence score noted $conf_p$ and $conf_e$ respectively. This score is based on the support, the provenance, the number of occurrences, the number of strategies and the iteration. A pattern similarity metric indicates the proximity between two (candidate) patterns. The notion of confidence score, as well as the one for operation, strategy, pattern similarity and (candidate) pattern, are further detailed in the next sections.

8.2.2 Workflow

Given two labels, SPIDER generates patterns and derives to a relationship. As illustrated in Figure 8.1, this pattern generation capability is guaranteed by the following two processes:

- **Pattern Generation** is in charge of detecting candidate patterns by using examples or provided entities and of generalizing these candidate patterns to obtain patterns for a given type of relationship.
- **Example Generation** exploits the previously generated patterns in order to discover new examples which satisfy the type of relationship.

[2]VBZ=Verb, 3rd person sing. present, DT=Determiner, IN=Preposition or subordinating conjunction

The knowledge base stores all generated examples and patterns. They are not only used to maintain the system continuously running, but they can be exploited from a user perspective too. One can query SPIDER in order to discover all possible types of relationship between two input entities. The second use case, similar to the entity search task, enables users to discover the second entity which satisfy a given relationship. The last use case is the example generation, when a user is interested in the discovery of two entities for a given type of relationship.

8.3 Pattern Generation

The pattern generation process either requires a few examples for a given type of relationship so that patterns for this type of relationship can be automatically generated (supervised), or it directly tries to guess the type of relationship for two given labels (unsupervised). The process is similar in both modes and it is composed of three main steps: extension of entities, extraction of candidate patterns from the collection of documents and their refinement into patterns.

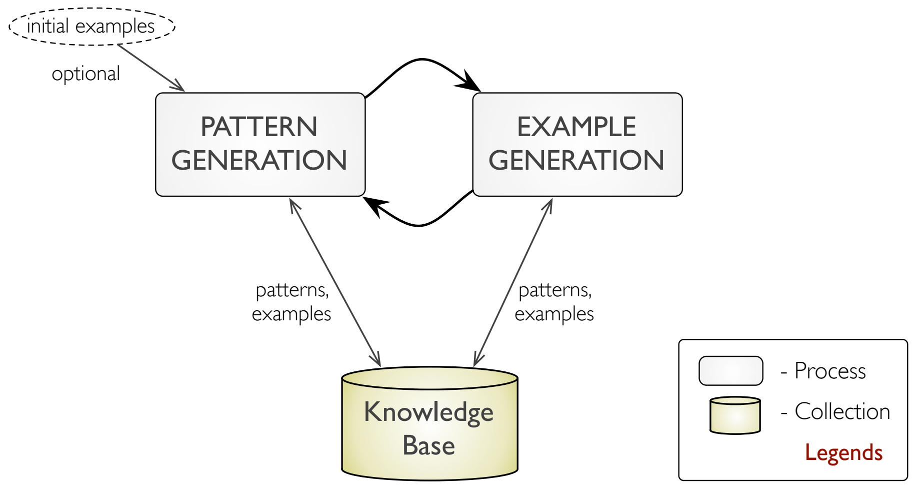


Figure 8.1: Workflow of SPIDER.

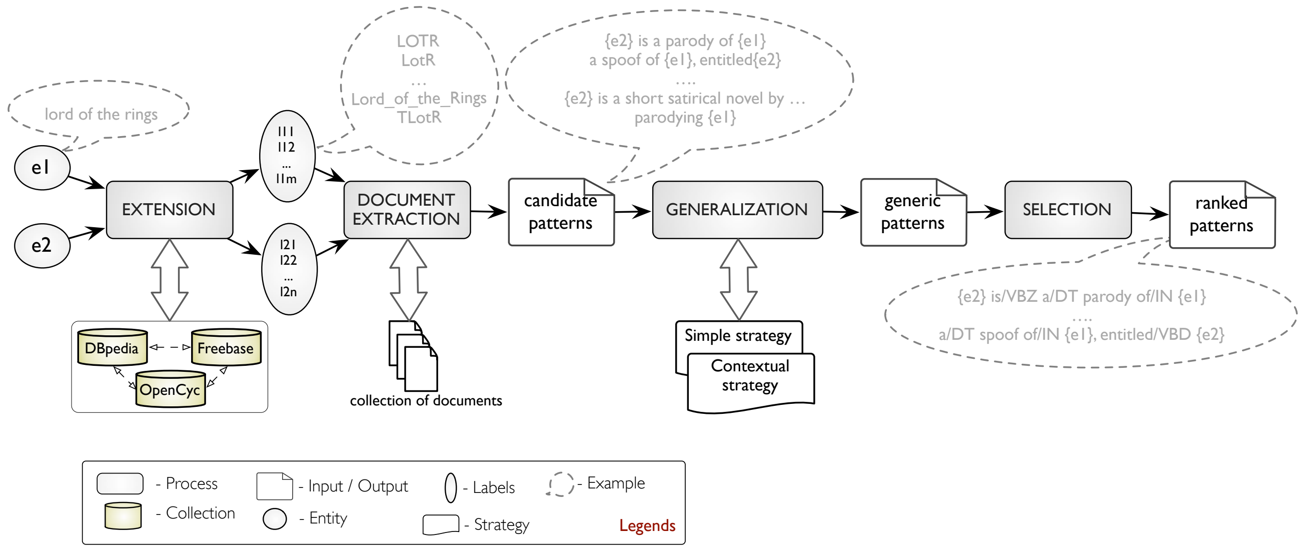


Figure 8.2: The Pattern Generation Process.

8.3.1 Extending Entities

In a document, entities are not uniquely identified by a label but they have alternative labels or spelling forms. Therefore, extending these entities with their alternative labels is a crucial step and it requires the correct identification of the entity. For instance, the entity *"Lord of the Rings"* can be labeled *"LOTR"* or *"The Lord of the Rings"*. To avoid missing potentially interesting relationships, we search for these alternative forms of spelling in the documents. Given an entity e represented by a label l, the goal is to discover its set of alternative labels $\mathcal{L}_e = \{l, l^1, l^2, ..., l^n\}$. The idea is to match the entity against LOD semantic knowledge bases to obtain this list of alternative labels. Namely, we build various queries by decomposing the initial label and we query in the aliases attributes of knowledge bases (i.e., *common.topic.alias* for Freebase, *wikiPageRedirects* for DBpedia, etc.). In most cases, several candidate entities are returned and the tool tries to automatically select the correct one.

The process of automatically selecting the correct entity is achieved as follows. First, an AND query is constructed with the two labels. Clusters of documents are built representing documents belonging to a set of specific type of entities. The n number of words around labels are extracted and stemming performed on words. Our assumption is based on the fact that documents mentioning the same entities tend to have similar words. Therefore, a graph of semantically related words is built. The most important documents in the cluster are then compared against the abstract of the automatically selected entities. Next, we extract frequent terms from the most important documents in the result set and use these frequent words as extensions. Note that if disambiguation is not possible, we discard the example and we do not use it for subsequent pattern generation. The result of the extension process is a list of alternative labels as illustrated in Figure 8.2.

Contrary to the Prospera system [153], which assumes that the entities exist with their type in the YAGO knowledge base, our approach is more flexible. At first, we also rely on a knowledge base to obtain a list of alternative spellings. Most knowledge bases provide this property (e.g., *wikiPageRedirects* for DBpedia or *alias* for FreeBase). In case of disambiguation, i.e., an alternative spelling which could be appropriate for more than one entity, we apply the most common usage as the knowledge base does. For instance, an entity labeled "JRR" could have the following meanings: "John Ronald Reuel" and "Journal of Radiation Research", but the former one has a most popular usage. Note that this ambiguity mainly exist with acronyms and homonyms. This issue can be solved easily due to the second label of the query: the association between "Furusawa Yoshiya" (the current editor-in-chief of the Journal of Radiation Research) and "John Ronald Reuel" does not return any results, while a query composed of the extended labels "Furusawa Yoshiya" and "Journal of Radiation Research" provides thousands of documents.

The main issue in this step deals with the absence of the two labels in any knowledge base, which means that the entities cannot be extended. The number of retrieved documents in that case could not be sufficient to extract good candidate patterns. The first solution consists of analyzing these retrieved documents to detect potential alternative spellings by applying metrics such as tf-idf and Named Entity Recognition techniques. Another possibility is to relax the similarity constraint when searching a label in a knowledge base. In other words, a strict equality measure would not be applied between the label and the candidate spelling forms from a knowledge base. Rather n-grams or Levenshtein similarity metrics with a high threshold would be a better choice [50]. An interactive mode with the user can let him/her select which candidate spelling is the correct one. We let this specific case for future work since experiments show that due to the large number of knowledge sources, entities can usually be extended. The last interesting point of this extension process is that it automatically provides a URI for each entity, thus enabling the storage of the relationship as a triple.

8.3.2 Extraction of Candidate Patterns

As depicted in Figure 8.2, the outcome of document extraction process is candidate patterns. Given the lists of extended labels for both entities, our tool associates all alternative labels of the first entity to all labels of the second entity (Cartesian product) to build different queries. The documents resulting from these queries are ranked according to their relevance score. The candidate patterns are extracted by parsing these documents and locating the sentences with co-occurrence of both entities (defined by a maximum number of words between them, currently 15 words). Note that we include in the candidate patterns the text before and after the entities to obtain full sentences. The final step aims at refining the candidate patterns to obtain patterns.

8.3.3 Generalization into Patterns

The main goal is to generalize the extracted candidate patterns. To ensure flexibility in our approach, this process is performed using operations and strategies. A confidence score enables the selection of the best patterns.

Operations and Strategies

In our context, various operations are applied to candidate patterns to generalize them. Given a set of candidate patterns $\mathcal{CP}$, an operation $o \in \mathcal{O}$ will return a subset $\mathcal{CP}'$ of these candidate patterns. We do not describe all individual operations but we rather present the main categories with a few examples.

- **Clean**. As the name suggests, this type of operation is in charge of cleaning the candidate patterns from useless words, plural forms, etc. It also discards irrelevant candidate patterns (e.g., those with a too short middle text t_m).
- **Tagging** is a category which enables the processing of natural language. For instance, such an operation is based on Part-of-Speech (POS) to annotate the candidate patterns and generalize them. Another one relies on Named Entity Recognition to detect entities in the candidate patterns (date, locations, works, persons, etc.).
- **Merge**. Candidate patterns may be very similar. In that case, it is interesting to merge them. The decision for merging usually requires a rule with

a comparison function and a decision-maker. In this context, the comparison functions are similarity measures (N-grams, Jaro-Winkler, etc.) while the decision-maker is a threshold. For example, such a rule could be *(3grams similarity with threshold* > 0.8*)*.

We could define more operations based on other works. For instance, we could rely on the Falcons tool to remove entities in the candidate patterns [47], use different similarity metrics to merge the candidate patterns [50], etc. All these operations can be combined into a strategy.

A strategy is defined as a sequence of operations applied to a set of candidate patterns and it returns a set of patterns. More formally, a strategy $s =< o_1, o_2, ..., o_n >$ is a function such that $s(\mathcal{CP}) \rightarrow \mathcal{P}$. The advantage of the strategies is the promotion of flexibility since the design of new strategies is simple. In addition, a pool of different strategies reduces the probability of a "blockage" of the system. One of the simplest strategy consists of merging identical candidate patterns and POStagging them. In the next section, we present a contextual strategy based on frequent terms.

The Contextual Strategy

This strategy, used later in the experiments, is based on term frequency. Our intuition is based on two facts. First, most candidate patterns contain a few interesting terms to denote the type of relationship. For instance, the sentence *"Bored of the Rings is a parody of Lord of the Rings"* mainly includes one meaningful term (*parody*). This means that the verb "is" or the determinant "a" could be replaced by other terms of the same nature. Second, many similar approaches only consider the text between the labels of the two entities. Therefore, they miss interesting patterns. On the other hand, SPIDER takes into account the whole context surrounding the two entities when needed and identifies part(s) which should be stored as a pattern.

That is the reason for indexing the most frequent terms from all candidate patterns after a cleaning process[3]. By applying stemming to these frequent terms and matching these terms to the Wordnet dictionary, we are able to build clusters of concept based on Wordnet relationships such as synonyms, direct hyponyms, related terms, etc. Each cluster is labeled using one of its terms, i.e., the most centric one for representing the concept given the Resnik distance between all terms [170]. The main issue is the

[3]Frequent, not meaningful words for the relationship are removed: "the", "is", etc.

selection of the relevant cluster(s) for the given type of relationship. Indeed, several clusters which represent distinct concepts could be created. For instance, the two entities *"Lord of the Rings"* and *"Tolkien"* may lead to the creation of three clusters: *book*, *fantasy* and *writer*. Therefore, we apply the Resnik distance between the label of each cluster and the type of relationships to select the relevant cluster(s). Finally, we use a POS-tagger for all words that are not frequent. A frequent term may be replaced by any related term from its cluster. This generality enables the merging of similar patterns. Examples of patterns are shown in Figure 8.2.

8.3.4 Selection of Patterns

The last issue deals with the selection or ranking of the generic patterns. Thus, a confidence score noted $conf_p$ is computed for each pattern p with Formula 8.1. Our intuition is to exploit all information which allowed the discovery of the patterns and to compare a pattern with the ones of the same type of relationship.

$$conf(p) = \left(\frac{\alpha sup_p + \beta occ_p + \gamma prov_p}{\alpha + \beta + \gamma} \right) \tag{8.1}$$

The support sup_p is defined as the ratio between the number of examples ex_p that this pattern is able to discover and the total number of examples ex_τ discovered by all patterns of the same type of relationship τ. Note that the support cannot be computed at the first iteration.

Similarly, the occurrency occ_p stands for the number of candidate patterns which led to the generation of the pattern p. It is normalized by the total number of candidate patterns used to generalize all patterns of the same type of relationship τ.

$$supp_p = \frac{ex_p}{ex_\tau} \qquad occ_p = \frac{occ_p}{occ_\tau}$$

The provenance $prov_p$ refers to the relevance of the documents from which the candidate patterns which generalize a given pattern have been extracted. The relevance is evaluated given three metrics: the relevance score namely applies tf-idf on the content of the document and its values are bounded by a maximal value which depends on the query. PageRank[4] is widely known due to the Google search engine. The PageR-

[4]http://j.mp/Clueweb09-Pagerank (Last checked October 2012)

ank scores of our collection are in the range $[0.15, 10]$. Finally, SpamScore indicates the probability that a document is a spam or not [129]. The idea is to average the scores returned by these three metrics for all documents from which patterns have been derived. We note $\mathcal{D}_p$ this set of documents for the pattern p, and d_p^i a document of this set. The following formula computes a score in the range $[0, 1]$ to evaluate the average relevance of this set of documents, and thus the provenance of the pattern:

$$prov_p = \frac{\sum^{d_p^i} \frac{1}{3}\left(\frac{relevance(d_p^i)}{max(relevance(\mathcal{D}_p))} + \frac{spamscore(d_p^i)}{100} + \frac{prank(d_p^i)}{10}\right)}{|\mathcal{D}_p|}$$

8.4 Relationship Discovery

In the previous step, we have generated patterns for a given type of relationship. Our approach aims at discovering relationships between entities in three different use cases: discovering the type of relationship, searching for an entity, and discovering new examples. Since these use cases roughly tackle the same challenge from different angles, we first describe the different issues related to the exploitation of the patterns.

8.4.1 Challenges for Exploiting patterns

Using the generated patterns to discover knowledge from plain texts involves the addressing of the following two challenges: pattern similarity and NER.

Pattern similarity. When analyzing sentences in a document, SPIDER needs to evaluate the similarity between a sentence and a pattern. Thus, we have designed a pattern similarity metric. The intuition which underlies our metric is twofold: (i) the presence of frequent terms in the sentence is crucial while there is more flexibility for less important terms and (ii) the position of the words should be taken into account. First, the sentence is transformed (cleaning, POS-tagging and replacing the mentions of the entities) so that both the sentence s and the pattern p are composed of POS-tagged terms. The pattern may also include a list of frequent terms, which is noted $\mathcal{FT}_p$. The idea is to compute Δ, the minimal total distance to transform the sentence into the pattern. Thus, our metric is an adaptation of the Levenshtein distance [123]. However, we do not compute a number of operations (delete, transform or add a term) between two words or characters, but rather we evaluate the semantic distance between two words. Given the i^{th} word w_p^i associated to the tag t_p^i in the pattern p and the word w_s^j with the tag t_s^j in the sentence s, we compute their semantic distance $semdist(w_p^i, w_s^j)$ using the following formula:

$$semdist(w_p^i, w_s^j) = \begin{cases} 0.0 & \text{if } w_s^j \in \mathcal{FT}_p \\ resnik(w_p^i, w_s^j) & \text{if } w_s^j \notin \mathcal{FT}_p \text{ and } t_p^i = t_s^j \\ 1.0 & \text{otherwise} \end{cases}$$

Namely, the distance is equal to 0 if the word in the sentence is a frequent term. POS-tagged terms whose tags are identical (e.g., two verbs) have a similarity obtained by applying the Resnik distance in Wordnet [170]. Else, words with different tags have the maximal distance. The minimal total distance $\Delta(p,s)$ between a pattern and a sentence is computed by the matrix algorithm of the Levenshtein distance[5] using the semantic distance for all pairs of words. This distance is normalized in the range $[0,1]$ with Formula 8.2 which assesses the similarity $pattsim$ between the pattern p and the sentence s.

$$pattsim(p,s) = \frac{1}{1+\Delta(p,s)} \tag{8.2}$$

Our metric is flexible because extra or missing words in a sentence do not significantly affect the similarity value. Similarly, less important words are mainly compared on their nature (POS-tag). Finally, we can select the sentences which are modeled by a pattern according to a threshold.

Named Entity Recognition (NER). When a sentence in a document corresponds to a pattern, our approach needs to identify and extract entities contained in the sentence. Thus, the NER issue is crucial as it determines (part of) the output. Indeed, it is necessary to correctly identify the (labels of) entities in the sentence based on a pattern. To solve this issue, we rely on the formalism of our patterns: since they have been POS-tagged, the tags serve as a delimiter and may constraint the candidate entities.

8.4.2 Discovering the Type of Relationship

In this first use case, the user provides two labels (representing an entity) and the goal is to determine the possible type(s) of relationships between these two entities.

[5] `http://j.mp/Levenschtein` (Last checked October 2012)

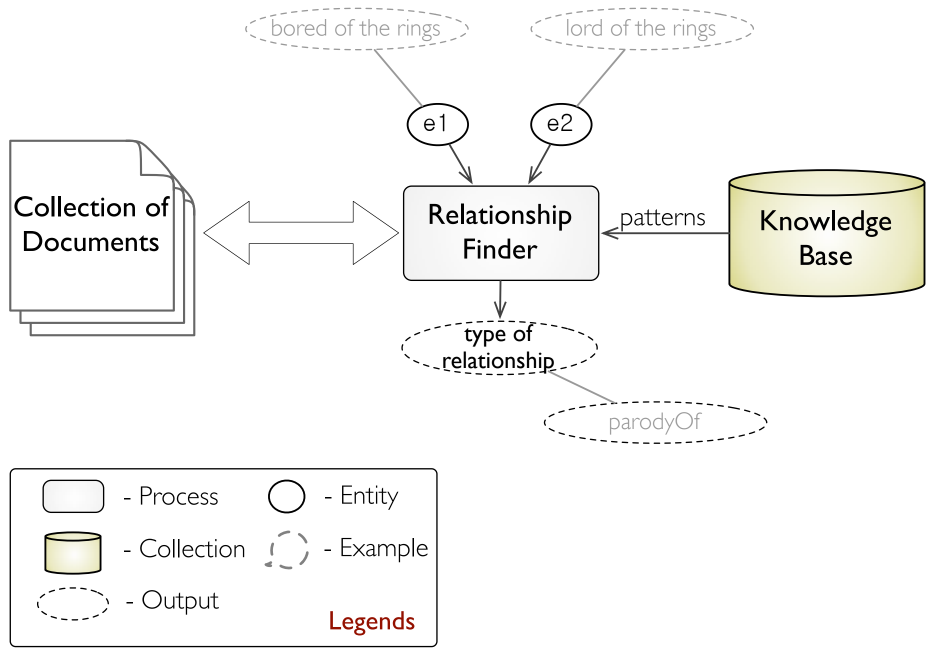


Figure 8.3: Discovering the Type of Relationship.

This scenario is illustrated with Figure 8.3. The two entities are first extended to obtain their alternative labels (see Section 8.3.1). A set of documents is analyzed to extract all sentences which contain one label of each entity, and these sentences are then compared to all patterns stored in SPIDER's knowledge base (see Section 8.4.1). If a sentence is similar to a trustable pattern (with a sufficient confidence score), then the type of relationship corresponding to this pattern is proposed to the user. Note that if there is no trustable pattern in the knowledge base, the user has the possibility to provide training data to the system (see Section 8.6.2).

8.4.3 Searching for an Entity

This second use case, illustrated with Figure 8.4, aims at discovering an entity to satisfy the rest of the relationship, e.g., *the list of shops which sell Lord of The Rings movies* or *the list of authors who wrote fantasy books*. Given an entity and a type of relationship, SPIDER extracts the frequent terms of the corresponding patterns from the knowledge base and it extends the entity by finding its alternative labels. Then, it associates all possible pairs composed of a frequent term and an alternative label

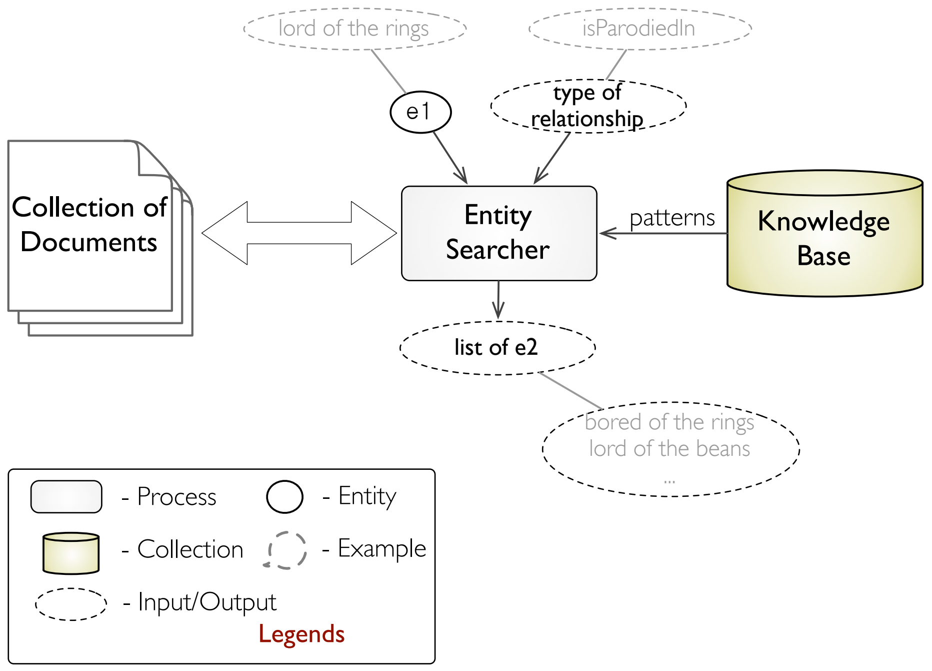


Figure 8.4: Searching for an Entity.

for querying the collection. The sentences in these documents containing both a frequent term and an alternative label are evaluated with the patterns, and in case of a sufficient similarity between them, the NER component tries to detect the second entity in the sentence (see Section 8.4.1).

8.4.4 Discovering New Examples

The last use case depicted in Figure 8.5 refers to the discovery of new examples. This step is at the basis of the *never-ending* feature which enables the feeding of the knowledge base with additional training data for generating new patterns. SPIDER selects in the knowledge base the patterns of a given type of relationship. It retrieves a set of documents by querying each de-tagged pattern (or only their frequent terms). We compute the similarity between a pattern and the sentences of the documents which include a frequent term of this pattern. If the sentence is modeled by the pattern, then we apply the NER techniques for discovering the two entities. Note that each discovered example has a confidence score, which is computed with the same formula as in Section 8.3.4 for the confidence of a pattern, except that sup_e replaces

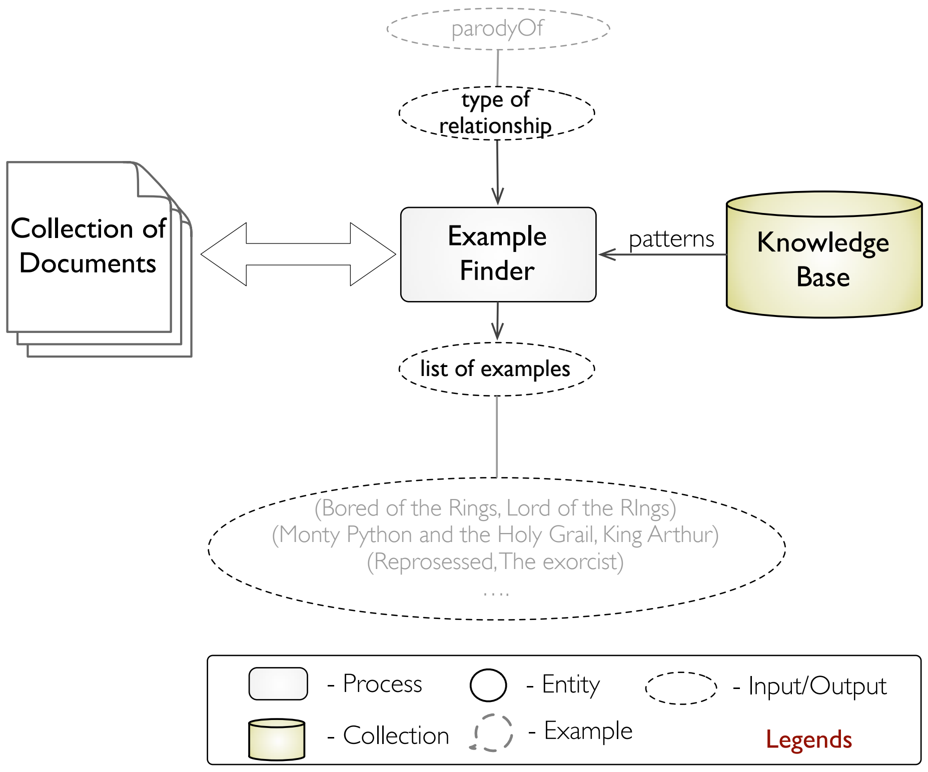


Figure 8.5: Discovering New Examples.

sup_p and it indicates the number of patterns which discovered this example. Examples with a very low confidence are discarded while others are stored in the knowledge base.

8.5 Scalability

Large scale indexing and searching requires a smarter way to arrange a large collection rather than a sequential access to the collection. This problem is further exacerbated by additional processing, such as gathering the anchor texts, calculating PageRank and SpamScore and identifying important concepts (e.g. title, headlines, boldfaced/italicized text). The idea is to obtain these features by parallelizing and performing the task on partitions of documents.

SPIDER gathers the anchor texts as we believe anchor texts are an interesting feature of Web documents. An anchor text is a clickable text in a hyperlink which provides an independent descriptive label of the target Web document. It can be seen as a mini-summary of the target document. Anchor texts are traditionally given a higher weight by search engines, because they are relevant for the so-called *target document*. Furthermore, we store all above mentioned important concepts which affect the weight of the document along with the PageRank and SpamScore. To retrieve and rank the relevant documents for a given query, SPIDER computes and aggregates the **relevance score** for these factors (anchor texts, title, headlines, boldfaced text and content). This score is based on tf-idf and it is used to evaluate the relevance of the document with respect to the query.

In order to efficiently process documents, we distribute the jobs into several machines. MapReduce inspired techniques have been popular to tackle such tasks. Therefore, the collection is indexed with Hadoop enabling efficient indexing and searching. To compute statistics, SPIDER makes use of Pig[6] which is a high level platform for analyzing a large collection of data. The MapReduce programming model [63] adopted in SPIDER enables scaling out the knowledge extraction capabilities of SPIDER. Originally proposed by Google engineers in 2004, MapReduce is commonly used for processing large data sets in a distributed computing environment, usually cluster of commodity machines. Shortly, it defines two functions **map** and **reduce** and computations are based on key value pairs. The map function takes a value and outputs *key value* pairs. The reduce function accepts a *key* and a *list of values*. Since documents are self-contained chunk of information, parallelizing the parsing job can be achieved easily. In SPIDER pattern generation, mappers accept the document identifier i and the document itself d as shown in the Algorithm 2 (line 1).

The reducers then emit intermediate occurrences of the patterns if this occurrence is above a given threshold. The second step in SPIDER is generating examples. The generation of examples is illustrated in the Algorithm 3. The mapper takes again a document as input and generates examples using all the patterns discovered so far. The reducers then combine all the examples for a particular pattern.

Additionally, we propose a document partition to incrementally provide results on a subset of the collection. The general idea is that highly ranked documents should be a better source for obtaining patterns than those with lower PageRank, SpamScore and relevance score values. The size of a partition depends on the quality of the obtained patterns and examples. SPIDER is able to automatically tune the ideal size

[6]http://pig.apache.org (Last checked October 2012)

Algorithm 2 MapReduce functions for discovering patterns.

```
1: function MAP($i, d$)
2:     $P_i \leftarrow discover\_patterns(d)$
3:     for all $p \in P_i$ do
4:         $emit\_intermediate(p, 1)$
5:     end for
6: end function

7: function REDUCE($p, V$)
8:     $occurrence \leftarrow 0$
9:     for all $v \in V$ do
10:        $occurrence \leftarrow occurrence + v$
11:    end for
12:    if $occurrence > THRESHOLD$ then
13:        $emit(p, occurrence)$
14:    end if
15: end function
```

of a partition. Initially, the documents are sorted by their PageRank, SpamScore and the relevance score as described above. The top k documents are selected for analysis from the head of the ranked list of documents. For the i-th round, the cursor is moved to the range $[k, i \times k]$ and the documents in that range are picked out for analysis. Furthermore, when there are too few patterns discovered for two given labels out of these initial set of documents, the partition size is subsequently adjusted to a higher number. The combination of scores as well as the partitioning mechanism makes obtaining the URLs of the documents and their content for a given query fast, i.e., it only requires a few seconds. This efficiency is demonstrated in Section 8.6.3.

8.6 Experiments

This evaluation section first describes the protocol and then the experiments in terms of quality and performance.

Algorithm 3 MapReduce functions for generating examples.

```
1: $P \leftarrow initialize\_patterns()$
2: function MAP($i, d$)
3:     for all $p \in P$ do
4:         $X \leftarrow generate\_examples(p, d)$
5:         for all $x \in X$ do
6:             $emit\_intermediate(p, x)$
7:         end for
8:     end for
9: end function

10: function REDUCE($p, X$)
11:     $X_p \leftarrow \emptyset$
12:     for all $x \in X$ do
13:         $add(X_p, x)$
14:     end for
15:     $emit(p, X_p)$
16: end function
```

8.6.1 Protocol

Experiments were performed using server class machines with Intel Core i7 processor and 24GB of RAM memory. Our collection of documents is the English portion of the ClueWeb09 dataset (around 500 million documents). For the components, we have used the contextual strategy, with the Maxent POS-tagger provided with StandfordNLP[7]. The NER component is based on the OpenNLP toolkit[8].

Evaluation is performed using well-known metrics such as precision (number of correct discovered results divided by the total number of discovered results). The recall can only be estimated since we cannot manually parse the 500 million documents to check if some results have been forgotten. However, we show that the number of correct results increases over time.

[7]http://nlp.stanford.edu/software/CRF-NER.shtml (Last checked October 2012)
[8]http://opennlp.apache.org/ (Last checked October 2012)

8.6.2 Quality Results

In this section, we present our results in terms of quality of label extension, relationship discovery and comparison with state of the art baseline knowledge extraction tools. The evaluation of relationship discovery depends on the quality of generated patterns and hence we present this evaluation rather than the pattern generation itself. We do not evaluate pattern generation, because the evaluation of discover relationship depends on pattern generation anyway.

Extending Entities

The first experiment deals with the evaluation of the SPIDER's entity extension mechanism. A ground truth is required for this type of evaluation. In our case, this ground truth has been semi-automatically constructed by mining labels of each entity from DBpedia and Freebase ($dbpedia-owl : wikiPageRedirects$ and $freebase : common.topic.alias$). It has been manually checked and cleaned[9]. The ground truth

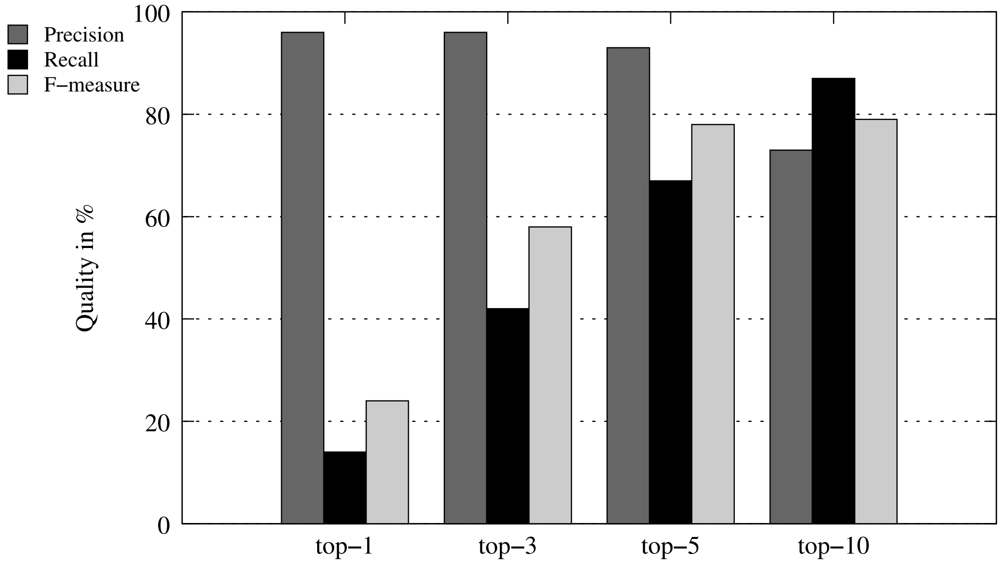


Figure 8.6: Quality Results of Label Extension.

has been limited to 10 labels per entity. However, not every entity has many labels. Thus the number of labels in the ground truth is equal to 4018. The precision at

[9]For example, *History_of_Oslo* is a DBpedia redirection for the entity *Oslo*, but it is arguably not considered as label

top-k is computed as the ratio between the number of correct labels discovered at top-k and the number of discovered labels at top-k. However, the recall is computed differently[10]: it represents the ratio between the number of correct labels discovered at top-k and the total number of expected labels (4018). F-measure is harmonic mean between precision and recall.

Figure 8.6 depicts the values of the three metrics for different top-k. We first note that SPIDER achieves a high precision at top-1 (96%). The correct labels discovered at top-1 only represent a small fraction of the expected labels (4018), thus resulting in a low recall value (14%). Yet, this recall value increases when considering more labels (at top-3, top-5 and top-10) to reach 86%. This improvement does not have a negative impact on the precision at top-3 and top-5 (scores above 90%) but it slightly decreases at top-10 (73%). This decrease is mainly attributed to the false positives. However, it has previously been shown [184] that these false positives can be filtered out with a similarity threshold. For the next experiments, we have tuned SPIDER so it returns 5 labels for an entity. Indeed, the top-5 ensures an acceptable tradeoff between precision and recall.

Relationship Discovery

We evaluate the quality obtained by SPIDER when running the first use case (Section 8.4.2). Given two labels (representing entities), we search for the correct type of relationship which links them. To fulfill this goal, we have manually designed a set of 200 relationships, available at this URL[11].

Note that the type of relationship associated to each example is the most expected one, but several types of relationship are possible for the same example. Table 8.1 provides a sample of examples (e.g., *Obama, Hawai*) and some candidate types of relationship discovered by SPIDER (e.g., *birthplace*). A **bolded** type of relationship indicates that it is correct for this example. A second remark about our set of relationships deals with the complexity of some relationships (e.g., <*cockatoo, tail, yellow*>). The last column shows the initial confidence score computed for the candidate relationship.

The quality is measured in terms of precision at different top-k. Indeed, SPIDER outputs a ranked list of relationship types according to their confidence scores. In ad-

[10]Obtaining a “local recall” at each top-k implies that the labels of the ground truth are ranked for each entity.

[11]http://www.idi.ntnu.no/~takhirov/spider/rels.txt (Last checked April 2013)

Example	Discovered type of relationship	Confidence score
Obama, Hawai	**birthplace**	0.42
	senator	0.31
	president-elect	0.18
cockatoo, yellow	amazon	0.32
	parrot	0.31
	tail	0.16
eucalyptus, myrtaceae	plant	0.51
	family	0.43
	specie	0.27
Bartolomeo Cristofori, piano	**inventor**	0.60
	instrument	0.43
	maker	0.19
Dave Grohl, Nirvana	Cobain	0.34
	band member	0.27
	drummer	0.16
Bored of the Rings, Lord of the Rings	**parody**	0.53
	links	0.24
	Middle-Earth	0.23

Table 8.1: Examples of Discovered Types of Relationship and Confidence Scores.

dition, our approach is able to run with or without training data. Thus, we have tested the system when a few training data have been provided. Using 1 training example means that the system has randomly selected 1 correct example for bootstrapping the system. Experiments with the training data are based on cross-validation and 5 runs reduce the impact of randomness.

The manual validation of the discovered relationships has been performed by 8 people. This manual validation includes around 3000 invalid relationships and 600 correct ones, and it facilitates the automatic computation of precision. In addition, we are able to estimate the recall, i.e. to evaluate the number of correct types discovered during a run w.r.t. all validated types. This is an estimation because there may exist more correct types of relationship than the ones which have been validated. Besides, a discovered type may have a different spelling from a validated type while both have the same meaning, thus decreasing the recall.

Figures 8.7 and 8.8 respectively depict the average precision and the average recall for the 200 relationships by top-k and by number of provided training data. We first notice that SPIDER achieves low quality without training data (precision from 40% at top-1 to 30% at top-10). The estimated recall values are also quite low at top-1 because there is an average of 3 correct types of relationships for each example. The top-3 results are interesting with 5 training data: the precision is acceptable (more than 80%) while the recall value (32%) indicates that one type of relationship out of three is correctly identified. Since our dataset contains complex relationships, this configuration is promising for bootstrapping the system. Precision strongly decreases at top-5 and top-10, mainly because each example roughly includes 3 types. However, the top-5 and top-10 recall values indicate that we discover more correct examples. Finally, we notice that providing a few training data (5 examples) enables at least a 10% improvement both for precision and recall. This remark is important since our approach aims at running perpetually by reusing previously discovered examples and patterns.

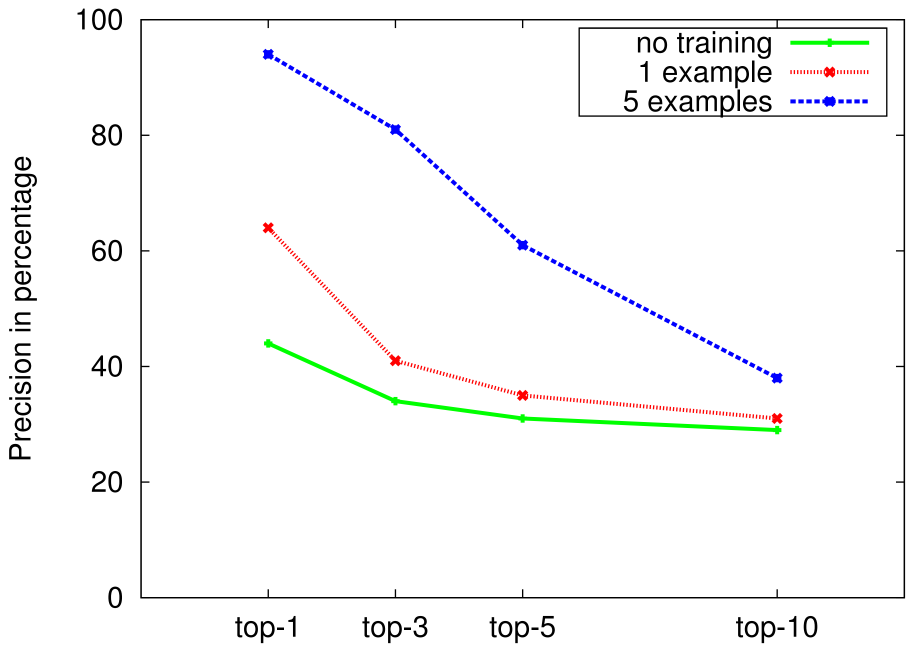


Figure 8.7: Precision according to top-k and Training Data.

The quality results are subject to the complexity of the set of relationships, since we have selected some complex ones to discover, such as <*"cockatoo", "tail", "yellow"*>. Other problems of disambiguation occurred, for instance the example *"Chelsea", "London"* mainly returns types of relationships about accommodations because Chelsea is

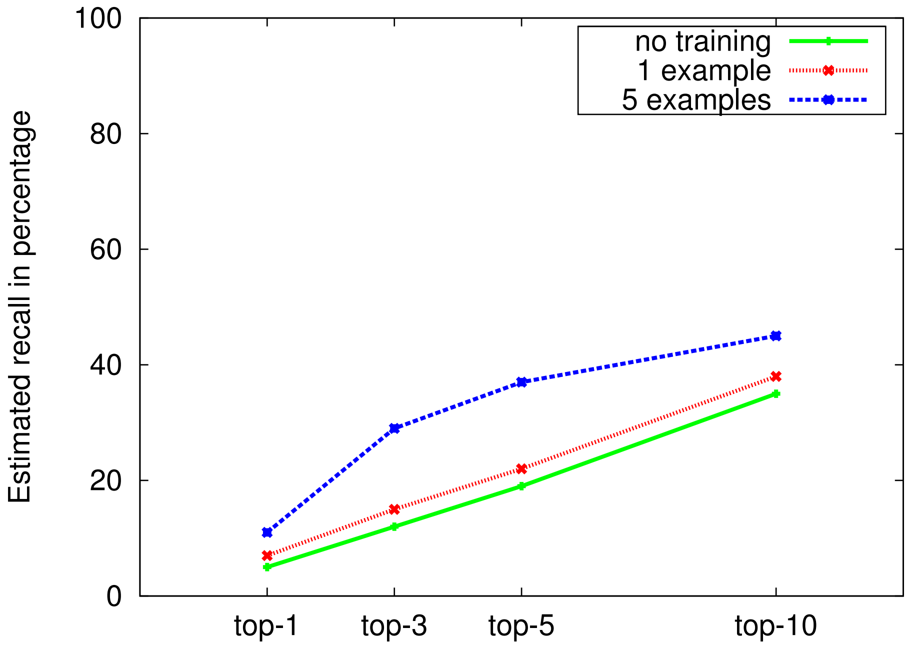


Figure 8.8: Estimated Recall according to top-k and Training Data.

identified as a district of London and not as the football team. Contrary to other systems, SPIDER does not use counter seeds. These incorrect examples could help for filtering the output.

Baseline Comparison for Quality

A final experiment aims at comparing our system with two other approaches, ReadTheWeb (NELL) [43] and Prospera [153], both described in Section 3.2. These two approaches have been chosen as baseline because the dataset along with the results are available online. An evaluation of these tools is described online[12], which corresponds to our third use case (Section 8.4.4). Since the seed examples are available, we have used them as training data. Table 8.2 summarizes the comparison between the three systems in terms of estimated precision, as explained in the experiments reported in [43, 153]. Similarly to Prospera and ReadTheWeb, our precision is an estimation due to the amount of relationships to validate. Namely, 1000 random types have been validated for each relationship. The average precision of the three systems is the same (around 0.91). However, the total number of facts discovered by

[12] http://www.mpi-inf.mpg.de/yago-naga/prospera/

Relation	**RTW**	**Prospera**	**SPIDER**
AthletePlaysForTeam	**1.00** (456)	0.82 (14,685)	0.80 (15,234)
TeamWonTrophy	0.68(397)	0.94 (98)	**0.96** (92)
CoachCoachesTeam	**1.00** (329)	0.88 (1,013)	0.90 (1,629)
AthleteWonTrophy	n/a	0.92(10)	**0.94** (124)
AthletePlaysInLeague	n/a	0.94 (3,920)	**0.95** (4,211)
CoachCoachesInLeague	n/a	**0.99** (676)	0.89 (741)
TeamPlaysAgainstTeam	**0.99** (1,068)	0.89 (15,170)	0.93 (15,729)
TeamPlaysInLeague	n/a	0.89 (1,920)	**0.95** (2,409)
TeamMate	n/a	**0.86** (19,578)	0.84 (31,752)

Table 8.2: Estimated Precision values (with Number of Discovered Facts) obtained by ReadTheWeb (RTW), Prospera and SPIDER.

SPIDER (71,921) is 36 times higher than ReadTheWeb (2,112) and 1.3 times higher than Prospera (57,070), outperforming both baselines.

Prospera provides slightly better quality results than our approach on *AthletePlaysForTeam* relation. However, several factors have an influence on the precision results between Prospera, ReadTheWeb and SPIDER First, Prospera is able to use seeds and counter seeds while we only rely on positive examples. On the other side, Prospera includes a rule-based reasoner combined with the YAGO ontology. Although SPIDER does not support this feature, the combination of POS-tagged patterns and NER techniques achieves outstanding precision values.

8.6.3 Performance

To assess the efficiency of SPIDER, we performed scalability experiments in terms of processing documents (retrieval and pre-processing) and discovering relationships (extraction).

Scalability for Processing Documents

Since knowledge extraction systems deal with large collections of documents, they need to be scalable. Figure 8.9 depicts the performance of SPIDER for retrieving and preprocessing (i.e., clean up the header, remove html tags) the documents. The total time (sum of retrieval and preprocessing) is also indicated. Although there is no caching, the total time is not significant for collecting and preprocessing one million documents (around 40 seconds). Note that in real cases, a conjunctive query composed of two labels rarely returns more than 20, 000 documents. The peak for retrieval at 600, 000 documents is due to an overhead processing from the thread manager. Increasing the number of threads above 400 leads to higher thread switching latency while decreasing this number only reports the peak earlier during the process. This issue could be simply solved by dispatching this task on different servers.

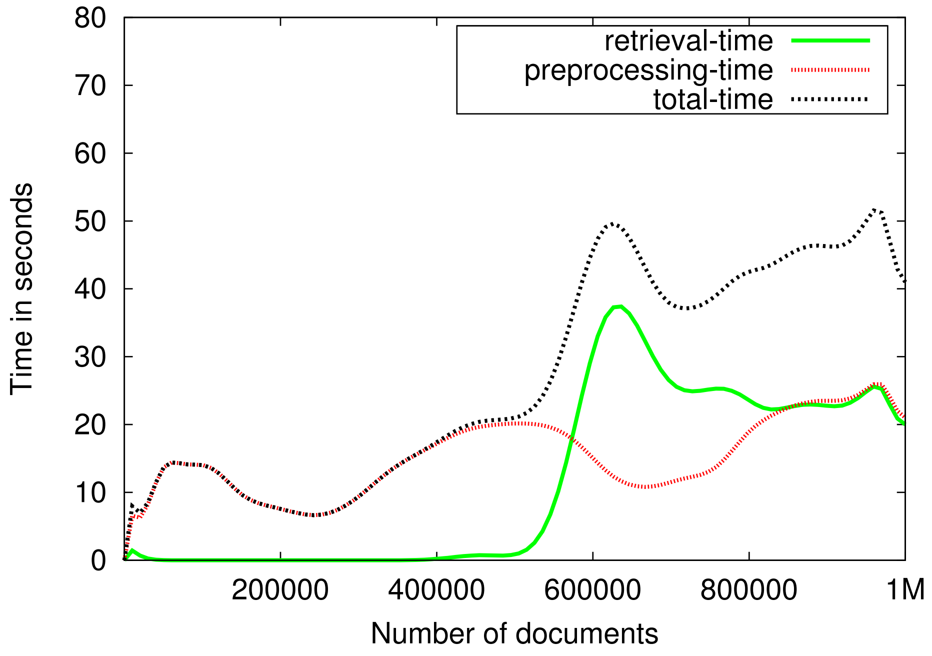


Figure 8.9: Retrieval and Preprocessing Performance.

Scalability for Discovering Relationships

Next, we study the performance of the whole process, from the retrieval of the documents to the generation of the patterns and the discovery of the type(s) of relationship. Since this experiment is query-dependent, we have chosen a query which returns a large amount of results (*"France"* and *"Paris"*). We used the most costly strategy, the contextual one, and without any training data. Figure 8.10 shows the performance in terms of elapsed time (on the left axis), number of candidate patterns and number of patterns (both on the right axis) according to the number of documents. We notice that the elapsed time is linear (30 seconds with 1000 documents to 390 seconds for

500000 documents). The number of extracted candidate patterns heavily increases with the number of documents, while the number of generated patterns is limited (from 2982 with 1000 documents to 17024 with 500000 documents). Among these generated patterns, many are discarded due to their low confidence scores.

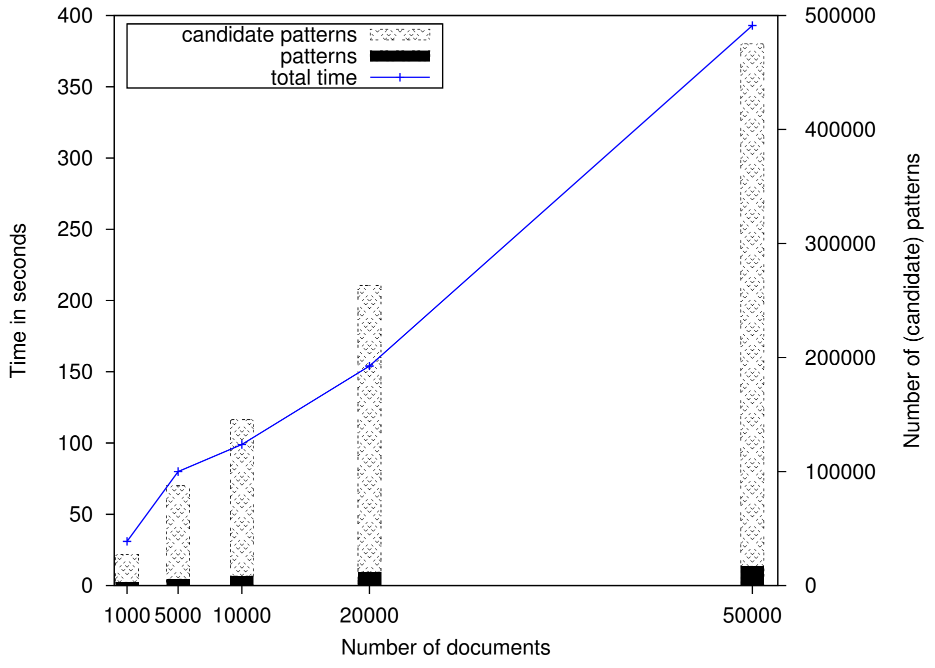


Figure 8.10: Query Performance

8.7 Summary

In this chapter, SPIDER has been discussed in detail, an approach to automatic extraction of binary relationships from large text corpora. The main advantage of SPIDER is to guarantee both a better quality and a strong improvement in terms of performance, thus providing new opportunities for discovering relationships at large-scale and dynamic environments. Finally, we have demonstrated the feasibility of SPIDER at Web-scale.

Additionally, to increase the quality and the consistency of generated facts, systems may either be based on general ontologies such as Yago [153] or on logical rules associated with a SAT solver [181]. The last trend in this domain deals with Open IE, in which the large scale aspect of the Web is taken into account [73]. However, none of these works clearly aim at building a semantic knowledge base, thus there is no linking with the LOD cloud.

Contrary to the oldest systems which include hard representations of patterns, Prospera and SPIDER includes a more flexible definition of the patterns, so that similar patterns can be merged. In addition, the patterns are at the sentence level, which means that the texts before, after and between the entities are considered. Most systems are not extensible while the SPIDER's architecture integrates individual operations and strategies for the extraction process. Using various strategies reduces the chances of an infinite loop where no more pattern and no more examples can be discovered. Our confidence score for a pattern or an example takes into account crucial criteria such as provenance. In addition, the support and the occurrence scores are correlated within the same type of relationship, thus the confidence in a pattern or an example may decrease over time. To the best of our knowledge, none of these approaches deal with the issue of identifying the alternative labels of an entity.

Although all approaches propose to discover new examples, only a few of them, including SPIDER, provides different applications such as the discovery of a type of relationship or the search for the second entity of a relationship. Finally, the approach presented in this chapter is scalable with document partitioning based on smart sorting using the SpamScore, PageRank and relevance score of the documents.

A deeper understanding of the impact of parameters and components, such as strategies, iterations or the extension process would clearly be an interesting issue to explore. Moreover, a rule-based component could be integrated not only to check the consistency of the knowledge base, but also to infer new relationships.

The next chapter presents KIEV– a prototype that supplements SPIDER. In particular, KIEV includes evidence based techniques to verify extracted facts.

9 Verifying Extracted Entities and Relations

9.1 Introduction

In order to improve the extraction results, in this chapter, we propose to tackle the challenge of verifying extracted relations. Our approach, KIEV[1] first extracts examples for a given relationship from textual documents by further developing the extraction approach proposed through the SPIDER prototype. Indeed, some relationships are rarely encompassed in the structured data sources, but they can be found in textual documents (such as the Web). Mining these relationships with a pattern-based technique involves the discovery of a large amount of examples. Thus, a verification of these examples is performed at two levels: (i) the type of relationship is checked with a machine learning approach and (ii) the extracted entities are matched to LOD for both verification and integration purposes. In addition to these challenges, our approach KIEV should perform reasonably well in terms of efficiency at the Web scale since every page is a potential source of examples and good patterns.

The rest of this chapter is organized as follows. Section 9.2 introduces the formalization of our problem and provides an overview of KIEV. Section 9.3 covers the first part of our approach, the discovery of examples by using patterns, while Section 9.4 and 9.5 focus on the evidence-based verification of these examples. Our experiments are detailed in Section 9.6. Finally, we provide a summary of chapter achievements in Section 9.7.

[1]KIEV – **K**nowledge and **I**nformation **E**xtraction with **V**erification

9.2 Overview

Our goal can be seen as the creation of a knowledge base of entities and relationships. Simply assuming the existence of a repository of domain entities would limit our approach. Rather, we extract entities from the textual documents, and as a consequence, our approach should also work with entities which have been previously identified (i.e., from a repository). A relationship is defined as a triple <$entity_1$, *type-of-relationship*, $entity_2$>. As an example, considering the 2006 *"The Departed"* movie directed by Martin Scorsese as a remake of the Andrew Lau's *"Infernal Affairs"* from 2002, the example would be represented as <*"Infernal Affairs", hasImitation, "The Departed"*>.

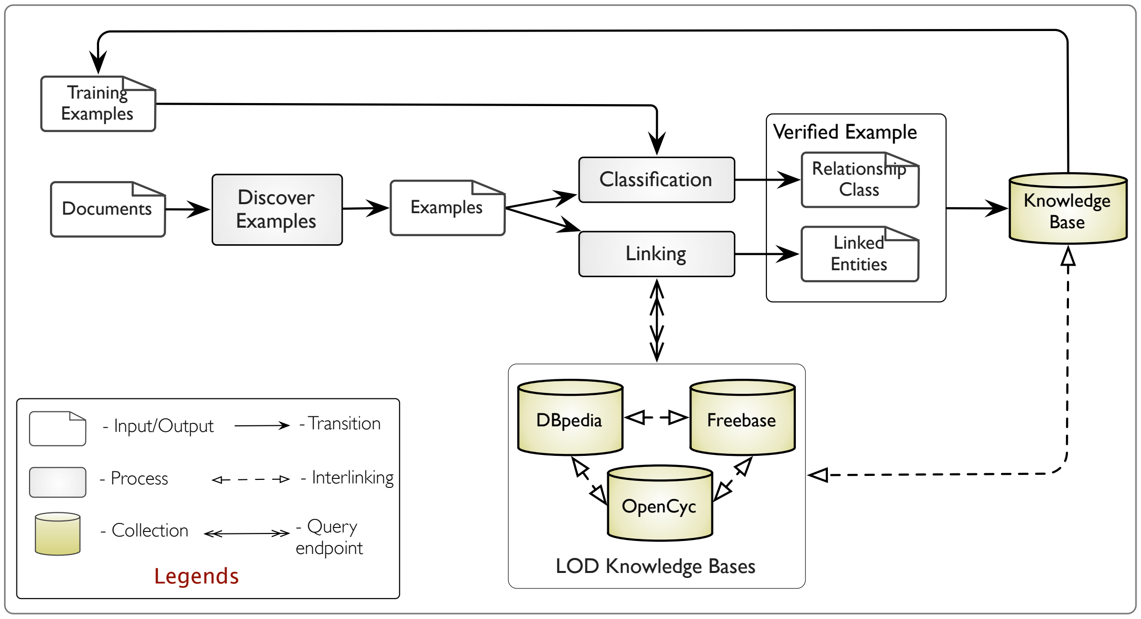


Figure 9.1: Overview of KIEV.

Figure 9.1 depicts the global overview of KIEV. Given a type of relationship, KIEV requires a collection of documents and a few training examples (verifying the types of relationship) to bootstrap a possible infinite loop. The first step consists of discovering examples from the textual collection (see Section 9.3). It is based on semantic tagging which combines Named Entity Recognition and Part of Speech tagging, and it generates many examples for the concepts contained in a sentence. Thus, a verification of the relevance for these examples is performed with two other processes.

The former checks if the extracted entities are effectively related with the type of relationship using a machine learning classifier (see Section 9.4). The latter process links both extracted entities of an example to their corresponding entities on the LOD cloud (see Section 9.5). Once an example is verified, it can be used as a training example to improve the classifier, but also to reinforce the confidence score of a pattern during the discovery process.

9.3 Discovering Examples

The core idea of our approach is to process the input as a stream of documents and to iteratively update our semantic knowledge base of entities. In this section, we describe the first part of our approach – discovering examples. An example for a given type of relationship is composed of two entities (e.g., for *imitation* type of relationship, an example is <*"Infernal Affairs", "The Departed"*>). Figure 9.2 provides the big picture of the example discovery workflow, whose goal is to generate a set of examples. Each process in the workflow of discovering examples is presented below.

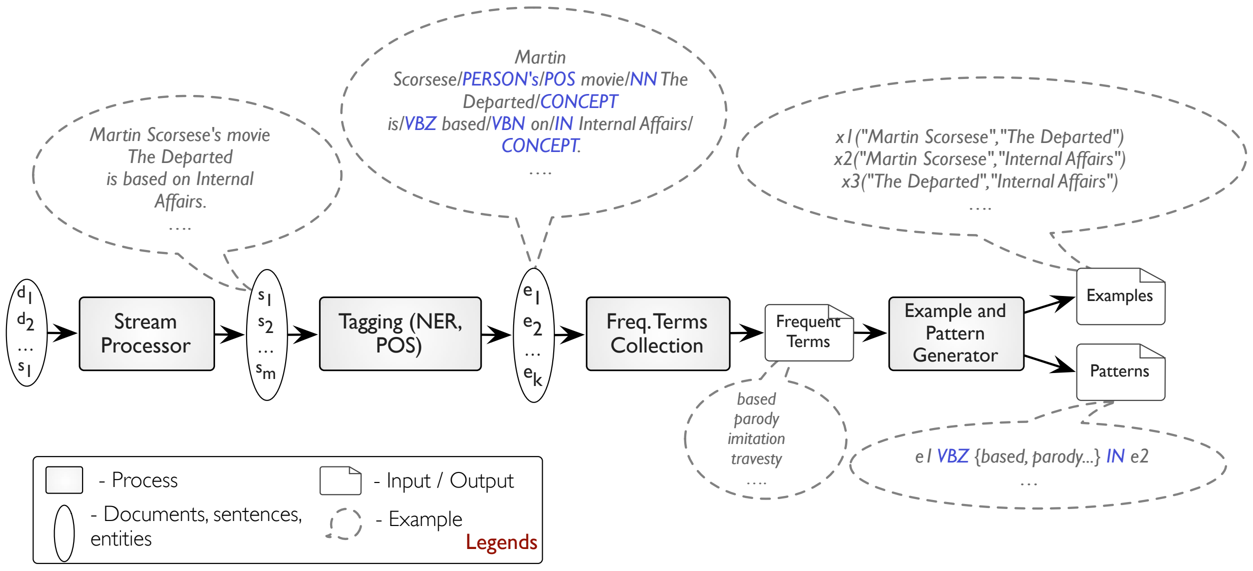

Martin Scorsese's movie The Departed is based on Internal Affairs. ....
Martin Scorsese/PERSON's/POS movie/NN The Departed/CONCEPT is/VBZ based/VBN on/IN Internal Affairs/CONCEPT. ....
x1("Martin Scorsese","The Departed") x2("Martin Scorsese","Internal Affairs") x3("The Departed","Internal Affairs") ....
d1 d2 ... sl
Stream Processor
s1 s2 ... sm
Tagging (NER, POS)
e1 e2 ... ek
Freq. Terms Collection
Frequent Terms
Example and Pattern Generator
Examples
Patterns
based parody imitation travesty ....
e1 VBZ {based, parody...} IN e2 ...
- Process
- Input / Output
- Documents, sentences, entities
- Example
Legends

Figure 9.2: Workflow of Example Discovery Process.

9.3.1 Stream Processing

Stream processor (SP) accepts as an input documents in textual form. The first task the SP performs is to pre-process the input. For example, this task may involve cleaning the html documents for tags, removing headers from emails, etc. At this point, we are interested in only obtaining text regardless of the quality. Each document $d \in \mathcal{D}$ is segmented into a list of sentences such that $d = \{S_i \mid i = 1 \dots N\}$ where N is the number of sentences. A sentence S_i is discarded if S_{i-1} and S_{i+1} contain no entities. This is because S_i may contain a personal pronoun referring to the previous sentence, e.g. *"**Martin Scorsese** is an American film director. **He** is the creator of Taxi Driver."*. Additionally, the sentences are filtered out to eliminate those that were likely to be noisy (broken and invalid sentences) and not useful for example discovery (e.g., non-English sentences, sentences missing verb, sentences with only uppercase letters or only with lowercase letters, sentences without capital letters, etc.). The next step deals with the semantic tagging of the selected sentences with named entity recognition (NER) and part-of-speech (POS) tags.

9.3.2 Tagging

For each sentence $s \in S_i$, named entity recognition is performed to detect the set of entities $\mathcal{E}$ (person, location, organization, dates, etc.). Consider a document containing the following sentence: *Infernal Affairs was followed by a 2006 American remake by Martin Scorsese entitled The Departed*. From this sentence, two *concepts* are detected and one *person*. Traditionally, NER is focused around the detection of common entities such as people, organization or geographic location. However, the recognition of domain specific entities poses a particular challenge because the NER tools usually require training examples for the types of entities to recognize. In our context of textual documents from the Web, providing such examples is not possible.

To avoid missing entities, a POStagger is first applied on all sentences. Our assumption is that entities are POStagged as *"noun"*. Thus, we consider that all nouns in the sentences are entities. A NER tool can confirm some of these entities. Although this assumption implies the identification of many incorrect entities, the next steps are in charge of discarding those irrelevant entities. The output of the semantic tagger is a set of semantically and structurally tagged sentences, from which we can extract frequent terms.

9.3.3 Frequent Terms Collection

Terms that appear frequently in the same sentence with a pair of entities are likely to be highly relevant to the pair of entities. For example, in the sentence *"Martin Scorsese's movie The Departed is based on Internal Affairs"*, frequent terms are *movie* and *based on* because they appear frequently together with the entities in the sentence.

In order to collect these frequent terms, all possible word n-grams are first identified in the sentence s. The top thousand most common words on the Web[2] are excluded and cannot be part of frequent terms. Then, the sentence s is splitted into a set of words. A list of n-grams is constructed out of this list. After the list of n-grams has been obtained, we look up Wordnet lexical database to obtain the list Φ of semantically related words. These words are grouped into unordered sets (synsets). Stopwords (e.g., "the", "a", "but" etc.) are removed and stemming is performed. The following Wordnet relations are used:

- synonymy (e.g., "writer" and "novelist"), words that denote the same concept and are interchangeable in many contexts.
- hyponym, a word whose semantics are included within that of another word[3], e.g., "The Departed is a movie".

Since the synsets obtained from Wordnet have a shared information content, i.e., hierarchy of is-a concepts, this list of semantically similar words can be larger than desired. Thus, to control the level of granularity of this list of concepts, we employ the Resnik similarity to prune those that are below a given threshold [170]. This similarity measure is applied between the segmented n-grams and each of the synsets in Φ. For example, the distance between "novel" and "book" is 0.29.

These frequent terms are generated for different objectives such as the classification of examples through features, but also to generate the examples as explained in the next part.

9.3.4 Example and Pattern Generator

Having obtained the lists of named entities and frequent terms, a set of candidate examples is built. One of our goals is to populate a knowledge base that can serve as

[2]This list is available from Microsoft Web N-gram Service: http://bit.ly/bFKSxz
[3]This is similar to *is-a* relationship

a repository of distinct entities. First, a set of unique pair of entities Θ is constructed such that $\Theta = \{(e_i, e_j) | e_i \neq e_j, e_i \in \mathcal{E}, e_j \in \mathcal{E}\}$. At first glance, it appears that we generate overly many examples and this most likely leads to a fair number of false positives. But we will show in section 9.4 that our classification approach effectively discards these irrelevant examples. Our basic assumption with generating so many examples is to reduce the likelihood of low recall.

At this time, we can generate patterns based on the information from the frequent terms collector. That is, we mask the named entities (e.g. "Infernal Affairs" $\Rightarrow e_1$, "The Departed" $\Rightarrow e_2$). The idea is to obtain entity and word independent patterns, as shown in the Figure 9.2. At the end of each iteration, a list of patterns is generated from the candidate examples. If the pattern had been generated before, its statistics are updated from the current iteration. For patterns $\{p_1, \ldots, p_n\}$, we compute the pattern similarity using the Levenshtein distance and those above a given threshold are merged into a set of patterns P_p. By now, we know the amount of patterns generated in this iteration (P_i). We note the list X_p of examples that support this pattern. The patterns generated at iteration i are ranked according to the following scoring function:

$$score(p) = \frac{\alpha \frac{occ(p)}{i} + \beta \frac{|P_p|}{|P_i|} + \gamma \frac{|X_p|}{|X|}}{\alpha + \beta + \gamma}$$

where $occ(p)$ is the number of iterations this pattern has been discovered out of total number of iterations i. X denotes the number of total examples in the system. The scores are normalized in the range $[0, 1]$. The patterns generated during this iteration will be used to discover new examples in the next iteration. These patterns will also be used as features during the classification process.

As previously explained, all of the examples discovered so far may not be correct. In the next section, we will show how a classifier effectively discards false positives.

9.4 Classification

The first part of the verification is to check that the candidate entities (represented with a label) are related with a type of relationship. Indeed, a sentence may contain different entities and the discovery process generates in that case incorrect examples, mainly because of the pattern-based matching. The classification aims at discarding these incorrect examples without prior knowledge about the two entities. To fulfill this goal, the verification process can be seen as a **classification problem** [149]. Given a set of features (properties), the idea is to find the correct class for a given example (extracted from a sentence). Each class represents a type of relationship (e.g., *imitation, adaptation*). For instance, the example *(James Cameron, Avatar)* should be classified in the class *creatorOf*. A specific class named *unknown relationship* is added as a garbage class to collect all incorrect examples or those that cannot be classified in another class. To select the correct class for an example, a classifier is trained using training examples, i.e., examples for which the correct class is already known. Although the training process depends on the type of classifier (e.g., decision tree, Bayes network), it mainly consists of minimizing the misclassification rate when classifying the training examples according to the values of their features [149]. To compute these values, each training example is used as a query over the document collection and all sentences containing the two entities of the example are analyzed given the following features: the frequency and the presence of any frequent terms (e.g., *parody*), the length and structure of the best-ranked pattern which generated the example (see Section 9.3.4), the average spamscore of the documents from which the pattern is extracted [53]. Note that this paper does not aim at designing a new classifier, but we rather use existing ones from the Weka environment [85]. More formally, an example $x \in \mathcal{X}$ is defined by a set of features $\mathcal{F}$. We note the set of training examples $\mathcal{T}$, with $\mathcal{T} \subseteq \mathcal{X}$. Each example can be assigned a class $c \in \mathcal{C}$. Given a (type of) classifier Γ, we formulate the training as a process to obtain an instance γ of this classifier as follows:

$$\Gamma(\mathcal{T}, \mathcal{F}, \mathcal{C}) \rightarrow \gamma$$

The advantage of building a generic classifier rather than many binary classifiers (for each type of relationship) is that the former enables the verification of different types of relationships. Consider a query for *"imitation"*, we could obtain the pair of entities *<"Infernal Affairs", "The Departed">* and *<"The Departed", Martin Scorsese">*. With a binary classifier for "imitation", we would only keep the first example. With a generic classifier, we would store both examples (classified in different classes). When an instance of a classifier which best minimizes the misclassification rate is trained, we can use this instance γ for assigning classes to the unclassified examples:

$$\gamma(\mathcal{X}, \mathcal{F}, \mathcal{C}) \rightarrow <(x_1, c_1), (x_2, c_1), (x_3, c_4), \ldots, (x_k, c_n)>$$

In our context, we cannot assume that the user provides many initial training data. A set of 5 to 10 examples for each class is realistic. However, some classifiers are robust with a few training examples while other classifiers achieve better results with more training data. Two problems arise from these remarks: the former is about selecting which examples should be added as training data while the latter deals with the choice of the classifier for each iteration. Let us discuss **the choice of the training data** first. To improve the robustness of the classifier, one has to train it with more data. To add new examples as training data, we have to select them among the sets of discovered examples from the previous iterations. We propose two strategies to achieve this goal. The first one (linking based) consists in selecting all examples that have been verified (with the classification step and the linking process) during any previous iterations. The second strategy (frequency based) is based on a frequency constraint: all examples which have been discovered in half of the previous iterations are added as training data during the current iteration. We believe that this selection of training data could be investigated further, e.g., when combining the two described strategies.

As for **the selection of the classifier**, the idea is the following: with the selected training examples, we generate instances of different types of classifiers (decision trees such as *J48* or *NBTree*, instance-based such as *KStar* or *IBk*, rule-based such as *NNge* or *JRip*, etc.). We perform cross-validation against the set of training examples for each instance of a classifier, and we compute the misclassification rate for each of them. The instance of classifier which achieves the minimal misclassification rate is selected to classify the examples discovered at this iteration. Such a strategy enables us to ensure that the best classifier is used for each iteration, but it also brings more flexibility to our approach.

We will show the impact of the training data and the type of classifiers in Section 9.6. The result of the classifier is a set of pairs, each of them composed of an example and its verified relationship class. The next step is to check whether the two extracted entities have a corresponding LOD entity.

9.5 Entity Linking

Entity Linking is the task of discovering local entity's correspondence in another data source [184]. The interest in linking entities is increasing rapidly due to the LOD movement. Note that linking does not imply coreference resolution is performed, but linking partially solves the coreference resolution problem. For example, the local entities “Martin Scorcese” and “Scorcese” are both linked to the same DBpedia entity *Martin_Scorsese*. The kind of linking we are performing here differs from structure-based linking as we only have labels at our disposal. The core of the idea is to match the entity against existing general purpose semantic knowledge bases such as DBpedia or Freebase to obtain corresponding LOD entities. Namely, we build various queries by decomposing the initial label and we query in the *descriptive text* attributes of knowledge bases (i.e., *common.topic.article* for Freebase, *dbpedia-owl:abstract* for DBpedia, etc.). In most cases, several candidate entities are returned and the task deals with automatically selecting the correct one. To fulfill this goal, the intuition is based on the hypothesis that the document about entity e and the descriptive text of LOD entity l should be fairly similar. Linking is performed for each entity of each document. That means that each document where e is mentioned serves as a context for disambiguation and matching against LOD knowledge bases. We note $\vec{\xi}$ the vector of terms in e’s document, while $\vec{\Lambda}$ represents the vector of terms of l. Terms in both documents are treated using *bag-of-words* method and both the context of e and the descriptive text of l are represented as a point in an n-dimensional term space. The cosine similarity score between the vectors $\vec{\xi}$ and $\vec{\Lambda}$ is calculated as follows:

$$sim(\vec{\xi},\vec{\Lambda}) = \frac{\sum_{i=1}^{n} \vec{\xi}_i \times \vec{\Lambda}_i}{\sqrt{\sum_{i=1}^{n}(\vec{\xi}_i)^2} \times \sqrt{\sum_{i=1}^{n}(\vec{\Lambda}_i)^2}}$$

where n is the size of the vocabulary. The terms in both vectors are based on classical *tf/idf* scores while the vocabulary is created out of the whole document collection. The top ranked entities are chosen as candidates for further comparison. This last comparison is performed on labels (and optionally “redirects” property) of the two entities to ensure a reasonable similarity in the label of e and one of the labels of l (e.g. “rdfs:label” and “dbpedia-owl:wikiPageRedirects” for DBpedia). This comparison is necessary because even though the similarity function returns a sufficiently high cosine similarity score, the labels should also be lexically similar. At this stage, the three well-known similarity measures are applied (Jaro Winkler, Monge Elkan and Scaled Levenshtein) as described in [184]. The top linked LOD entity is stored and is considered as a candidate until the end of the iteration. At the end of an iteration, all verified relationships (both by the classification and the linking) are converted into triples: for each entity, some triples express the link to LOD, the different labels and other possible attributes. One triple represents the relationship between the two entities and the type of relationship. Thus, the knowledge base is populated iteratively and can run continuously.

9.6 Experiments

To assess the effectiveness of our approach, we have conducted a number of experiments which are presented below.

9.6.1 Protocol

Our document collection is the English subset of the ClueWeb09 dataset[4] which consists of 500 million documents. This dataset is used by several tracks of the TREC conference [15]. For semantic tagging, several text processing tools have been used, including OpenNLP[5] (for tokenization and sentence splitting), the StanfordNLP[6](for POS tagging). For classification, six classifiers of different types were applied, namely the classic Naïve Bayes, the rule-based (NNge, DecisionTable), tree-based (J48, RandomForest) and lazy (KStar). These classifiers are included in the Weka software [85]. As for linking, we have used the DBpedia[7] dataset version 3.7 which contains 3,550,567 triples. Apache Lucene was employed for the backend indexing. Running KIEV for one type of relation on a subset of the collection took roughly 20 minutes.

9.6.2 Quality of Discovery Process

In this experiment, we focus on the movie dataset (remakes). Examples of relationships of interest include *imitation*, *adaptation* and *creator*. The ground truth for this dataset was obtained from the IMDb[8] movie database. This ground truth contained 545 entries out of the total 1052 remake pairs. For the remaining 507 we could not

[4]http://lemurproject.org/clueweb09/ (Last checked October 2012)
[5]http://opennlp.apache.org/ (Last checked October 2012)
[6]http://nlp.stanford.edu/ (Last checked October 2012)
[7]http://wiki.dbpedia.org/Downloads37 (Last checked October 2012)
[8]http://imdb.com (Last checked October 2012)

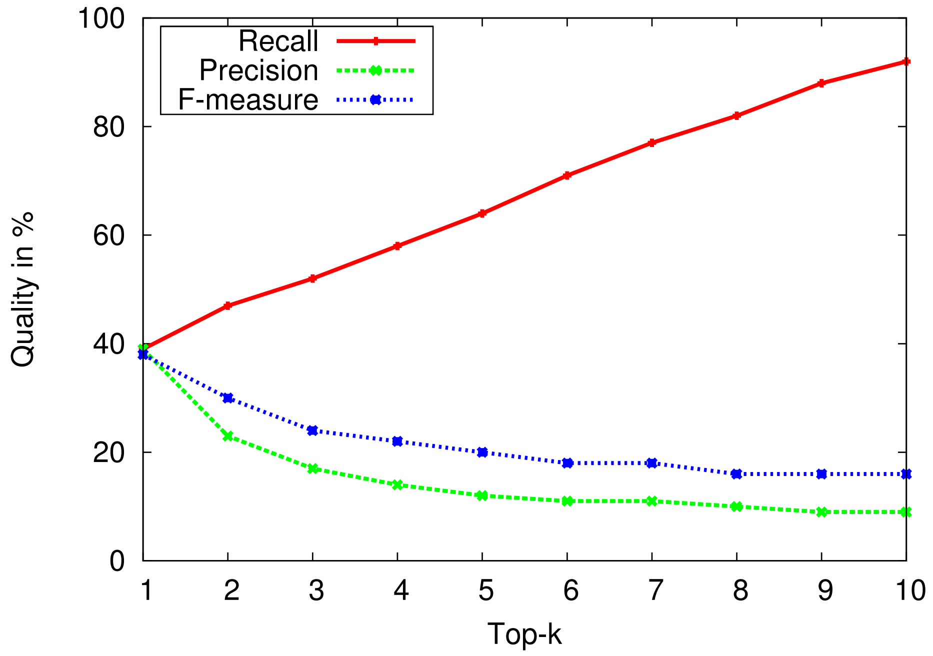


Figure 9.3: Before Verification.

find suitable documents in our collection. The reason for this is twofold. First, a number of movies were in non-English language. Second, a significant number of movies were created before the *Information Age*, i.e., those produced earlier than 1970s. Additionally, some examples were only mentioned in a few documents.

Figures 9.3, 9.4, 9.5 and 9.6 demonstrate the results of our experiments with or without the evidence-based verifications. The quality is presented in terms of well-known information retrieval measures - recall, precision and F-measure. Extracted examples are ranked and thus presented by top-k. In our context, the recall (at top-k) is the fraction of extracted correct examples (at top-k) out of the total number of correct examples (at top-k), while the precision (at top-k) is the number of extracted correct examples (at top-k) out of the total number of extracted examples (at top-k). F-measure is the harmonic mean of precision and recall.

We first notice that before the verification process, the precision score is quite low (≈39%) at top-1. This is because the discovery process extracts quite a lot of incorrect examples (false positives). As we increase the top-k, the recall also increases and eventually peaks at 87% at top-10. This trend illustrates that our approach achieves fairly high recall value but at the expense of precision. We tackle this issue with our

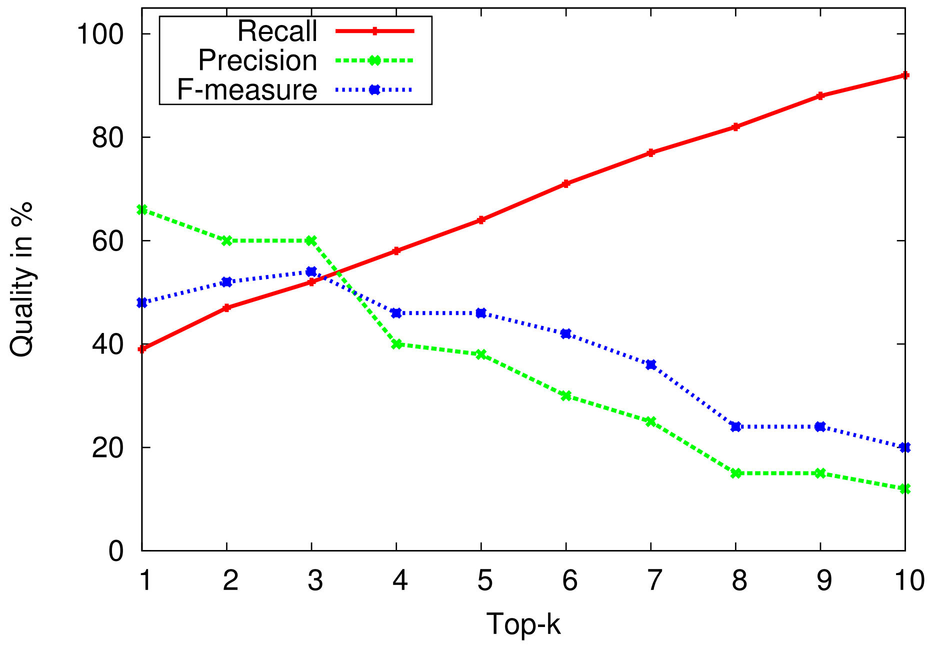


Figure 9.4: Linking only.

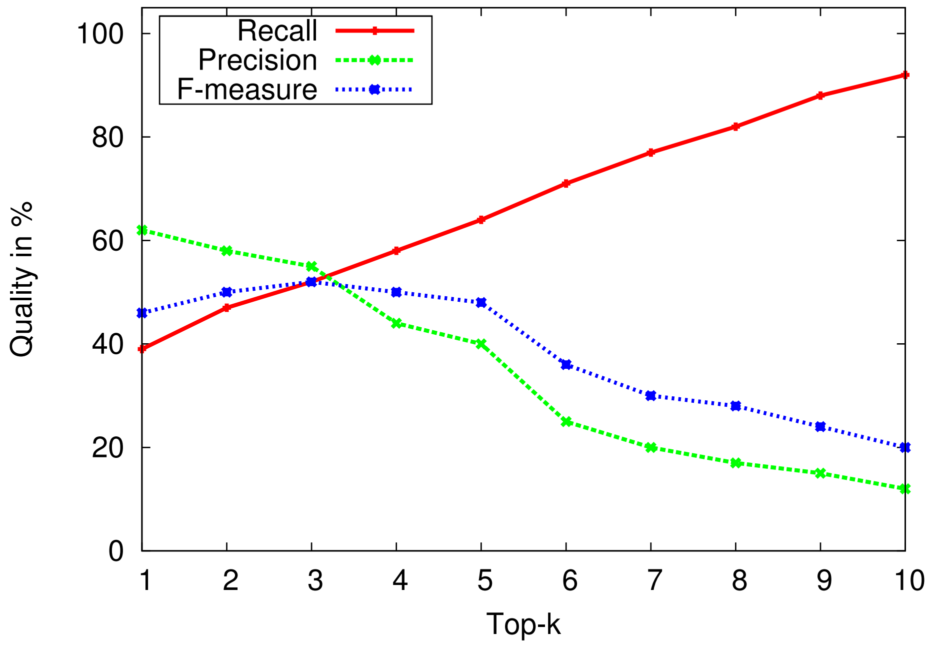


Figure 9.5: Classification only.

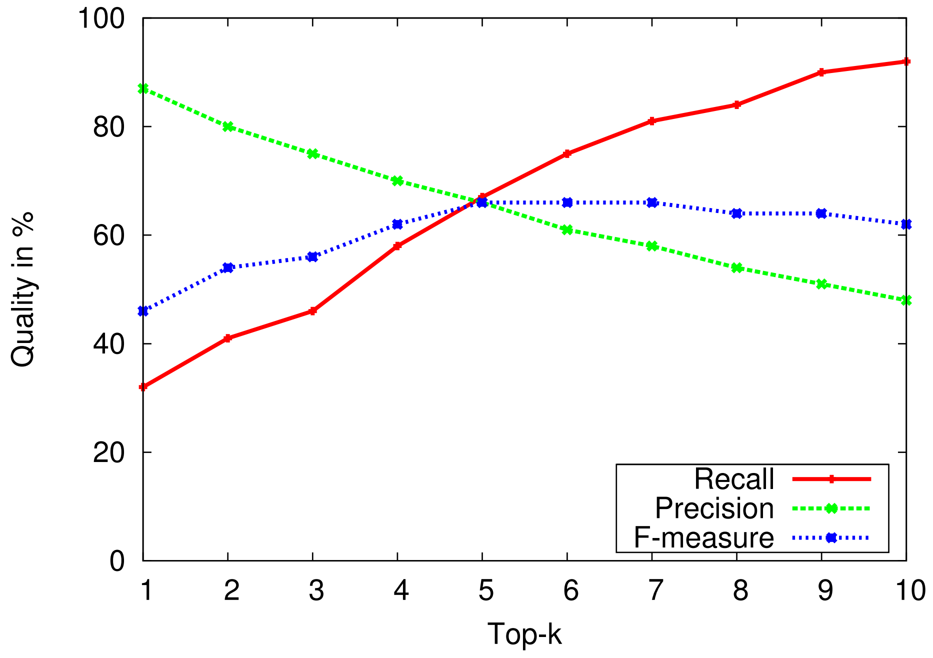


Figure 9.6: With Classification and Linking.

verification techniques, i.e., classification and linking. To show the benefit of both verification steps, the individual results of these steps are depicted in Figure 9.4 and 9.5. The recall value for both classification only and linking only is very similar to the values before verification, thus confirming that the individual verification do not discard many correct relationships. And the precision values for both steps, which are lower than the precision score after verification at any iteration, indicate that classification and linking do not discard the same incorrect examples. Thus, they enable a higher precision when they are combined.

Figure 9.6 illustrates that the verification process is effective to discard incorrect examples (precision score reaching ≈85% at top-1). However, a few correct relations were also discarded (a ≈6% decrease of recall at top-1), mainly due to the missing of a link to LOD of one of the entities. Furthermore, this phenomenon involves changes in the ranking of the extracted examples. Correct relationships can be promoted to a higher top, thus increasing the recall value of the highest top (e.g., at top-5). Finally, the benefit of the verification process clearly appears at top-10, since the plots have a close recall value (≈87%) but the verification discarded half of the incorrect examples (50% precision).

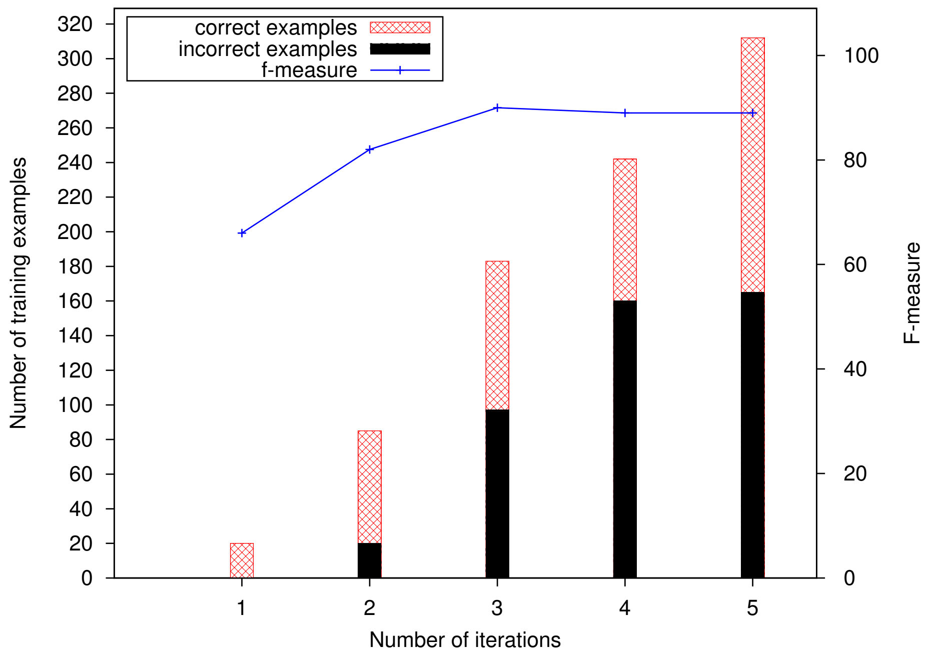


Figure 9.7: Impact of Training Examples with Frequency based Strategy.

9.6.3 Impact of the Training Data

The example discovery process *feeds* the classifier with new training data for the subsequent iteration. In this experiment, we have studied the impact of the selection of this training data by comparing the two strategies described in Section 9.4.

Frequency based strategy.

The frequency based strategy accounts for the frequency of a given example being discovered in all iterations. Initially, the user provides a set of 20 training examples (5 per relation type). If a given example is discovered repeatedly on each iteration, the intuition behind this strategy is that this example is most likely valuable and is promoted as a training example in the next iteration. Figure 9.7 illustrates the impact of the training data at the i-th iteration. On the y axis, we have the number of training examples that is used by our classifiers. On the right $y2$ axis, we have the harmonic mean F-measure obtained by the best performing classifier at the i-th iteration. The best performing classifier is the one with the highest F-measure during the classification with 10-fold cross-validation against the training data. Note that from one iteration to the other, the best performing classifier may be different because the set of training data evolves. For example, *KStar* was selected as the best classifier for the first iteration, but *J48* performed better in the second iteration.

On the plot, 85 examples discovered during the first iteration are selected for training for the second iteration. However, 20 of them are incorrect, i.e. false positives (shown as a black bar). Both the number of correct and incorrect examples increases as we move towards the 5-th iteration, eventually reaching 312 and 165 examples respectively for correct and incorrect examples. The high number of examples can be explained as follows. The frequency based strategy promotes as training data examples which appear at least 50% of the time in the previous iterations. Thus the number of added examples can potentially grow high. Yet, the F-measure obtained on the remakes dataset does not suffer much from the presence of incorrect examples (stable around 89% after the 3-rd iteration).

Linking based strategy.

The linking based strategy provides a harder constraint than the frequency based strategy when selecting the training data. Indeed, the candidate examples have to be verified both by the classification and by the linking process. Let us study the impact of this strategy over the quality of results by analyzing Figure 9.8. It presents the F-measure value achieved by the best generated classifier and the evolution of the number of training examples for five iterations.

The first remark about this plot deals with the F-measure scores, which are higher than those of the frequency based strategy from iterations 1 to 5. Another interesting phenomenon with this strategy is that the number of examples selected as training data (y axis) is lower than the one of the frequency based strategy. Indeed, the linking based strategy requires that both entities of an example are linked to LOD. Thus, this number is dramatically reduced, i.e., 200 in linking based strategy versus 312 in the frequency-based strategy at the fifth iteration. Finally, the number of incorrect examples is much lower in the linking based strategy too.

These remarks about the total number of correct examples together with the higher F-measure value are clear indicators that the linking based strategy is quality oriented while the frequency based strategy is performance oriented (simple and fast computation). The latter strategy is more appropriate for quickly generating training examples.

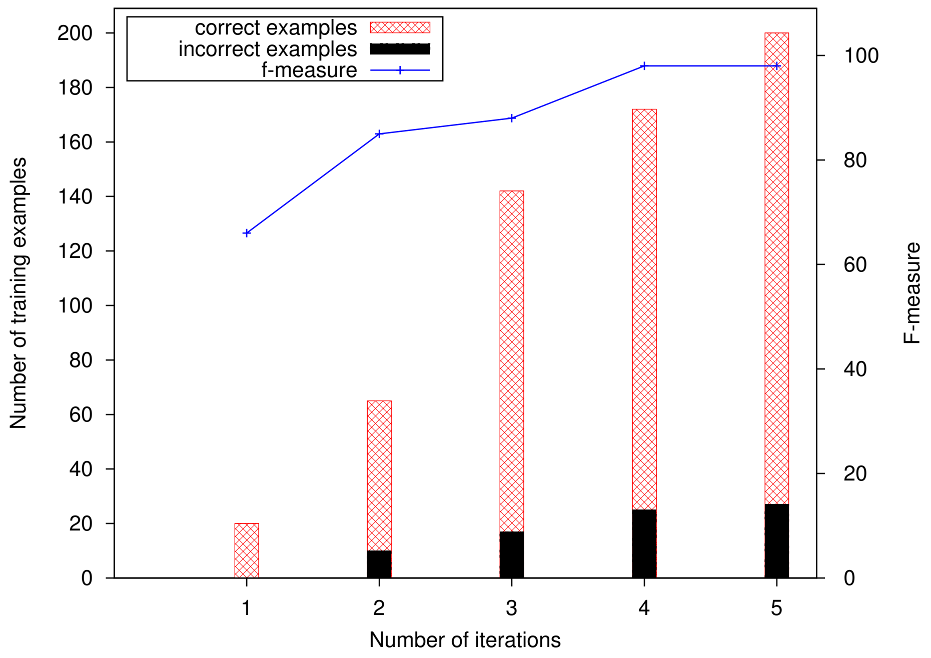


Figure 9.8: Impact of Training Examples with Linking based Strategy.

9.7 Summary

We have presented our novel approach KIEV for populating a knowledge base with entities and relationships. Our approach enables the analysis of a large amount of documents to extract examples (of entities) with their expected type of relationships after each iteration. A verification step ensures an acceptable quality for these extracted relationships by discarding irrelevant examples (classification) and by discovering the corresponding LOD entities (entity linking). Experiments performed on different datasets confirm the significant benefit of the verification step, thus enabling our approach to run continuously and to use new examples as training data to strengthen both the produced classifier and consequently the verification process.

When the knowledge base is publicly available, support for integrate user feedback to address the potentially contradictory cases between the two verification steps (classification and linking) would be useful to many applications. Another interesting feature would be an extension to discover any type of relationship, for example from an ontology, which could be achieved by automatically defining the features and the training examples.

Part V

Linking Entities

10 Linking Entities to LOD

10.1 Introduction

Linking is the final step in the generic framework proposed in this thesis that serves as an important step to both the metadata extraction and for the prototypes SPIDER and KIEV. Linking local entities to the LOD cloud is a solution with many benefits. First, it could enable the automatic enrichment of entities discovered in existing metadata with additional attributes and relationships. For instance, we may discover the relationship between a book and the screenplay that is based on the same novel by looking up the work in the LOD cloud. Secondly, the LOD cloud can be used to verify or guide the FRBR-based interpretation of existing information provided about the product. For instance, when using titles and authors to identify works there can be a large number of false positives if there are many translations or adaptations of the same work. The LOD cloud can be used to verify the proper works or to single out the work entities that are of main interest to end users.

There are different approaches to linking and this chapter presents an attribute-based linking approach. In particular, the focus is on linking FRBR work entities to their corresponding LOD entities, and it lies in the intersection of two domains. The former is **entity search**, since we want to discover equivalent entities based on their information. However, one of the entities we intend to match is a semantic entity in the LOD cloud, which deals with **entity ranking**.

The *entity search* problem, also known as *record linkage* or *entity resolution*, is a crucial task for data integration or data cleaning [76]. It mainly aims at identifying entities (objects or data instances) which represent the same real-world entity. Contrary to existing approaches, which are designed to match entities represented in a relational framework [113], we apply entity search to RDF entities. Besides, most of them are based on machine learning techniques, and require training data. Another major difference deals with the quality of the data sources: in our context, we can assume that the data from the LOD cloud does not contain many errors for a given entity.

On the Web, a similar task, called *entity ranking*, involves the discovery of an entity's main page, contrary to traditional search engines which propose documents mentioning a given entity. This task has been extensively studied and two initiatives have an entity ranking track arranged every year: *Initiative for the Evaluation of XML Retrieval* (INEX) [87] and *Text Retrieval Conference* (TREC) [194]. Most approaches which take part in these tracks are either based on information retrieval or semantic web [172, 192].

The main difference between our work and entity ranking is the availability of information. In our context, the type of the searched entity is not always specified, or with a broader topic. Conversely, the work that we want to match to a LOD entity can include useful information such as creator, year, or categories.

10.2 Overview

The linking approach presented in this chapter consists of two parts: blocking and matching. The motivation for this is as follows. The amount of data available on LOD is very large, and even the querying of only one data source can be time consuming. Thus, the goal is first to reduce the search space by obtaining a subset of LOD entities, a process called *Blocking*. Then, we can apply fine-grained *matching* techniques on these entities to compute their degree of similarity with the given entity. This process outputs a ranking for these LOD entities according to their similarity degree. Figure 10.1 sums up our approach for matching local entities to LOD. As a running example, we use a work entitled *The fellowship of the ring (LOTR)*. It includes the following (incomplete) list of attributes: *novel* as type, *JRR Tolkien* as creator, *science fiction & fantasy* for categories and no creation date.

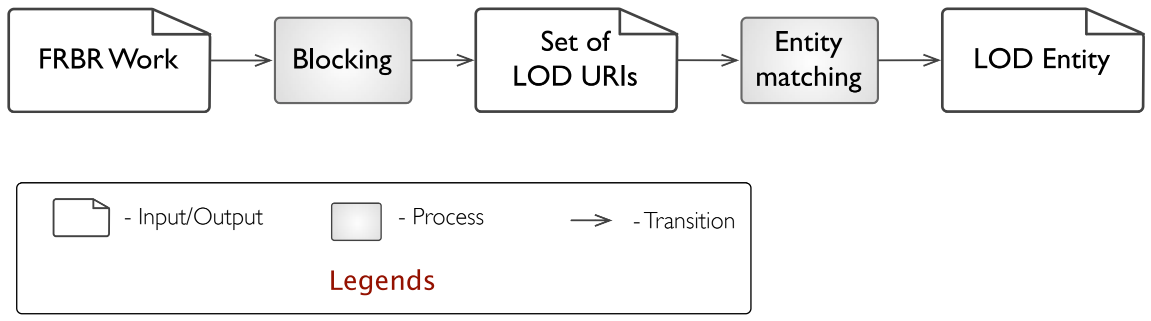


Figure 10.1: Workflow of Entity Matching.

10.3 Matching Local Entities to LOD

10.3.1 The Problem

We have a set of entities $\mathcal{W}$ and a set of LOD entities $\mathcal{L}$. Note that the LOD entities are linked to other entities by relationships, but we do not need this feature at this stage. All works and entities have a set of attributes. Considering a work $w \in \mathcal{W}$ and a LOD entity $l \in \mathcal{L}$, we note $\mathcal{F}$ the set of attributes shared by w and l. To assess a degree of similarity between w and l, we compute similarity values between their shared attributes. For an attribute $f \in \mathcal{F}$ shared by w and l, a similarity function is defined as follows:

$$sim_f(w,l) \rightarrow [0,1]$$

The similarity function returns values between 0 and 1, 0 indicating attribute f of w and l is completely dissimilar and 1 if the attribute f is identical for both w and l.

To compute similarity between entity pairs, we first need to obtain a list of possible candidates. This list of candidates is obtained through the *blocking* process.

10.3.2 Blocking

In entity matching, the large amount of entities implies to have a method for reducing the search space. For instance, if an entity has a title, a simple blocking method could be the matching of entities that share at least a common word in their titles.

In our context, this blocking process is required for two reasons. First, we query remote services with their potential issues (e.g., network overload, query limitations). Secondly, the set of entities is very large: more than 3.4 millions entities for DBpedia[1], 12 millions for Freebase[2] and thousands of entities for OpenCyc[3], which are only a few of the data sources from the LOD cloud. As a consequence, we need to have a heuristic to retrieve only a subset of entities against which we apply matching techniques. The following techniques can be used to search for a LOD entity [29]:

- Knowing or generating the correct URI of the entity, which cannot be applied in our context;
- Querying SPARQL endpoints;
- Querying a Lookup engine.

Both SPARQL and Lookup queries can return a set of LOD entities that match the search query. Thus, we reduce the search space by using these services, since they return an acceptable number of results (in our case usually between 0 and 200). In order to increase the probability of obtaining the correct entity in the search results, we need to build different queries based on the information contained in the work's attributes.

We have identified three interesting attributes of a work that can be used to generate a set of queries: title, creator and type. However, these attributes cannot be used directly in the query. They need to be transformed to remove extra information, to split creator's name, or to broaden a type. The idea is to create a set of query tokens for each of these attributes. More formally, we want to obtain three sets *titles*, *creators* and *types* containing query tokens such as:

$$\begin{aligned} \textit{titles} &\rightarrow \{title, normalized\_title\} \\ \textit{creators} &\rightarrow \{creator_1, ..., creator_k\} \\ \textit{types} &\rightarrow \{type, ext\_type_1, ..., ext\_type_m\} \end{aligned}$$

The *titles* set contains the full title of the work, and a normalized title in which extra information (e.g., inside parenthesis) and useless grammatical words are removed.

[1]http://dbpedia.org/ (Last checked January 2011)
[2]http://www.freebase.com/ (Last checked January 2011)
[3]http://www.opencyc.org/, (Last checked January 2011)

In other words, this normalized title only includes the most important words after a normalization process [74]. For the creators set, each creator's name is used as a query token. Finally, the *types* set contains the type of the work and its extensions. These extensions are hypernyms and synonyms from a predefined list (obtained from Wordnet[4]). For instance, the type *novel* is extended with *print* and *book*.

Once we have produced the three sets with their query tokens, we can combine the query tokens to generate a query. Combining these tokens is required either to obtain more results or to disambiguate. For instance, the novel entitled *airport* only returns a list of airports if the type is not included in the query. So the idea is to perform all combinations of 1, 2 or 3 tokens, each token belonging to a different set, and use these combinations as queries. All results returned by each query are merged based on the unique entity URI. Note that if all individual tokens do not return any results, there is no need to send queries which include this combination. At the end of this blocking process, we obtain a set of LOD entities (represented by their URI) against which we apply refined matching techniques.

We have generated different queries for our example work dealing with *The fellowship of the ring (LOTR)*. Table 10.1 shows some of these queries and provides the number of results returned by a Lookup service. Here, we highlight the need for sending multiple queries. Even with a well-known artistic work such as *The fellowship of the rings*, the lookup did not return any results with the full title, hence the need to simplify this title. Similarly, a query including the normalized title and the type of the work did not provide any results, contrary to the normalized title combined with an extended type.

10.3.3 Entity Matching

After the blocking step, we obtain a normalized set of LOD entities, and we need to match them against our work. To fulfill this goal, we first identify which shared attributes can be matched, and then we describe the similarity functions applied to these attributes. A global similarity value between a work and a LOD entity is finally computed, and filters may be used to discard some of the matched entities.

[4]http://wordnet.princeton.edu, last accessed January 2013

Type of Query	**Query**	**# Returned Entities**
title	The fellowship of the ring (LOTR)	0
norm_title	fellowship ring	5
title + creator	The fellowship of the ring (LOTR) JRR Tolkien	0
norm_title + creator	fellowship ring JRR Tolkien	0
norm_title + type	fellowship ring novel	0
norm_title + ext_type	fellowship ring book	1
norm_title + ext_type	fellowship ring print	0
creator + type	JRR Tolkien novel	0
creator + ext_type	JRR Tolkien book	1

Table 10.1: A Subset of Generated Queries for our Work Example.

Identifying Attributes.

First, we have identified the most important attributes that we can use to compare a local entity and a LOD entity. Although these attributes depend on the data sources we have on both sides (work and entity), five attributes are at least very common:

1. Title. In our running example, the work title has the value “The fellowship of the ring (LOTR)”;

2. Type of work/entity. For instance, the work type of *The fellowship of the ring (LOTR)* is “novel” while the type of the corresponding entity is “book”;

3. Creator. All artistic works have one or more creators. “J.R.R. Tolkien” is the creator of our example work;

4. Categories. They represent the genres or domains to which the artistic work belongs. *The lord of the Rings* categories may include “heroic fantasy”, “Middle Earth universe” or “science fiction & fantasy”;

5. Date of creation. *The fellowship of the ring (LOTR)* has been originally created in “1954”.

The first three attributes are in most cases present in both work and entity. On the contrary, the last two attributes may lack in one or both data sources. Although the year of creation may be misleading, it is useful in specific cases. Dealing with the work about the movie *the lord of the rings : the return of the king*, there exist a first movie produced by Bass and Rankin in 1980 and a second one by Peter Jackson in 2003. If the creator's names are lacking or subject to mistakes, the dates could help us to disambiguate the two candidate movies. Finally, the idea is to compute the similarity for these five shared attributes of a work and an entity.

Computing Individual Similarity Values.

We compute a similarity value between the same attributes of the work and the entity. However, the nature of these attributes are different: the *title* and *creator* are plain text while the *categories* are a set of words. The *type* is a word from a finite set of values while the year can have different formats. As a consequence, we need different similarity measures for matching these attributes. Schema matching and ontology alignment research fields have provided many techniques to discover similar elements in various data sources that we can apply in our context [74].

Attributes title and creator: To measure the similarity between character strings, we have selected three terminological similarity measures: Jaro Winkler, Monge Elkan and Scaled Levenshtein (see Section 2.5.2. Combining these similarity measures enables us to avoid the drawbacks related to one of the measure (e.g., the Levenshtein returns high similarity for small-sized strings which are very dissimilar [50]. Given the titles t (respectively creators c) of a work w and a LOD entity l, we compute the following similarity sim_{title} (resp. sim_{creat}) as the average between the three similarity measures:

$$sim_{title}(w,l) = \frac{jaro(t_w, t_l) + monge(t_w, t_l) + leven(t_w, t_l)}{3}$$

Attribute categories: As these categories are represented by a set of strings, we define a very basic similarity function sim_{cat} between two sets. It computes the number of identical categories between the set of categories of a work w and a LOD entity l.

$$sim_{cat}(w,l) = \frac{|cat_w \cap cat_l|}{\max(|cat_w|, |cat_l|)}$$

Attribute type: The type (extracted from its manifestations for the work) is limited to predefined values such as *book*, *movie*, *novel*. As the number of values is not large, we have built a small taxonomy extracted from the Wordnet hierarchy. To compute the similarity between two types, we can therefore apply the Resnik similarity [170]. It evaluates the similarity of these types based on the concepts that subsume them in our taxonomy. Figure 10.2 depicts a part of our taxonomy. For instance, the similarity value between the types *book* and *novel* in our taxonomy is equal to 0.29.

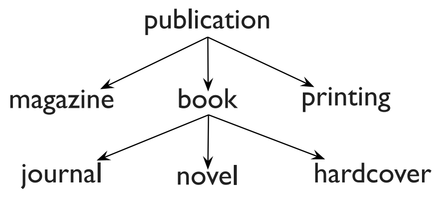


Figure 10.2: A Fragment of Our Taxonomy for Matching the Attributes *type*.

Attribute date: The idea is to extract only the year, which is a meaningful temporal granularity for artistic works. Thus, we compare the date value with several predefined patterns to extract the year, both for the work and for the entity. If the extracted years from the work and the entity are identical, the similarity function returns 1. Else, it returns a 0 value. Back to our running example: Table 10.2 shows a LOD entity with each of its attribute's value. The last column indicates the similarity value for the attribute with regards to the corresponding attribute of the work (which is detailed in Section 10.3.1). We notice that the title and the creator are terminologically similar (similarity values around 0.8). As the work does not contain a date, the similarity value for creation date equals 0.

Attribute	**LOD Property Value**	**FRBR Work Attribute Value**	**Similarity Value**
Title	The Fellowship of the Ring	The fellowship of the ring (LOTR)	0.77
Type	Book	Novel	0.29
Creator	J._R._R._Tolkien	JRR Tolkien	0.81
Categories	Fantasy	science fiction & fantasy	0.00
Date	1954-07-24	-	0.00

Table 10.2: Attributes and Similarity Values of the LOD Entity *The_Fellowship_of_the_Ring*.

Computing a Global Similarity Value.

From these attribute similarity values, we are able to derive a global similarity value. We have chosen a weighted average function to aggregate the values of all individual similarities. The global similarity value is computed with the following formula, where w is the work and l is the LOD entity, i.e., $w \in \mathcal{W}$ and $l \in \mathcal{L}$:

$$\text{sim}(w,l) = \frac{\alpha sim_{title}(w,l) + \beta sim_{type}(w,l) + \gamma sim_{creat}(w,l) + \delta sim_{cat}(w,l) + \zeta sim_{year}(w,l)}{\alpha + \beta + \gamma + \delta + \zeta}$$

In our running example, the DBpedia entity *The_Fellowship_of_the_Ring* and the work have a global similarity value equal to 0.37. As a comparison, the DBpedia entity related to the movie *The_Fellowship_of_the_Ring* obtains a similarity value of 0.22.

Filtering the Candidate Matches.

Similarly to many matching approaches, we can filter the candidate matches by selecting those with a similarity value above a given threshold. A correct tuning of this threshold is crucial since it directly impacts the quality. Note that a constraint filter could also be applied in our context: if the work deals with a *movie*, then all LOD entities with a *book* type should be discarded. We demonstrate in Section 10.4 the impact of a threshold filter. As a result, all remaining entities discovered for a work can be ranked given their similarity values. Similarly to most matching approaches, the user still needs to decide if one of the proposed entities corresponds to the work. However, we show in our experiment results that our approach often ranks the correct entity at the first position.

10.3.4 Discussion

First, the LOD cloud is incomplete, i.e., it does not contain all entities that correspond to the FRBR works. Yet, our blocking process may return several LOD entities, hence the need to compute their degree of similarity with the work. On the contrary, there may be no LOD entity returned by the blocking process. This does not mean that the LOD entity corresponding to the work does not exist. The benefits of our approach are threefold. First, it enables the verification of FRBRized data. But it can also be used to add new entities in the LOD cloud when a work has no corresponding entity. Specialized knowledge bases already use this mechanism to automatically create entities for a generic knowledge base, often with incomplete information. The last benefit is the semantic enrichment. Once a LOD entity is validated as correct for a given work, we can enrich this work by adding attributes extracted from the LOD entity. In addition, we can infer some simple relationships. For instance, we can link our work *The fellowship of the ring (LOTR)* with the other works of the trilogy thanks to the DBpedia property *dbpprop:books-of*.

10.4 Experiments

In our experiments we used a version of the FRBRpedia tool [69] to convert metadata retrieved from the Amazon bookstore (using the Amazon Product Advertising API[5]). A list of the 80 best selling fiction authors from Wikipedia[6] was used to query for product descriptions and a test sample of 684 distinct FRBR works was generated from the Amazon results.

The challenge is to discover a correct entity on the LOD cloud for each of these works. In this experiment, we have chosen DBpedia as our main source of corresponding entities. Note that our approach is not limited to this knowledge base and that we could have used another source such as Freebase or OpenCyc. However, DBpedia is regarded as the center of this LOD cloud as it has the largest number of connections to other data sources. First, we detail our experimental protocol. Then, we explain the quality results given a number of parameters. We finally open the discussion with faced issues.

10.4.1 Protocol

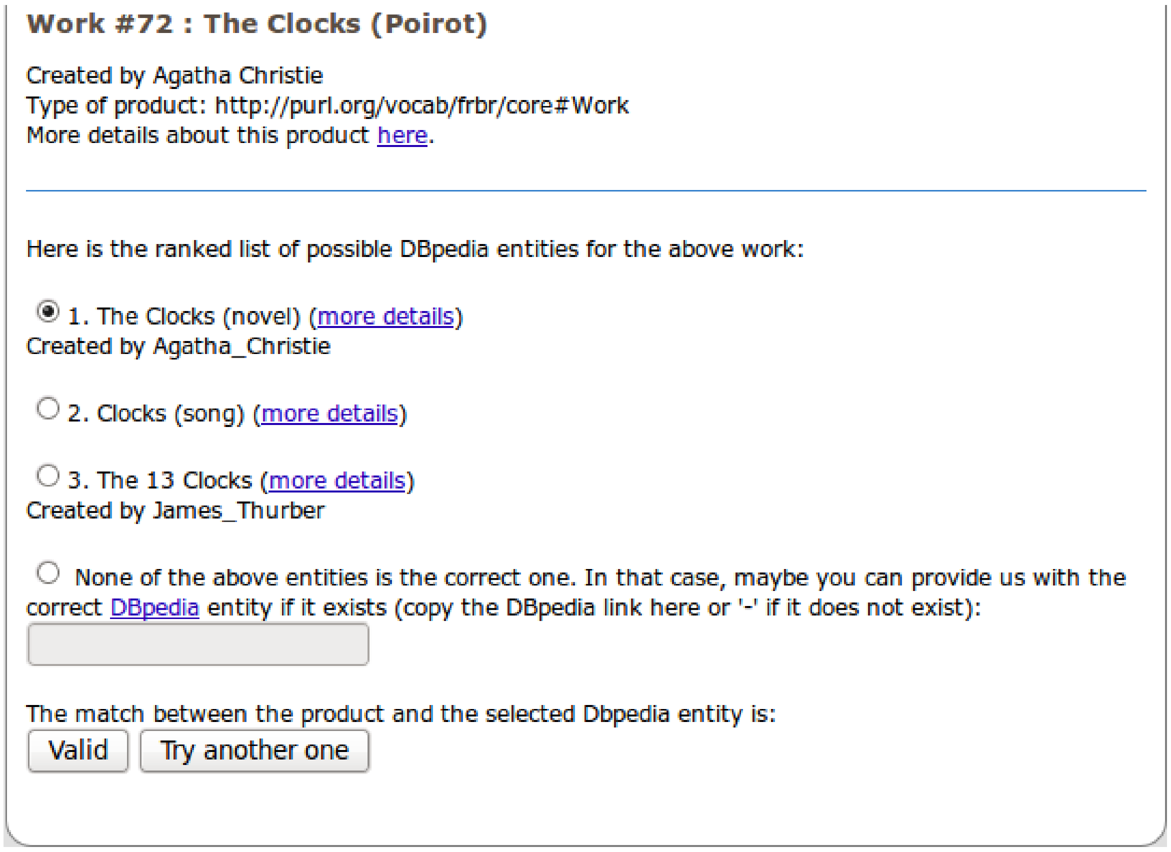
To reduce the search space, we could use SPARQL or Lookup queries. However, we have noticed that SPARQL queries are time-consuming with multiple constraints involving free-text strings. Thus, we use the *Lookup* API provided by DBpedia[7] to obtain a subset of DBpedia URIs representing entities that could correspond to the work using various queries as explained in Section 10.3.2. We used this reduced set of URIs as candidate matches for a given work. Matching techniques presented in Section 10.3.3 have been applied between the attributes of a work and those of the candidate matches. During this initial set of experiments, the global similarity value is computed with all weights equal to 1, which means that we do not promote any attribute. Similarly, we did not apply any filter to this global similarity value (i.e.,

[5]Amazon Product Advertising API, Ver 2010-10-01, `http://j.mp/amznProductAPI` (Last checked April 2011)

[6]`http://en.wikipedia.org/wiki/List_of_best-selling_fiction_authors`, (Last checked January 2011)

[7]DBpedia Lookup Interface, `http://lookup.dbpedia.org` (Last checked December 2010)

the threshold value is tuned to 0). The blocking and matching processes for the 684 works were performed in 10 to 12 minutes (without caching). Finally, we ranked the candidate matches for each work. For half of the 684 works, we were not able to discover any DBpedia entity. The remaining 343 works have at least one DBpedia entity. We presented the top-3 candidate matches for manual validation. Figure 10.3 depicts the user interface for validating the candidate matches. This validation step was performed by 8 different people from our research group, which means that they have to check all proposed LOD entities and decide whether it corresponds to the given work (based on available information, such as creators, titles, summaries, or types). If none of the proposed entities is correct, participants validated the work by manually searching DBpedia. This manual validation forms a ground truth for the collection, based on which we are able to compute quality results of our approach.



Work #72 : The Clocks (Poirot)

Created by Agatha Christie
Type of product: http://purl.org/vocab/frbr/core#Work
More details about this product here.

Here is the ranked list of possible DBpedia entities for the above work:

● 1. The Clocks (novel) (more details)
Created by Agatha_Christie

○ 2. Clocks (song) (more details)

○ 3. The 13 Clocks (more details)
Created by James_Thurber

○ None of the above entities is the correct one. In that case, maybe you can provide us with the correct DBpedia entity if it exists (copy the DBpedia link here or '-' if it does not exist):

The match between the product and the selected Dbpedia entity is:

Figure 10.3: User Interface for Validating a Match Between a Work and a DBpedia Entity.

10.4.2 Quality of Results

To assess the quality, we study the impact of the three parameters, namely the top-K matches, the threshold filter and the tuning of the weights in the global similarity value. Let us begin with the **top-K**. The number of correct discovered matches (true positives) at top-1, top-2 and top-3 are shown in Table 10.3. Most of the correct matches (189) are ranked at the top. At top-3, we only discover 12 more entities. Thus, our approach is able to present to the user the correct DBPedia entity at the top of the ranking.

	Top-1	**Top-2**	**Top-3**
Number of True-Positives	**189**	197	201

Table 10.3: Number of True Positives by Top-K

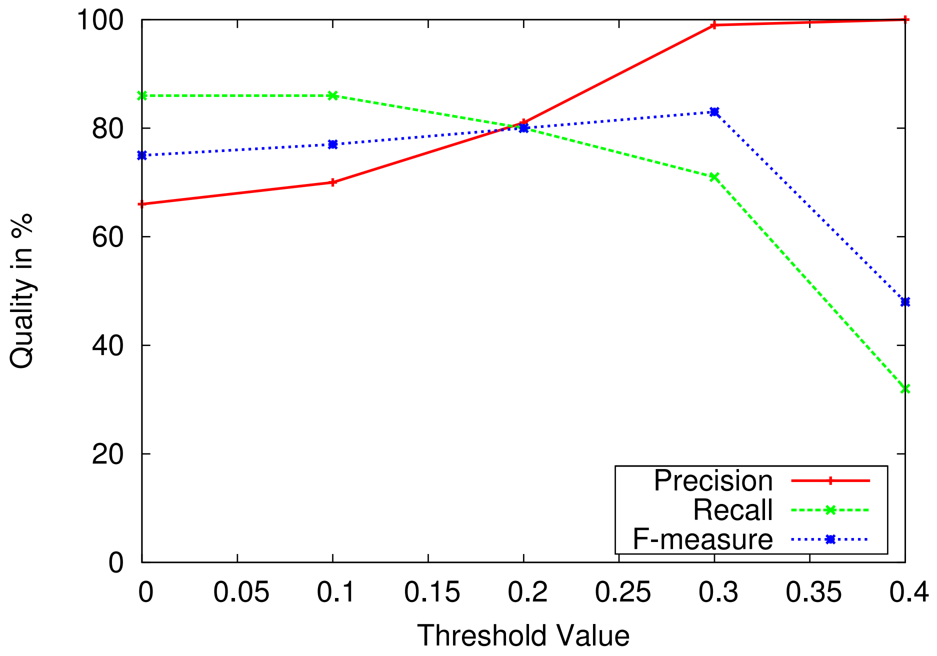


Figure 10.4: Quality Results (precision, recall, f-measure) w.r.t. a Threshold Filter

The following experiment deals with **the impact of the threshold filter** (see Section 10.3.3). We compute the quality in terms of precision, recall and f-measure, as discussed in [74]. Precision represents the percentage of correct matches among those discovered while recall stands for the percentage of correct matches discovered by our

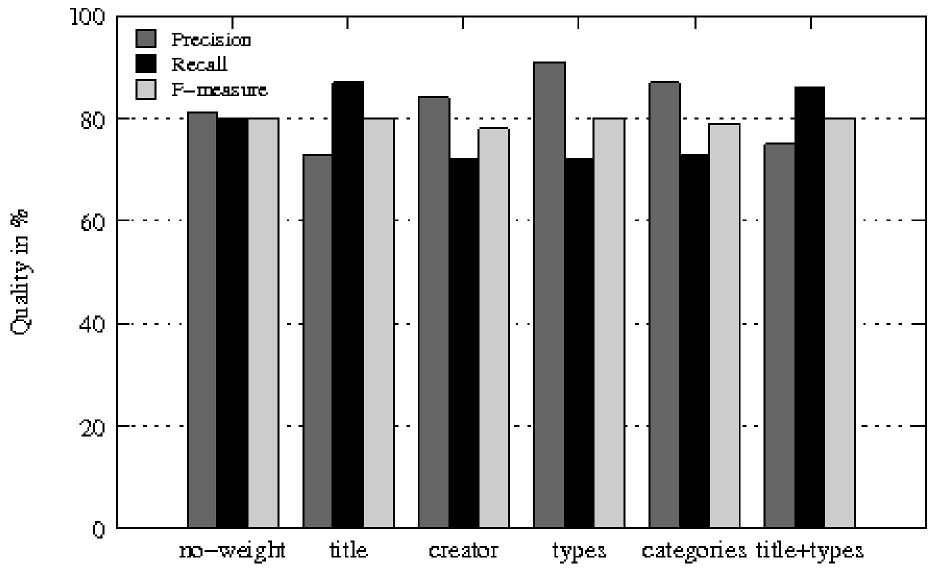


Figure 10.5: Quality Results w.r.t. the Weights of Individual Similarities

approach w.r.t. the total number of correct matches. F-measure is a tradeoff between precision and recall. Figure 10.4 depicts the quality obtained by our approach at top-1 when the threshold value for filtering matches varies. Without any threshold (value equal to 0), the f-measure reaches 76%. The recall value is around 85%, which means that we do not miss too many correct matches. However, we still discover many incorrect matches (precision at 66%). When we increase this threshold value, then the precision value increases while the recall score decreases. A balanced f-measure value (80%) is achieved for a 0.2 threshold. With higher threshold values, we are able to reach 100% precision, but at the expense of recall (71%).

In the last experiment we study **the impact of weights in the global similarity function** (see Section 10.3.3). We have previously shown that a threshold value equal to 0.2 provides balanced results between precision and recall, so we have used this value in this experiment. Figure 10.5 depicts the top-1 quality when we apply a weight to one or more individual similarity measures. For instance, the precision, recall and f-measure values respectively equal 73%, 87% and 80% with a weight on the *title* similarity measure. We notice two interesting points. The former deals with a weight on the title which enables the promotion of recall (87%). Indeed, when a work matches a LOD entity, their titles are often similar. But this high title similarity is limited by the other individual similarity functions. Thus, tuning the weight of the title allows us to discover more correct matches, but at the expense of precision. The latter point is the weight applied to types which promotes precision (91%). Indeed, a hard constraint on the types avoids the discovery of matches involving a work and a LOD entity with different types (such as movie and book).

10.5 Discussion

Our first observation is concerned with the quality of the input data and of their conversion into FRBR. This process obviously has an impact when linking to DBpedia. For instance, the search results from Amazon need to be cleaned. Indeed, they can contain dirty data such as "The Lord of the Rings: The Return of the King (Widescreen Edition)" and unrelated products (given the query). We also faced several issues with the DBpedia knowledge base. A lack of information in the DBpedia entity leads to no match, a case which may occur for DBpedia entities which are automatically created from other knowledge bases but with incomplete attributes. Similarly, an entity page can redirect to a related entity page (e.g. author, concept, event). As for the experiment results, our global similarity measure is reliable since most correct matches are discovered at top-1 and we miss only a few entities. Furthermore, the approach is flexible with the weights and the threshold, which both enable users to promote either precision or recall.

10.6 Summary

In this chapter, we have presented a generic framework to link a local entities to their corresponding LOD entity, using a query builder as blocking process and refined similarity measures as matching process. As a result of experiments with DBpedia, we have successfully discovered the correct DBpedia entity for most products.

The presented framework can be integrated with other LOD data sources (e.g. *Freebase*, *LastFM*). Indeed, linking a work to a specialized database (e.g. *MusicBrainz* for musical work) may provide a higher probability for discovering the correct match than a general knowledge base. Nevertheless, the results of this experimental study indicates that the approach can be used as a basis both for verification purposes and for semantic enrichment. The verification and linking are useful for example in cases when we automatically extract entities and their relations and at the same time we would want to link these entities and relations to the LOD. This has been used as basis for verification in Chapters 8 and 9 for an automatic entity and relation extraction method.

11 An Application of Linking

11.1 Linking Product Information

In this chapter, we demonstrate a linking of Web product information to the LOD by using entity matching techniques. One of the goals with linking to LOD is to verify the FRBRized information. We have used the DBpedia knowledge base but the approach is sufficiently generic to be applied to other knowledge bases.

11.2 Overview

We briefly describe how we transform a Web product from Amazon into the FRBR model and how we link the FRBR work to DBpedia. Figure 11.1 depicts the architecture of our system, which is detailed below.

Based on the product information extracted from Amazon bookstore, we first need to create (at least) four FRBR entities: work, manifestation(s), expression(s), and actor(s). The first step consists of identifying the work. We use the external service *OCLC Classify API*[1] knowledge base to retrieve the work corresponding to the product.

Although OCLC includes a large collection of works, it occurs that the service does not return any result, for instance with products specific to a country. In the future, we plan to integrate other methods for identifying a work such as z39.50[2]. The second

[1] http://classify.oclc.org (Last checked December 2012)
[2] http://www.loc.gov/z3950/ (Last checked December 2012)

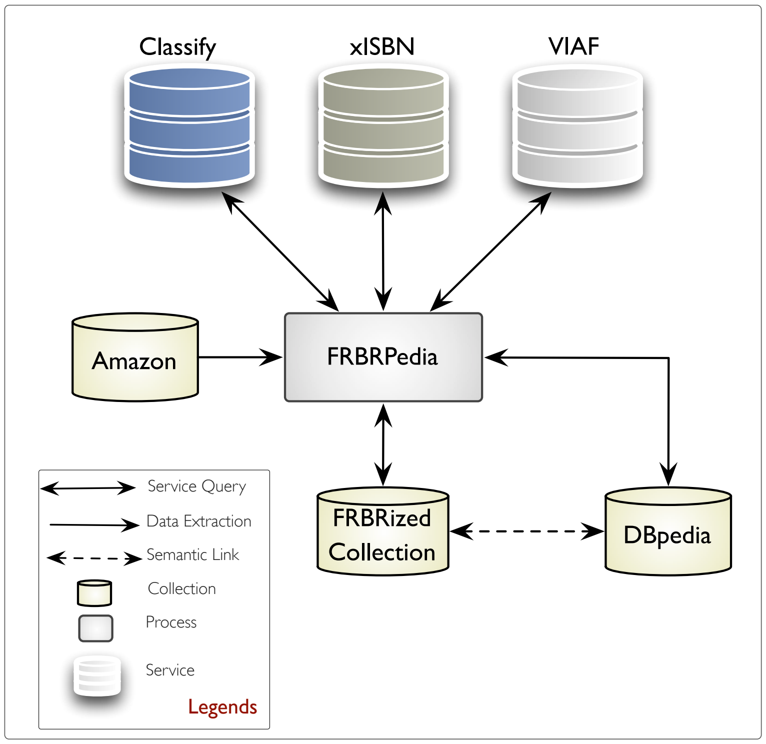


Figure 11.1: The Architecture of FRBRpedia.

step of our FRBRization process deals with the discovery of related manifestations for the previously identified work. Similarly, we chose an OCLC Service, *xISBN*, since a search in the same database increases our chances of finding related manifestations. The xISBN Web service returns ISBNs and other information associated with an individual intellectual work that is represented in the WorldCat catalog.

From the set of related manifestations, we can automatically generate the third category of FRBR entities: expressions. By analyzing attributes such as language and translator, we are able to create expressions, whose identifier is automatically generated. Next, the actors who contributed to this work are identified through the Virtual International Authority File (VIAF), a joint project between national libraries. This service aims at gathering authoritative names from many libraries into a global service freely available on the Web. Finally, the work, its expressions, manifestations and actors are linked during the last step. The final output of our FRBRization process is a set of RDF files for each entity type.

Linking the FRBR work to DBpedia is performed in two steps. First, we exploit information contained in the work (title, creator, categories, etc.) to query a DBpedia service named Lookup[3]. Indeed, the large amount of data stored in DBpedia cannot be fully searched without this kind of specific service, which returns a set of DBpedia entities related to the query. Our tool sends different queries to Lookup and it analyses all returned URIs with matching techniques. The goal of this second matching step is to rank the DBpedia entities and detect which one corresponds to the FRBR work. Therefore, we apply various similarity measures [74] between the attributes shared both by the work and the DBpedia entities (i.e., title, creator, date, categories and type). The computed similarity values are then aggregated into a global score using a weighted average function. This global score enables the ranking of all DBpedia entities for a given work, and the ones with the highest scores are presented to the user for validation. Once the user has validated the correct DBpedia entity, we finally display the FRBRized data enriched with DBpedia information.

11.3 Demonstration

Imagine we would like a FRBRized version of Agatha Christie's novel *"And Then There Were None"* sold on Amazon[4]. To achieve this goal with our tool, the user has two options. The former is a direct access at the webpage of FRBRpedia[5]. Figure 11.2 illustrates the main window of the FRBRpedia prototype when accessed through the Web interface. The latter is using a plugin which is activated when the user browses an Amazon product.

This browser plugin[6] is embedded using the GreaseMonkey extension for major browsers. The script is installed from the FRBRpedia webpage and the browser automatically detects if the user is in the product display page of amazon.com. In this case, a link appears on the product page. By clicking this button, the user is led to the FRBRpedia page. The main difference between the direct access and the plugin usage is that the identifier (ISBN or Amazon ASIN number) for the product is automatically filled in with the plugin.

With our Agatha Christie example, the ISBN *0312330871* appears in the input field

[3]http://lookup.dbpedia.org (Last checked December 2012)

[4]http://www.amazon.com/dp/0312330871/ (Last checked January 2012)

[5]http://november.idi.ntnu.no/frbrpedia/ (Last checked December 2012)

[6]http://november.idi.ntnu.no/frbrpedia/amzn2frbr.user.js (Last checked October 2012)

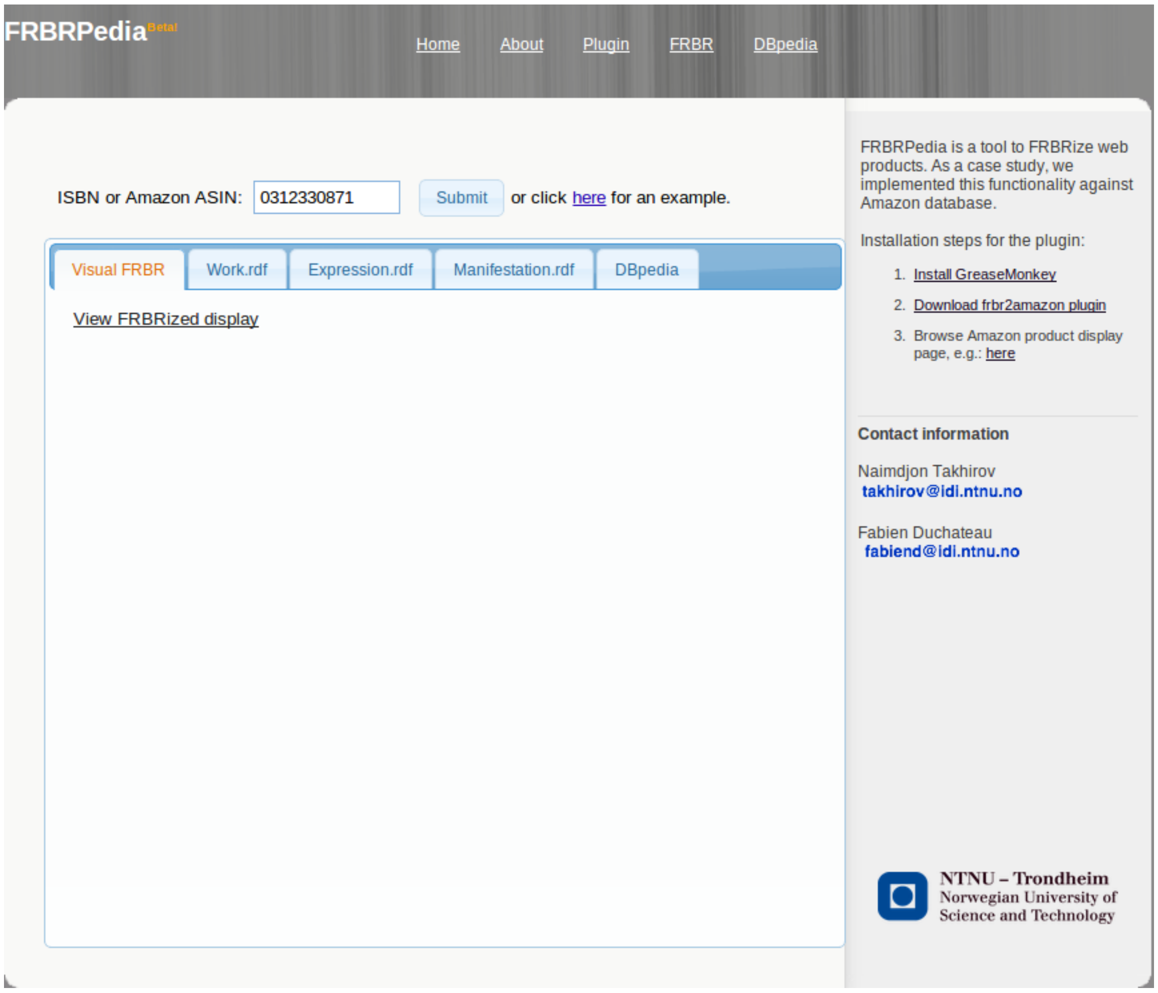
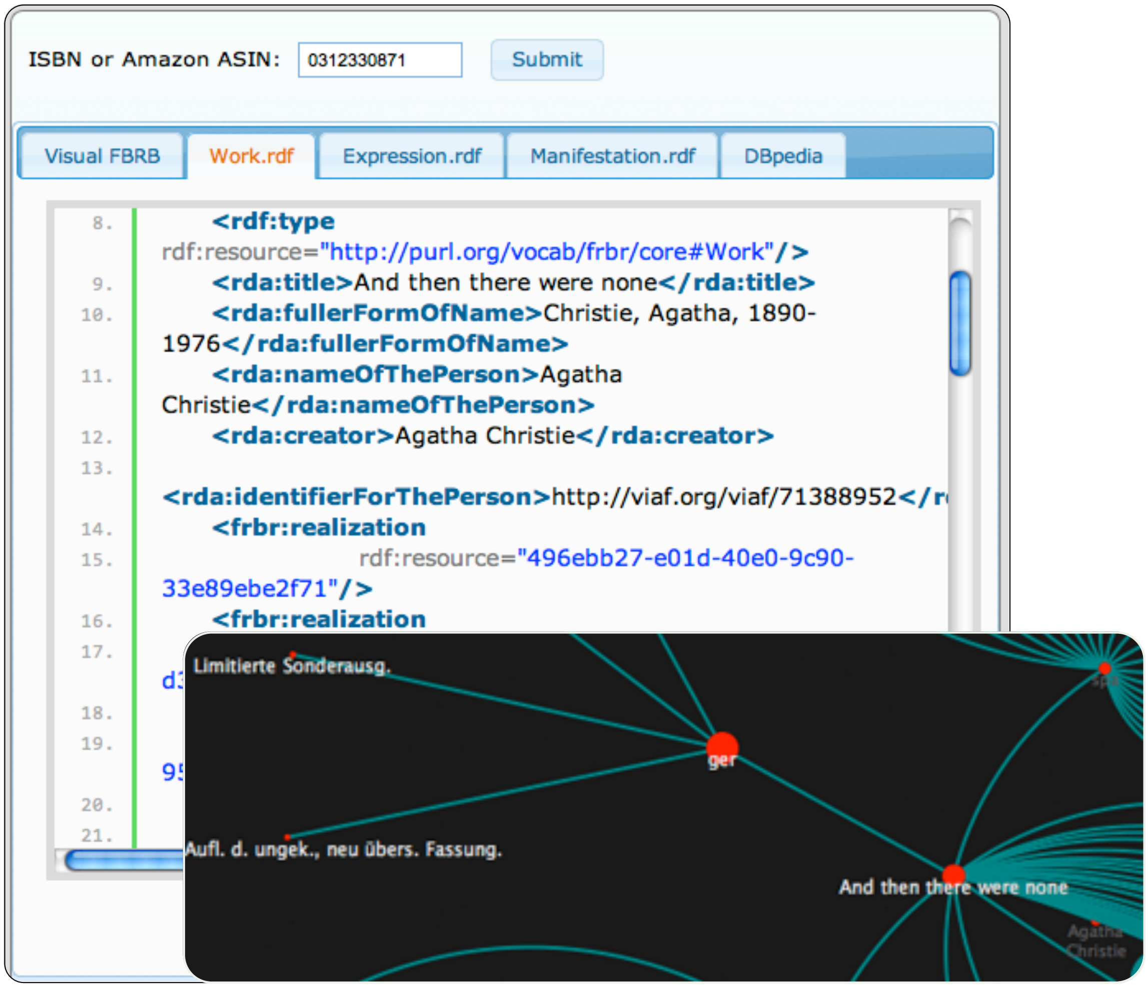

FRBRPedia Beta!
Home
About
Plugin
FRBR
DBpedia
ISBN or Amazon ASIN: 0312330871
Submit
or click here for an example.
Visual FRBR
Work.rdf
Expression.rdf
Manifestation.rdf
DBpedia
View FRBRized display
FRBRPedia is a tool to FRBRize web products. As a case study, we implemented this functionality against Amazon database.
Installation steps for the plugin:
1. Install GreaseMonkey
2. Download frbr2amazon plugin
3. Browse Amazon product display page, e.g.: here
Contact information
Naimdjon Takhirov
takhirov@idi.ntnu.no
Fabien Duchateau
fabiend@idi.ntnu.no
NTNU – Trondheim
Norwegian University of Science and Technology

Figure 11.2: Screenshot of FRBRPedia.

for the identifier. Once the form is submitted, the tool generates FRBR entities and it discovers the corresponding DBpedia entity (see Section 11.2). Depending on the number of manifestations for the work, the performance of the whole process varies from one second to one minute. In Figure 11.3, the FRBR entities, represented as RDF files, are shown in separate tabs with a syntax highlighting feature to promote readability. The corresponding DBpedia entity is also displayed in the fifth tab. Our tool has automatically selected as corresponding DBpedia entity, the one with highest similarity value. In our example , the selected DBpedia entity is *dbpedia.org/resource/And_Then_There_Were_None*. Additionally, the system creates a JSON file which is used for visualization of the FRBR entities. This visualization is performed using the HyperTree included in the JavaScript Visualization Toolkit [21].



```
    <rdf:type
rdf:resource="http://purl.org/vocab/frbr/core#Work"/>
    <rda:title>And then there were none</rda:title>
    <rda:fullerFormOfName>Christie, Agatha, 1890-
1976</rda:fullerFormOfName>
    <rda:nameOfThePerson>Agatha
Christie</rda:nameOfThePerson>
    <rda:creator>Agatha Christie</rda:creator>

<rda:identifierForThePerson>http://viaf.org/viaf/71388952</r
    <frbr:realization
            rdf:resource="496ebb27-e01d-40e0-9c90-
33e89ebe2f71"/>
    <frbr:realization

d3

95
```

Figure 11.3: Screenshot of FRBRPedia Displaying the Work RDF and a Fragment of Visualization.

The bottom part of Figure 11.3 depicts a fragment of visualization of Agatha Christie's work. Namely, a german translation is displayed along with different manifestations.

11.4 Summary

The prototype FRBRpedia demonstrates the use of linking method developed in Chapter 10 and its application in practice on the example of how to enhance the product metadata of Amazon's website. The product metadata was exploited and complementary information about specific intellectual content of the product (typically books, DVDs, etc) was populated using a publicly available Web APIs. Although the prototype developed merely for demonstration purposes, it clearly shows that such exploratory interfaces will enable users to learn about new content that otherwise might not be easily accessible. Further work on the FRBRpedia prototype focusing on enhancing the graphical interface of Web sites such as Amazon will likely be even more beneficial for end-users.

Part VI

Conclusion and Future Work

12 Conclusions and Future Work

12.1 Conclusions

The main goal of the research presented in this thesis is to study the use of automatic methods to populate knowledge bases. Specifically, this thesis explored automatic methods to populate knowledge bases considering the cultural heritage data as the initial input.

A major requirement in entity-centric knowledge bases is that semantics related to entities are made explicit by reducing ambiguities and missing information. Tackling these problems were the starting points and this work developed synergistic methods to populate knowledge bases. The general conclusion is that complementing knowledge extraction from metadata with the external sources results in less amount of missing and ambiguous information and in a more complete knowledge base. Combining cultural heritage data and unstructured documents for extraction is thus a more mutually advantageous conjunction of the two sources than using one source alone and the populated knowledge base bears a better quality in terms of completeness and accuracy.

The issue of populating knowledge bases raised several questions and the answers to those questions are presented below.

12.2 Answers to Research Questions

The research questions posed in the Section 1.3 are answered in the following:

RQ1. *Can we identify entities and relationships described implicitly in structured data?*

The main challenge in answering this question lies in the extraction of a correct set of entities that is described in the metadata including the type of entity, attributes and relationships. As we have shown in Chapter 6, a correct set of attributes have been chosen that indicate uniqueness of an entity, since often entities in metadata are identified by the descriptive text. Using this *descriptive identification* method yielded good results as demonstrated in the experiments with some room for improvements. The conclusive remark is that it is possible to identify entities by their descriptions and an additional improvement can be achieved by incorporating a user feedback mechanism.

RQ2. *How to extract complementary information about entities and relationships?*

This question has been explored through the development of the prototype SPIDER presented in Chapter 8 which builds on the results of entity-oriented extraction of knowledge from the metadata. Missing or ambiguous information extracted from metadata is a major bottleneck and we demonstrated the applicability of pattern-based relation extraction approach to address the problem. An additional hurdle is caused by the differences in the labels of entities in natural language documents. This issue has been addressed in SPIDER through label extension technique. This list of alternative labels is then used during the extraction process. The general remark is that combining cultural heritage data and unstructured documents for extraction is more advantageous and utilizes the synergy that can be achieved when combining the two sources.

RQ3. *What methods can be used to verify extracted knowledge?*

A specific solution – the prototype KIEV– was developed to address this question through a verification-aware extraction of semantic relationships from a large text corpora. KIEV leverages the pattern-based extraction capabilities of SPIDER. In many cases, a sentence may contain different entities and the discovery process in that case may generate incorrect examples. KIEV tackles this issue with two evidence-based methods to verify the extraction semantic information.

- *Classification* method is used to check the type of relationship with a machine learning approach. The classification aims at discarding these incorrect examples without prior knowledge about the two entities. Therefore, the verification process is seen as a *classification problem* and using various features a generic classifier used for verification, proved to be effective.

- *Linking* was found to be applicable for the verification task. The idea of discovering local entity's correspondence in another data source was leveraged using existing semantic knowledge bases on the LOD. In particular, the descriptive text of knowledge bases were queried to obtain the initial list of candidates and then a context-driven approach was employed for disambiguation.

RQ4. *How to perform linking of extracted entities to entities in other semantic knowledge bases?*

The main challenge in exploring this research question was given a local entity, how to effectively discover its corresponding entity or entities in other knowledge bases. In Chapter 10 it has been demonstrated that linking can be performed effectively when the properties of entities are exploited. The attribute-driven linking approach used the attributes of the local entity for the matching process. To address the issue of dealing with huge amount of data, the proposed approach uses a *blocking* mechanism to reduce the search space. The result of the linking was either creation of an *owl:sameAs* relationship for the local entity that will link to equivalent entities in other knowledge bases or the reuse of already established identifier. An application was demonstrated through the FRBRpedia prototype which used the *work* entity defined in the

FRBR model and discover the matches of those works in the semantic knowledge bases.

12.3 Summary of Contributions

The major contributions of this work can be summarized as follows:

- An approach for exploiting metadata to support fine-grained extraction of entities, their attributes and relationships. The framework includes three parts: representation, semantics and metrics. The representation provides a flexible format for explicitly encoding structures in an entity-oriented fashion. The semantics part supports the enhancement and correction of the original metadata. Furthermore, the framework includes semantic enrichment of existing metadata by using external knowledge bases and services. Finally, the metrics are used to evaluate the quality of the extraction.
- Proposed an approach to interpret metadata beyond the typical realm of the collection managed by the memory institutions. The use of sound conceptual models for interpreting metadata about various entity types such as book information facilitates reuse, enables discovery, maximizes interlinking of related entities and results in network effects that add even more value to the data. We consider FRBR as an underlying conceptual domain model and demonstrate that a network of entities and relationships can be extracted out of product descriptions found on the Web. This hypothesis was supported with an experimental evaluation.
- Extraction of missing or ambiguous semantic entity information from unstructured sources to address the issues raised by the results of metadata interpretation. The prototype SPIDER was developed to tackle this challenge at Web-scale and can also extract new entities and relationships thus supporting the novelty aspect. The proposed method is an extension of existing pattern-based techniques with major extensions. In particular, we incorporated several additional components: flexible definitions patterns with pattern similarity function to compute the semantic distance between patterns, provenance-aware confidence score which consists of normalized sum of relevance score, page spam-score and PageRank, identifying the alternative labels of entities, and finally the large-scale aspect to deal with Web-scale

knowledge base population.

- Verification of extraction with evidence based methods when populating knowledge bases. Two verification methods were developed in the prototype KIEV as an extension to SPIDER: classification and linking. With the classification method, we train a generic classifier with a range of features. For the linking, we perform automatic context-driven discovery of corresponding LOD entity.

- Present a two-step linking technique that establishes connection between equivalent entities. In this approach, given the local entity we discover corresponding entities across the data sources in the LOD cloud. To tackle the problem of large data volume, we first reduce the search space with a blocking mechanism. In the second step, the property-aware linking between the local entity and the LOD entity is achieved. The practical applicability of the proposed approach has been demonstrated with the FRBRpedia tool in the example of linking FRBR *work* entities to their corresponding entities in the DBpedia.

12.4 Future Work

The contributions presented in this thesis advance the state of the art in knowledge base population. However, the problem is far from being solved completely and there are many areas with the high potential for improvement and some future directions are presented below.

12.4.1 Interpreting Metadata

The metadata interpretation approach presented in this thesis can be further improved in several directions. The approach relies on a predefined set of rules for interpreting metadata. These rules are refined for each collection because of the differences in cataloging practices in the community. For each new collection, the existing set of rules may not be applicable or may require substantial changes to correctly extract entities, their attributes and relationships from the metadata.

Another point is the use of identifiers for the entities. The interpretation typically assigns locally generated identifiers to primary entities. This feature can be improved to discover and reuse existing identifiers in the LOD. The framework already has a mechanism to perform this task (Chapter 10) and this feature can seamlessly be integrated into the identity generation step.

The approach presented in this work is flexible when interpreting entities in the MARC based metadata using the FRBR model. Further investigation is needed to explore the methods applicability in the context of other metadata encoding schemes and using other sound conceptual models.

12.4.2 Extracting Entities and Relationships

The provenance feature of knowledge extraction included in this work, considers the characteristics of a document within the collection. The distributed nature of the Web and independent knowledge extraction efforts along with the LOD will in the future require provenance that is distributed. A grand challenge in this regard is how to handle conflicting facts from individual data sources.

The extraction methods in SPIDER and KIEV considered static collections which can result in incomplete or outdated content over time. Even with regular extraction efforts there will be a gap between extraction times and this temporal gap can affect the quality of content in knowledge bases. The dynamic nature of the Web brings new possibilities and at the same time poses new challenges. Social media sites (blogs, news, user contributed reviews and comments) are an important part of our digital society and have the potential to improve the facts stored in the knowledge bases both in terms of coverage and accuracy. Therefore, a practical tool for capturing the latest information from the social media and enriching it into the form of relational facts is interesting future work. A preliminary outline of the challenges and opportunities is discussed in [154].

This work explored extraction of binary relationships, i.e. relationships between two participating entities. There are cases in which it is highly desirable to include for example a third dimension to the extracted fact. For instance, a common feature of news website is the ability by users to rate a given article or comments to the article. Therefore, there a third dimension in this case is the rating given by the user in addition to the user and the article.

So far, prominent projects including this work focused in monolingual extraction of knowledge, namely collection of English language documents. There are many other languages in the world and using numerous language for extraction will boost recall. Moreover, some facts can only be extracted based on the local language of participating entities. One of the possible problems deals with patterns developed for one language may not be directly applicable another language due to the various linguistic features. Finally, there are issues with contradictory facts. For a given country, a fact may be true, and false for another country (e.g. geopolitics).

12.4.3 Linking Entities

The proposed linking feature exploits attributes of an entity to discover corresponding entities in the LOD. The next step after establishing a link, typically involves an integration of the two entities in some kind of data fusion applications (e.g. merging attributes of the two entities.

Name variants for entities are still a major problem in the entity linking task and the issue is far from being solved. Humans often easily understand and match a name variant to a canonical entity. However, it is a complex task to perform automatically and may significantly reduce recall for some applications.

Incorporating probability for entity matching in knowledge base population task is essential to be able to deal with the uncertainty. Uncertain data have recently gained popularity in the database domain [182] and it has been studied at large to repair or clean databases. A probabilistic model describes the data that is observable from a system. Particularly, methods of Bayesian modeling as well as Markov Logic may be used for efficient inferencing during knowledge base population. In this regard, issues such as *"How to model uncertainty?"*, *"Which modeling approach is most effective?"* need to be explored. Furthermore, uncertainty can be useful in such cases as operating in open domain extraction mode and lack of predefined training data.

A related area of research is accelerating knowledge bases by recommending edits. Specifically, the *knowledge base acceleration* (KBA) initiative – Cumulative Citation Recommendation (CCR) task – was initiated in 2012 at the TREC conference. The main goal of the KBA systems is to help humans expand general purpose encyclopedic knowledge bases like Wikipedia by automatically recommending edits based on incoming streams of documents. A core step in this process is that of identifying relevant content, i.e., filtering documents that would imply modifications to the attributes or relations of a given target entity. A multi-step classification approach was proposed [16] to tackle this issue. The CCR task is further developed in 2013 and additional components have been included: novelty or salience. Additionally, the 2013 edition of KBA, now includes a new task, called slot filling which basically refers to detecting changes to the attribute values of an entity[1].

[1]http://trec-kba.org/trec-kba-2013.shtml (Last checked April 2013)

13 Appendix FRBR-ML XML schema.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:marcxml="http://www.loc.gov/standards/marcxml/schema/"
        elementFormDefault="qualified">
    <xs:element name="collection">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element ref="manifestation" minOccurs="1" maxOccurs="
                    unbounded"/>
                <xs:element ref="expression" minOccurs="0" maxOccurs="unbounded
                    "/>
                <xs:element ref="work" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="person" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="corporateBody" minOccurs="0" maxOccurs="
                    unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="manifestation" type="manifestation"/>
    <xs:element name="expression" type="expression"/>
    <xs:element name="work" type="work"/>
    <xs:element name="person" type="person"/>
    <xs:element name="corporateBody" type="corporateBody"/>

    <xs:complexType name="manifestation">
        <xs:sequence>
            <xs:element ref="controlfield"/>
            <xs:element ref="datafield"/>
            <xs:element ref="identifier" minOccurs="0" maxOccurs="1"/>
            <xs:element name="title" type="title" minOccurs="1" maxOccurs="1"/>
            <xs:element name="publisher" type="publisher" minOccurs="0"
                maxOccurs="1"/>
            <xs:element name="description" type="description" minOccurs="0"
                maxOccurs="1"/>
            <xs:element name="series" type="series" minOccurs="0" maxOccurs="1"
                />
            <xs:element ref="expression" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="identifier" use="required"/>
        <xs:attribute name="type" use="required" type="xs:decimal"/>
        <xs:attribute name="embodimentOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="producer" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="alternate" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="alternateOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="relation" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="embodimentOf"/>
                    <xs:enumeration value="producer"/>
                    <xs:enumeration value="alternate"/>
                    <xs:enumeration value="alternateOf"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

    <xs:complexType name="expression">
        <xs:sequence>
            <xs:element name="language" type="language" minOccurs="0" maxOccurs
                ="1"/>
            <xs:element name="title" type="title" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="datafield"/>
            <xs:element ref="work"/>
            <xs:element name="translator">
                <xs:complexType>
                    <xs:sequence><xs:element ref="person"/></xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="realizationOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="abridgement" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="abridgementOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="adaption" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="adaptionOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="arrangement" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="arrangementOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="complement" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="complementOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="embodied" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="imitation" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="imitationOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="successor" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="successorOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="translation" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="translationOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="relation" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="realizationOf"/>
                    <xs:enumeration value="abridgement"/>
                    <xs:enumeration value="abridgementOf"/>
                    <xs:enumeration value="adaption"/>
                    <xs:enumeration value="adaptionOf"/>
                    <xs:enumeration value="arrangement"/>
                    <xs:enumeration value="arrangementOf"/>
                    <xs:enumeration value="complement"/>
                    <xs:enumeration value="embodied"/>
                    <xs:enumeration value="imitationOf"/>
                    <xs:enumeration value="revisionOf"/>
                    <xs:enumeration value="successor"/>
                    <xs:enumeration value="successorOf"/>
                    <xs:enumeration value="summarization"/>
                    <xs:enumeration value="summarizationOf"/>
                    <xs:enumeration value="supplement"/>
                    <xs:enumeration value="supplementOf"/>
                    <xs:enumeration value="transformation"/>
                    <xs:enumeration value="transformationOf"/>
                    <xs:enumeration value="translationOf"/>
                    <xs:enumeration value="translation"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

    <xs:complexType name="work">
        <xs:sequence>
```

```
            <xs:element name="title" type="title"/>
            <xs:element name="hasSubject" type="subject"/>
            <xs:element ref="datafield"/>
            <xs:element name="person" type="person" minOccurs="0" maxOccurs="1"
                />
        </xs:sequence>
        <xs:attribute name="adaption" type="xs:NMTOKEN" use="optional"/>
        <xs:attribute name="adaptionOf" type="xs:NMTOKEN" use="optional"/>
        <xs:attribute name="complement" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="complementOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="creator" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="imitation" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="imitationOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="successor" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="successorOf" use="optional" type="xs:NMTOKEN"/>
        <xs:attribute name="relation" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="adaption"/>
                    <xs:enumeration value="adaptionOf"/>
                    <xs:enumeration value="complement"/>
                    <xs:enumeration value="complementOf"/>
                    <xs:enumeration value="creator"/>
                    <xs:enumeration value="realization"/>
                    <xs:enumeration value="imitation"/>
                    <xs:enumeration value="imitationOf"/>
                    <xs:enumeration value="successorOf"/>
                    <xs:enumeration value="successor"/>
                    <xs:enumeration value="summarization"/>
                    <xs:enumeration value="summarizationOf"/>
                    <xs:enumeration value="supplement"/>
                    <xs:enumeration value="supplementOf"/>
                    <xs:enumeration value="transformation"/>
                    <xs:enumeration value="transformationOf"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

    <xs:complexType name="corporateBody">
        <xs:sequence>
            <xs:element name="name" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="datafield"/>
        </xs:sequence>
        <xs:attribute name="hasCreated" type="xs:IDREFS" use="optional"/>
        <xs:attribute name="hasPublished" type="xs:IDREFS" use="optional"/>
        <xs:attribute name="relation">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="hasCreated"/>
                    <xs:enumeration value="hasPublished"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

    <xs:complexType name="person">
        <xs:sequence>
            <xs:choice>
                <xs:element name="fullName" minOccurs="0" maxOccurs="1"/>
                <xs:element name="name" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                        <xs:attribute name="first" type="xs:string"/>
                        <xs:attribute name="last" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
            </xs:choice>
            <xs:element ref="datafield"/>
        </xs:sequence>
        <xs:attribute name="creatorOf" type="xs:IDREFS" use="optional"/>
        <xs:attribute name="relation">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="creatorOf"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="identifier">
        <xs:complexType>
            <xs:all>
                <xs:element ref="controlfield"/>
                <xs:element ref="datafield"/>
            </xs:all>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="title">
        <xs:sequence>
```

```
            <xs:element ref="datafield" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="publisher">
        <xs:sequence>
            <xs:element ref="datafield" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="description">
        <xs:sequence>
            <xs:element ref="datafield" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="series">
        <xs:sequence>
            <xs:element ref="datafield" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="language">
        <xs:sequence>
            <xs:element ref="datafield" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="subject">
        <xs:sequence>
            <xs:element ref="datafield" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="datafield" type="dataFieldType"/>

    <xs:complexType name="dataFieldType">
        <xs:sequence maxOccurs="unbounded">
            <xs:element name="subfield" type="subfielddatafieldType"/>
        </xs:sequence>
        <xs:attribute name="id" use="optional" type="idDataType"/>
        <xs:attribute name="tag" type="tagDataType" use="required"/>
        <xs:attribute name="ind1" type="indicatorDataType" use="required"/>
        <xs:attribute name="ind2" type="indicatorDataType" use="required"/>
    </xs:complexType>

    <xs:element name="controlfield" type="controlFieldType"/>

    <xs:complexType name="controlFieldType">
```

```
        <xs:simpleContent>
            <xs:extension base="controlDataType">
                <xs:attribute name="id" type="idDataType" use="optional"/>
                <xs:attribute name="tag" type="controltagDataType" use="required"
                    />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:simpleType name="controlDataType">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="preserve"/>
        </xs:restriction>
    </xs:simpleType>


    <xs:simpleType name="controltagDataType">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="preserve"/>
            <xs:pattern value="00[1-9A-Za-z]{1}"/>
        </xs:restriction>
    </xs:simpleType>


    <xs:simpleType name="tagDataType">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="preserve"/>
            <xs:pattern
                value="(0([1-9A-Z][0-9A-Z])|0([1-9a-z][0-9a-z]))|(([1-9A-Z][0-9
                    A-Z]{2})|([1-9a-z][0-9a-z]{2}))"
            />
        </xs:restriction>
    </xs:simpleType>


    <xs:simpleType name="indicatorDataType">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="preserve"/>
            <xs:pattern value="[\da-z ]{1}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="subfielddatafieldType">
        <xs:simpleContent>
```

```
                <xs:extension base="subfieldDataType">
                    <xs:attribute name="id" type="idDataType" use="optional"/>
                    <xs:attribute name="code" type="subfieldcodeDataType" use="
                        required"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
        <xs:simpleType name="subfieldDataType">
            <xs:restriction base="xs:string">
                <xs:whiteSpace value="preserve"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="subfieldcodeDataType">
            <xs:restriction base="xs:string">
                <xs:whiteSpace value="preserve"/>
                <xs:pattern value="[\dA-Za-z!&quot;#$%&amp;’()*+,-./:;&lt;=&gt;?{}_
                    ^‘~\[\]\\]{1}"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="idDataType">
            <xs:restriction base="xs:ID"/>
        </xs:simpleType>
    </xs:schema>
```

References

[1] Trond Aalberg. *Supporting Relationships in Digital Libraries*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, April 2003.

[2] Trond Aalberg. A Process and Tool for the Conversion of MARC Records to a Normalized FRBR Implementation. In Shigeo Sugimoto, Jane Hunter, Andreas Rauber, and Atsuyuki Morishima, editors, *Proceedings of the 9th International Conference on Asian Digital Libraries (ICADL'06)*, volume 4312 of *Lecture Notes in Computer Science*, pages 283–292, Heidelberg, Germany, 2006. Springer.

[3] Trond Aalberg, Frank Berg Haugen, and Ole Husby. A Tool for Converting from MARC to FRBR. In Julio Gonzalo, Costantino Thanos, M. Felisa Verdejo, and Rafael C. Carrasco, editors, *Proceedings of the 10th European Conference Research and Advanced Technology for Digital Libraries, (ECDL '06)*, volume 4172 of *Lecture Notes in Computer Science*, pages 453–456, Heidelberg, Germany, 2006. Springer.

[4] Trond Aalberg, Tanja Mercun, and Maja Žumer. Coding FRBR-structured bibliographic information in MARC. In Hsin-Hsi Chen and Gobinda Chowdhury, editors, *Proceedings of the 13th international conference on Asia-pacific digital libraries (ICADL '11)*, volume 7008 of *Lecture Notes in Computer Science*, pages 128–137, Heidelberg, Germany, 2011. Springer.

[5] Trond Aalberg and Maja Žumer. Looking for Entities in Bibliographic Records. In George Buchanan, Masood Masoodian, and Sally Jo Cunningham, editors, *Proceedings of the 11th International Conference on Asian Digital Libraries*

(ICADL'08), volume 5362 of *Lecture Notes in Computer Science*, pages 327–330, Heidelberg, Germany, 2008. Springer.

[6] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39 – 59, 1994.

[7] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of the 5th ACM conference on Digital libraries (DL '00)*, pages 85–94, New York, NY, USA, 2000. ACM.

[8] Vladimir Alexiev, Michael Breu, Jos de Bruijn, Dieter Fensel, Ruben Lara, and Holger Lausen. *Information Integration with Ontologies: Experiences from an Industrial Showcase*. Wiley, 1st edition, 2005.

[9] Enrique Alfonseca, Marius Paşca, and Enrique Robledo-Arnuncio. Acquisition of instance attributes via labeled and related instances. In Fabio Crestani and Stéphane Marchand-Maillet and Hsin-Hsi Chen and Efthimis N. Efthimiadis and Jacques Savoy, editor, *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10)*, pages 58–65, New York, NY, USA, 2010. ACM.

[10] Alia K. Amin, Jacco van Ossenbruggen, Lynda Hardman, and Annelies van Nispen. Understanding cultural heritage experts' information seeking needs. In Ronald L. Larsen, Andreas Paepcke, José Luis Borbinha, and Mor Naaman, editors, *Proceedings of the 8th ACM/IEEE Joint Conference on Digital Libraries (JCDL '08)*, pages 39–47, New York, NY, USA, 2008. ACM.

[11] Douglas E. Appelt. Introduction to Information Extraction. *AI Communications*, 12(3):161–172, 1999.

[12] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC '07 and ASWC '07*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735, Heidelberg, Germany, 2007. Springer.

[13] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and

David Aumueller. Triplify: light-weight linked data publication from relational databases. In Juan Quemada and Gonzalo León and Yoëlle S. Maarek and Wolfgang Nejdl, editor, *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*, pages 621–630, New York, NY, USA, 2009. ACM.

[14] Krisztian Balog. *People Search in the Enterprise*. PhD thesis, University of Amsterdam, Amsterdam, Netherlands, 2008.

[15] Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. Overview of the TREC 2011 Entity Track. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of the 20th Text REtrieval Conference (TREC '11)*, Gaithersburg, MD, USA, 2012. National Institute of Standards and Technology.

[16] Krisztian Balog, Naimdjon Takhirov, Heri Ramampiaro, and Kjetil Nørvåg. Multi-step Classification Approaches to Cumulative Citation Recommendation. In *Proceedings of the 10th Open Research Areas in Information Retrieval (OAIR '13)*, pages 121–128, New York, NY, USA, 2013. ACM.

[17] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open Information Extraction from the Web. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI' 07)*, pages 2670–2676, Palo Alto, CA, USA, 2007. AAAI Press.

[18] Holger Bast, Alexandru Chitea, Fabian Suchanek, and Ingmar Weber. ESTER: efficient search on text, entities, and relations. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '07)*, pages 671–678, New York, NY, USA, 2007. ACM.

[19] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[20] Dave Beckett and Tim Berners-Lee. Turtle - Terse RDF Triple Language, W3C Team Submission, 2008. http://www.w3.org/TeamSubmission/turtle/.

[21] Nicolás García Belmonte. Creating interactive data visualizations for the web with the javascript infovis toolkit 2.0. In *YOW! Developer Conferences*, Brisbane, Australia, 2010.

[22] Rick Bennett, Brian F. Lavoie, and Edward T. O'Neill. The Concept of a Work

in WorldCat: An Application of FRBR. *Library Collections, Acquisitions, and Technical Services*, 27(1):45–59, 2003.

[23] Christian Bering, Witold Drozdzynski, Gregor Erbach, Clara Guasch, Petr Homola, Sabine Lehmann, Hong Li, Hans ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, Atsuko Shimada, Melanie Siegel, Feiyu Xu, and Dorothee Ziegler-eisele. Corpora and evaluation tools for multilingual named entity grammar development. In *Proceedings of Multilingual Corpora Workshop at Corpus Linguistics 2003*, pages 42–52, Lancaster, United Kingdom, 2003.

[24] Tim Berners-Lee. Getting into RDF & Semantic Web using N3. http://is.gd/fM5GN/, 2005.

[25] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 33(4):29–37, May 2001.

[26] Jagdev Bhogal, Andy Macfarlane, and Peter Smith. A review of ontology based query expansion. *Information Processing and Management: an International Journal*, 43(4):866–886, July 2007.

[27] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22, March 2009.

[28] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 2009.

[29] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia – A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7:154–165, September 2009.

[30] Roi Blanco, Peter Mika, and Sebastiano Vigna. Effective and efficient entity search in RDF data. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *Proceedings of the 10th international conference on the semantic web (ISWC '11)*, volume 7031 of Lecture Notes in Computer Science, pages 83–97, Heidelberg, 2011. Springer.

[31] Patrick Le Boeuf. FRBR and Further. *Cataloging & classification quarterly*, 32(4):15–52, 2001.

[32] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In Jason Tsong-Li Wang, editor, *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08)*, pages 1247–1250, New York, NY, USA, 2008. ACM.

[33] Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. Freebase: A Shared Database of Structured General Human Knowledge. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence AAAI (AAAI '07)*, pages 1962–1963, Palo Alto, CA, USA, 2007. AAAI Press.

[34] Andrew Eliot Borthwick. *A maximum entropy approach to named entity recognition*. PhD thesis, New York University, New York, NY, USA, 1999. AAI9945252.

[35] Jennifer Bowen. FRBR: coming soon to your library? *Library resources and technical services*, 49(3):175–188, 2005.

[36] Terje Brasethvik. *Conceptual Modeling for domain specific document description and retrieval - An approach to semantic document modeling*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2004.

[37] Sergey Brin. Extracting Patterns and Relations from the World Wide Web. In Paolo Atzeni, Alberto O. Mendelzon, and Giansalvatore Mecca, editors, *Proceedings of the International Workshop on the Web and Databases (WebDB '98)*, volume 1590 of *Lecture Notes in Computer Science*, pages 172–183, London, UK, 1999. Springer.

[38] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. Word-sense disambiguation using statistical methods. In Douglas E. Appelt, editor, *Proceedings of the 29th annual meeting on Association for Computational Linguistics (ACL '91)*, pages 264–270, Stroudsburg, PA, USA, 1991. Association for Computational Linguistics.

[39] George Buchanan, Jeremy Gow, Ann Blandford, Jon Rimmer, and Claire Warwick. Representing aggregate works in the digital library. In Edie M. Rasmussen, Ray R. Larson, Elaine G. Toms, and Shigeo Sugimoto, editors, *Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries (JCDL'07)*, pages 247–256, New York, NY, USA, 2007. ACM.

[40] Razvan C. Bunescu and Marius Paşca. Using Encyclopedic Knowledge for Named entity Disambiguation. In Diana McCarthy and Shuly Wintner, edi-

tors, *Proceedings of 11st Conference of the European Chapter of the Association for Computational Linguistics (EACL' 06)*, pages 9 – 16, Stroudsburg, PA, USA, 2006. The Association for Computational Linguistics.

[41] Kate Byrne. *Populating the Semantic Web – Combining Text and Relational Databases as RDF Graphs*. PhD thesis, University of Edinburgh, 2008.

[42] California Digital Library. The Melvyl Recommender Project. Technical report, University of California, 2006.

[43] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka Jr., and Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the 48th Annual Meeting on Association for Computational Linguistics (ACL '10)*, pages 1306–1313, Palo Alto, CA, USA, 2010. AAAI Press.

[44] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu, editors, *Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM '10)*, pages 101–110, New York, NY, USA, 2010. ACM.

[45] B. Chandrasekaran, John R. Josephson, and V. Richard Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14(1):20–26, January 1999.

[46] Surajit Chaudhuri, Venkatesh Ganti, and Rajeev Motwani. Robust Identification of Fuzzy Duplicates. In Karl Aberer, Michael J. Franklin, and Shojiro Nishio, editors, *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*, pages 865–876, Washington, DC, USA, 2005. IEEE Computer Society.

[47] Gong Cheng, Weiyi Ge, and Yuzhong Qu. Falcons: searching and browsing entities on the semantic web. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th international conference on World Wide Web (WWW '08)*, pages 1101–1102, New York, NY, USA, 2008. ACM.

[48] Tao Cheng, Xifeng Yan, and Kevin Chen-Chuan Chang. EntityRank: searching entities directly and holistically. In Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Flo-

rescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold, editors, *Proceedings of the 33rd international conference on Very Large Data Bases (VLDB '07)*, pages 387–398. VLDB Endowment, 2007.

[49] Marek Ciglan, Kjetil Nørvåg, and Ladislav Hluchý. The SemSets model for ad-hoc semantic list search. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *Proceedings of the 21st World Wide Web Conference 2012 (WWW '12)*, pages 131–140, New York, NY, USA, 2012. ACM.

[50] William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In Subbarao Kambhampati and Craig A. Knoblock, editors, *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb '03)*, pages 73–78, 2003.

[51] William W. Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-Markov extraction processes and data integration methods. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD '04)*, pages 89–98, New York, NY, USA, 2004. ACM.

[52] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC '99)*, pages 100 – 110. Association for Computational Linguistics, 1999.

[53] Gordon V. Cormack, Mark D. Smucker, and Charles L. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, October 2011.

[54] Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, January 1996.

[55] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In Donia Scott, Walter Daelemans, and Marilyn A. Walker, editors, *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL '04)*, pages 423 – 429, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[56] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan.

GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL '02)*, pages 168 – 175, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[57] Maria da Conceição Moraes Batista and Ana Carolina Salgado. Information Quality Measurement in Data Integration Schemas. In Venkatesh Ganti and Felix Naumann, editors, *Proceedings of the 5th International Workshop on Quality in Databases (QDB '07)*, pages 61–72, 2007.

[58] Bhavana Bharat Dalvi, William W. Cohen, and Jamie Callan. WebSets: extracting sets of entities from the web using unised information extraction. In Eytan Adar, Jaime Teevan, Eugene Agichtein, and Yoelle Maarek, editors, *Proceedings of the 5th ACM international conference on Web Search and Data Mining (WSDM '12)*, pages 243–252, New York, NY, USA, 2012. ACM.

[59] Nilesh Dalvi, Ravi Kumar, Bo Pang, Raghu Ramakrishnan, Andrew Tomkins, Philip Bohannon, Sathiya Keerthi, and Srujana Merugu. A web of concepts. In Jan Paredaens and Jianwen Su, editors, *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '09)*, pages 1–12, New York, NY, USA, 2009. ACM.

[60] Nilesh Dalvi, Ravi Kumar, and Mohamed Soliman. Automatic wrappers for large scale web extraction. *Proceedings VLDB Endowment*, 4(4):219–230, January 2011.

[61] Olivier Dameron, Daniel Rubin, and Mark Musen. Challenges in converting frame-based ontology into owl: the foundational model of anatomy case-study. *AMIA Annual Symposium Proceedings*, pages 181–185, 2005.

[62] Ian Davis, Bruce D'Arcus, and Richard Newman. Expression of Core FRBR Concepts in RDF. http://vocab.org/frbr/core.html, 2005.

[63] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of 6th osdi 20the Symposium on Operating System Design and Implementation (OSDI '04)*, pages 137–150. USENIX Association, 2004.

[64] Gianluca Demartini. *From people to entities: typed search in the enterprise and the web*. PhD thesis, Leibniz Universität Hannover, 2011.

[65] Lorcan Dempsey and Rachel Heery. Metadata: a current view of practice and issues. *Journal of Documentation*, 54(2):145–172, 1998.

[66] Martin Doerr. The cidoc conceptual reference module: An ontological approach to semantic interoperability of metadata. *AI Magazine*, 24(3):75–92, 2003.

[67] Martin Doerr and Patrick LeBoeuf. FRBRoo Introduction. *ICOM-CIDOC, version 0.8.1*, 2007.

[68] Christian Drumm, Matthias Schmitt, and Erhard Rahm. QuickMig - Automatic Schema Matching for Data Migration Projects. In Mário J. Silva and Alberto H. F. Laender and Ricardo A. Baeza-Yates and Deborah L. McGuinness and Bjørn Olstad and Øystein Haug Olsen and André O. Falcão, editor, *Proceedings of the 16th Conference on Information and Knowledge Management (CIKM '07)*, pages 107 – 116, New York, NY, USA, 2007. ACM.

[69] Fabien Duchateau, Naimdjon Takhirov, and Trond Aalberg. FRBRPedia: a Tool for FRBRizing Web products and Linking FRBR Entities to DBpedia. In Glen Newton, Michael J. Wright, and Lillian N. Cassel, editors, *Proceedings of the 11th ACM/IEEE Joint International Conference on Digital Libraries (JCDL '11)*, pages 455–456, New York, NY, USA, 2011. ACM.

[70] Gordon Dunsire. FRBRer model. http://metadataregistry.org/schema/show/id/5.html, February 2009.

[71] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open Information Extraction from the Web. *Communication of ACM*, 51(12):68–74, 2008.

[72] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Journal of Artificial Intelligence*, 165:91–134, June 2005.

[73] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: the second generation. In Toby Walsh, editor, *Proceedings of the 22nd international joint conference on Artificial Intelligence (IJCAI '11) - Volume Volume One*, pages 3–10, Palo Alto, CA, USA, 2011. AAAI Press.

[74] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, Heidelberg, Germany, 2007.

[75] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying Relations for Open Information Extraction. In Regina Barzilay and Mark Johnson, editors, *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP '11)*, pages 1535–1545, Palo Alto, CA, USA, July 27-31 2011. AAAI Press.

[76] Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(238):1183–1210, 1969.

[77] Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. A Statistical Model for Multilingual Entity Detection and Tracking. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL '04)*, pages 1–8, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[78] Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the 7th conference on Natural language learning at HLT-NAACL '03 - Volume 4 (CONLL '03)*, pages 168–171, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[79] Allen Foster and Nigel Ford. Serendipity and Information Seeking: An Empirical Study. *Journal of Documentation*, 59(3):321–340, January 2003.

[80] Massimo Franceschet, Donatella Gubiani, Angelo Montanari, and Carla Piazza. From Entity Relationship to XML Schema: A Graph-Theoretic Approach. In Zohra Bellahsene, Ela Hunt, Michael Rys, and Rainer Unland, editors, *Proceedings of the 6th International XML Database Symposium (XSym '09)*, volume 5679 of *Lecture Notes in Computer Science*, pages 165–179, Heidelberg, Germany, 2009. Springer.

[81] John R. Frank, Max Kleiman-Weiner, Daniel A. Roberts, Feng Niu, Ce Zhang, Christopher Ré, and Ian Soboroff. Building an Entity-Centric Stream Filtering Test Collection for TREC 2012. In *Proceedings of the 20th Text REtrieval Conference (TREC 2012)*, Gaithersburg, MD, USA, 2013. National Institute of Standards and Technology.

[82] Nuno Freire, José Luis Borbinha, and Pável Calado. Identification of FRBR Works Within Bibliographic Databases: An Experiment with UNIMARC and Duplicate Detection Techniques. In Dion Hoe-Lian Goh, Tru Hoang Cao, Ingeborg Sølvberg, and Edie M. Rasmussen, editors, *Proceedings of the 10th International Conference on Asian Digital Libraries (ICADL '07)*, volume 4822 of *Lecture Notes in Computer Science*, pages 267–276, Heidelberg, Germany, 2007. Springer.

[83] Nuno Miguel Antunes Freire. *Entity Recognition and Resolution in Poorly Structured Data*. PhD thesis, Instituto Superior Técnico da Universidade Técnica de Lisboa, Lisboa, Portugal, 2012.

[84] Richard B. Fuller and John McHale. *World Design Science Decade, 1965-1975:*. Number 3-6 in Five Two-year Phases of a World Retooling Design Proposed to the International Union of Architects for Adoption by World Architectural Schools, Southern Illinois University at Carbondale. World Resources Inventory, 1965.

[85] Stephen R. Garner. WEKA: The Waikato Environment for Knowledge Analysis. In *Proceedings of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995.

[86] Anna Gerber and Jane Hunter. A compound object authoring and publishing tool for literary scholars based on the IFLA-FRBR model. *International Journal of Digital Curation*, 4(2), 2009.

[87] Shlomo Geva, Jaap Kamps, Ralf Schenkel, and Andrew Trotman. Focused Retrieval and Evaluation. In *Working Notes of 9th International Workshop of the Inititative for the Evaluation of XML Retrieval (INEX '10)*, volume 6932 of *Lecture Notes in Computer Science*, Heidelberg, Germany, 2010. Springer.

[88] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Journal of Computational Linguistics*, 32:83–135, March 2006.

[89] Swapna Gottipati and Jing Jiang. Linking entities to a knowledge base with query expansion. In Regina Barzilay and Mark Johnson, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, pages 804–813, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[90] Peter H. Gray. The impact of knowledge repositories on power and control in the workplace. *Information technology & People*, 14(4):368–384, 2001.

[91] Ralph Grishman. Information Extraction: Techniques and Challenges. In Maria Teresa Pazienza, editor, *International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology (SCIE '97)*, volume 1299 of *Lecture Notes in Computer Science*, pages 10–27, Heidelberg, Germany, 1997. Springer.

[92] Ralph Grishman and Beth Sundheim. Message Understanding Conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics - Volume 1 (COLING '96)*, pages 466–471, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.

[94] Jon Atle Gulla, Hans Olaf Borch, and Jon Espen Ingvaldsen. Ontology learning for search applications. In Robert Meersman and Zahir Tari, editors, *OOTM Confederated International Conferences (1)*, volume 4803 of *Lecture Notes in Computer Science*, pages 1050–1062, Heidelberg, Germany, 2007. Springer.

[95] Jon Atle Gulla, Jin Liu, Felix Burkhardt, Jianshen Zhou, Christian Weiss, Per Myrseth, Veronika Haderlein, and Olga Cerrato. *Applied Semantic Web Technologies*, volume 2 of *Applied Semantic Web Technologies*, chapter Semantics and Search, pages 347–378. Taylor & Francis (Auerbach Publications), 2012.

[96] Jacques Guyot, Saïd Radhouani, and Gilles Falquet. Ontology-based multilingual information retrieval. In Carol Peters, Fredric C. Gey, Julio Gonzalo, Henning Müller, Gareth J. F. Jones, Michael Kluck, Bernardo Magnini, and Maarten de Rijke, editors, *Cross-Language Evaluation Forum, Working Notes Multilingual Track (CLEF '05)*, volume 4022 of *Lecture Notes in Computer Science*, pages 21–23, Heidelberg, Germany, 2005. Springer.

[97] Xianpei Han and Jun Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors, *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09)*, pages 215–224, New York, NY, USA, 2009. ACM.

[98] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations

among named entities from large corpora. In Donia Scott, Walter Daelemans, and Marilyn A. Walker, editors, *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL '04)*, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[99] O. Hassanzadeh and M. P. Consens. Linked Movie Data Base. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Kingsley Idehen, editors, *Proceedings of Linked Data on the Web (LDOW '09)*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[100] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics (COLING '92) - Volume 2*, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[101] Knut Hegna and Eeva Murtomaa. Data mining MARC to find: FRBR? In *Proceedings of the 68th IFLA General Conference and Council*, Glasgow, Scotland, 2002.

[102] Alan R. Hevner and Samir Chatterjee. *Design Research in Information Systems: Theory and Practice*. Integrated Series in Information Systems. Springer Publishing Company, Incorporated, New York, NY, USA, 1st edition, 2010.

[103] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.

[104] Thomas B. Hickey, Edward T. O'Neill, and Jenny Toves. Experiments with the IFLA Functional Requirements for Bibliographic Records (FRBR). *D-Lib Magazine*, 8(9), 2002.

[105] Geoffrey E. Hinton and Terrence Joseph Sejnowski. *Unsupervised Learning: Foundations of Neural Computation*. Computational Neuroscience. MIT Press, 1999.

[106] Lynette Hirschman and Robert Gaizauskas. Natural Language Question Answering: the View from Here. *Natural Language Engineering*, 7(4):275–300, December 2001.

[107] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In Regina Barzilay and Mark Johnson, editors, *Proceedings of the Conference on Empirical Methods*

in Natural Language Processing (EMNLP '11), pages 782–792, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[108] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):7–26, 12 2003.

[109] Eero Hyvönen. *Publishing and Using Cultural Heritage Linked Data on the Semantic Web*. Synthesis Lectures on Semantic Web: Theory and Technology. Morgan & Claypool, Palo Alto, CA, 2012. Available as paperback and ebook (9781608459988).

[110] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24(1):2–40, March 1998.

[111] Frank M. Shipman III and Raymond McCall. Supporting knowledge-base evolution with incremental formalization. In Beth Adelson, Susan T. Dumais, and Judith S. Olson, editors, *CHI*, pages 285–291. ACM, 1994.

[112] Mikael R. Jensen, Thomas H. Møller, and Torben Bach Pedersen. Converting xml dtds to uml diagrams for conceptual data integration. *Data and Knowledge Engineering*, 44(3):323–346, 2003.

[113] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data and Knowledge Engineering*, 69:197–210, February 2010.

[114] Dimitrios A. Koutsomitropoulos, Georgia D. Solomou, Andreas D. Alexopoulos, and Theodore S. Papatheodorou. Semantic web enabled digital repositories. *International Journal on Digital Libraries*, 10(4):179–199, 2009.

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

[116] Ravi Kumar and Andrew Tomkins. A Characterization of Online Search Behavior. *IEEE Data Engineering Bulletin*, 32(2):3–11, 2009.

[117] Nicholas Kushmerick. *Wrapper induction for information extraction*. PhD thesis, University of Washington, 1997. AAI9819266.

[118] Nicholas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI '97) - Volume One*, pages 729–737, Burlington, MA, USA, 1997. Morgan Kaufmann Publishers Inc.

[119] Carl Lagoze, Herbert Van de Sompel, Michael L. Nelson, Simeon Warner, Robert Sanderson, and Pete Johnston. Object Re-Use & Exchange: A Resource-Centric Approach. *CoRR*, abs/0804.2273, 2008.

[120] Carl Lagoze, Herbert Van de Sompel, Michael L. Nelson, Simeon Warner, Robert Sanderson, and Pete Johnston. A Web-based resource model for scholarship 2.0: object reuse & exchange. *Concurrency and Computation: Practice and Experience*, 24(18):2221–2240, 2012.

[121] Douglas B. Lenat. CYC: a large-scale investment in knowledge infrastructure. *Communication of ACM*, 38(11):33–38, November 1995.

[122] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In Virginia De Buys, editor, *Proceedings of the 5th annual international conference on Systems documentation (SIGDOC '86)*, pages 24 – 26, New York, NY, USA, 1986. ACM.

[123] Vladimir I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Journal of Soviet Physics Doklady*, 10:707, 1966.

[124] Library of Congress. MARC 21 Specifications for Records Structure, Character Sets and Exchange Media. *Network Development and MARC Standards Office*, January 2001.

[125] Library of Congress Network Development and MARC Standards Office. Functional Analysis of the MARC 21 Bibliographic and Holdings Formats. http://www.loc.gov/marc/marc-functional-analysis/functional-analysis.html, April 2006.

[126] Thomas Lin, Oren Etzioni, and James Fogarty. Identifying interesting assertions from the web. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors, *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09)*, pages 1787–1790, New York, NY, USA, 2009. ACM.

[127] Qiaoling Liu, Kaifeng Xu, Lei Zhang, Haofen Wang, Yong Yu, and Yue Pan. Catriple: Extracting Triples from Wikipedia Categories. In John Domingue and Chutiporn Anutariya, editors, *Proceedings of The Semantic Web, 3rd Asian Semantic Web Conference (ASWC '08)*, volume 5367 of *Lecture Notes in Computer Science*, pages 330–344, Heidelberg, Germany, 2008. Springer.

[128] Peter C. Lockemann, Heinrich C. Mayr, Wolfgang H. Weil, and Wolfgang H. Wohlleber. Data abstractions for database systems. *ACM Transaction Database Systems*, 4(1):60–75, March 1979.

[129] Thomas R. Lynam, Gordon V. Cormack, and David R. Cheriton. On-line spam filter fusion. In Efthimis Efthimiadis and Susan Dumais and David Hawking and Kalervo Järvelin, editor, *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '06)*, pages 123–130, New York, NY, USA, 2006. ACM.

[130] Alexander Maedche and Steffen Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[131] Martin Malmsten. Making a library catalogue part of the semantic web. In *Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications (DCMI '08)*, pages 146–152. Dublin Core Metadata Initiative, 2008.

[132] Hugo Miguel Álvaro Manguinhas, Nuno Miguel Antunes Freire, and José Luis Brinquete Borbinha. FRBRization of MARC records in multiple catalogs. In Jane Hunter, Carl Lagoze, C. Lee Giles, and Yuan-Fang Li, editors, *Proceedings of the 10th ACM/IEEE Joint Conference on Digital Libraries (JCDL '10)*, pages 225–234, New York, NY, USA, 2010. ACM.

[133] Cynthia Matuszek, Michael J. Witbrock, Robert C. Kahlert, John Cabral, David Schneider, Purvesh Shah, and Douglas B. Lenat. Searching for Common Sense: Populating Cyc; from the Web. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings of The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI '05)*, pages 1430–1435, Palo Alto, CA, USA, 2005. AAAI Press / The MIT Press.

[134] Diana Maynard and Hamish Cunningham. Multilingual adaptations of annie, a reusable information extraction tool. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume*

2 (EACL ’03), pages 219–222, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[135] Diana Maynard, Hamish Cunningham, Kalina Bontcheva, Roberta Catizone, George Demetriou, Robert Gaizauskas, Oana Hamza, Mark Hepple, and Patrick Herring. A Survey of Uses of GATE. Technical report, Department of Computer Science, University of Sheffield, 2000.

[136] Diana Maynard, Adam Funk, and Wim Peters. Using Lexico-Syntactic Ontology Design Patterns for Ontology Creation and Population. In Eva Blomqvist, Kurt Sandkuhl, François Scharffe, and Vojtech Svátek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP ’09)*, volume 516 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[137] Diana Maynard and Mark A. Greenwood. Large Scale Semantic Annotation, Indexing and Search at The National Archives. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC ’12)*, Turkey, May 2012. European Language Resources Association (ELRA).

[138] Diana Gabrielle Maynard. *Term Recognition Using Combined Knowledge Sources*. PhD thesis, Manchester Metropolitan University, Manchester, United Kingdom, June 2000.

[139] Kelly Mcgrath and Lynne Bisko. Identifying FRBR Work-Level Data in MARC Bibliographic Records for Manifestations of Moving Images. *The Code4Lib Journal*, 1(5), December 2008.

[140] Deborah L. McGuinness. Ontologies Come of Age. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web*, pages 171–194. MIT Press, 2003.

[141] Edgar Justin Meij. *Combining Concepts and Language Models for Information Access*. PhD thesis, University of Amsterdam, Amsterdam, Netherlands, 2010.

[142] Pablo N. Mendes, Max Jakob, and Christian Bizer. DBpedia: A Multilingual Cross-domain Knowledge Base. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC’2012)*, pages 1813–1817. European Language Resources Association (ELRA), 2012.

[143] Roberta Merchant, Mary Ellen Okurowski, and Nancy Chinchor. The multi-

lingual entity task (MET) overview. In *Proceedings of TIPSTER Text Program (Phase II) (TIPSTER '96)*, pages 445–447, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[144] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In Mário J. Silva and Alberto H. F. Laender and Ricardo A. Baeza-Yates and Deborah L. McGuinness and Bjørn Olstad and Øystein Haug Olsen and André O. Falcão, editor, *Proceedings of the 16th Conference on Information and Knowledge Management (CIKM '07)*, pages 233–242, New York, NY, USA, 2007. ACM.

[145] Peter Mika and Tim Potter. Metadata statistics for a large web corpus. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *Proceedings of the WWW2012 Workshop on Linked Data on the Web (LDOW '12)*, volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[146] Eric Miller, Uche Ogbuji, Victoria Mueller, and Kathy MacDougall. Bibliographic Framework as a Web of Data: Linked Data Model and Supporting Services. Technical report, Library of Congress, Washington, DC, USA, 2012.

[147] David Milne and Ian H. Witten. Learning to link with Wikipedia. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08)*, pages 509–518, New York, NY, USA, 2008. ACM.

[148] Marvin Minsky. A Framework for Representing Knowledge. Technical report, MIT AI Laboratory, Cambridge, MA, USA, 1974.

[149] T. Mitchell. *Machine Learning*. McGraw-Hill Education (ISE Editions), Oct. 1997.

[150] Alvaro E. Monge and Charles Elkan. The Field Matching Problem: Algorithms and Applications. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD '96)*, pages 267–270, Menlo Park, CA, USA, 1996. AAAI Press.

[151] Hannes Mühleisen and Christian Bizer. Web Data Commons - Extracting Structured Data from Two Large Web Corpora. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *Proceedings of the WWW2012*

Workshop on Linked Data on the Web (LDOW '12), volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[152] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Find your Advisor: Robust Knowledge Gathering from the Web. In Xin Luna Dong and Felix Naumann, editors, *Proceedings of the 13th International Workshop on the Web and Databases (WebDB '10)*, 2010.

[153] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In Irwin King, Wolfgang Nejdl, and Hang Li, editors, *Proceedings of the 4th ACM international conference on Web Search and Data Mining (WSDM '11)*, pages 227–236, New York, NY, USA, 2011. ACM.

[154] Ndapandula Nakashole and Gerhard Weikum. Real-time Population of Knowledge Bases: Opportunities and Challenges. In *The 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, The Knowledge Extraction Workshop*. ACL, 2012.

[155] Ndapandula T. Nakashole. *Automatic Extraction of Facts, Relations, and Entities for Web-Scale Knowledge Base Population*. PhD thesis, Saarland University, Max-Planck Institute for Informatics, Saarbrücken, Germany, December 2012.

[156] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Survey*, 41(2):10:1–10:69, February 2009.

[157] Roberto Navigli and Paola Velardi. Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1075–1086, July 2005.

[158] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. On the Modeling of Entities for Ad-Hoc Entity Search in the Web of Data. In Ricardo A. Baeza-Yates, Arjen P. de Vries, Hugo Zaragoza, Berkant Barla Cambazoglu, Vanessa Murdock, Ronny Lempel, and Fabrizio Silvestri, editors, *Proceedings of the 34th European Conference on Information Retrieval (ECIR '12)*, volume 7224 of *Lecture Notes in Computer Science*, pages 133–145, Heidelberg, Germany, 2012. Springer.

[159] Robert Neumeyer. *Semantic and Distributed Entity Search in the Web of Data*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2013.

[160] Natalya F. Noy. Semantic integration: a survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, December 2004.

[161] Natalya Fridman Noy, AnHai Doan, and Alon Y. Halevy. Semantic Integration. *AI Magazine*, 26(1):7–10, 2005.

[162] Marius Paşca. Outclassing wikipedia in open-domain information extraction: weakly-supervised acquisition of attributes over conceptual hierarchies. In Alex Lascarides, Claire Gardent, and Joakim Nivre, editors, *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)*, pages 639–647, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[163] Patrick Pantel and Marco Pennacchiotti. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL '06)*, pages 113–120, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[164] Aditya Parameswaran, Hector Garcia-Molina, and Anand Rajaraman. Towards the web of concepts: extracting concepts from large datasets. *VLDB Endowment*, 3:566–577, September 2010.

[165] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, December 2007.

[166] Guy Pierra, Jean-Claude Potier, and Eric Sardet. From digital libraries to electronic catalogues for engineering and manufacturing. *International Journal of Computer Applications in Technology (IJCAT)*, 18(1-4):27–42, 2000.

[167] Jan Pisanski, Maja Žumer, and Trond Aalberg. Identifiers: bridging language barriers. *International Cataloguing and Bibliographic Control*, 40(1), January/-March 2011.

[168] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the web of data. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabart, editors, *Proceedings of the 19th international conference on World Wide Web (WWW '10)*, pages 771–780, New York, NY, USA, 2010. ACM.

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

[170] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[171] Jenn Riley. Enhancing Interoperability of FRBR-Based Metadata. In *Proceedings of the International Conference on Dublin Core and Metadata Applications (DC '10)*, Pittsburgh, PA, USA, October 2010. DCMI.

[172] Henning Rode, Pavel Serdyukov, and Djoerd Hiemstra. Combining document- and paragraph-based entity ranking. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pages 851–852, New York, NY, USA, 2008. ACM.

[173] Matthew Rowe and Fabio Ciravegna. Data.dcs: Converting Legacy Data into Linked Data. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *Proceedings of the WWW2010 Workshop on Linked Data on the Web (LDOW'10)*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[174] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A Survey of Current Approaches for Mapping of Relational Databases to RDF. Technical report, The World Wide Web Consortium (W3C), 2009.

[175] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. LINDEN: linking named entities with knowledge base via semantic knowledge. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *Proceedings of the 21st international conference on World Wide Web (WWW '12)*, pages 449–458, New York, NY, USA, 2012. ACM.

[176] Geir Solskinnsbakk. *Contextual Semantic Search Navigation*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2012.

[177] Darijus Strasunskas and Stein L. Tomassen. The role of ontology in enhancing semantic searches: the EvOQS framework and its initial validation. *IJKL*, 4(4):398–414, 2008.

[178] Rob Styles, Danny Ayers, and Nadeem Shabir. Semantic MARC, MARC21 and the Semantic Web. In Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee, editors, *Proceedings of the WWW2008 Workshop on Linked Data on the Web (LDOW '08)*, volume 369 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[179] Fabian M. Suchanek. *Automated Construction and Growth of a Large Ontology*. PhD thesis, Saarland University, Max-Planck Institute for Informatics, Saarbrücken, Germany, 2009.

[180] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th international conference on World Wide Web (WWW '07)*, pages 697–706, New York, NY, USA, 2007. ACM.

[181] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. SOFIE: a self-organizing framework for information extraction. In Juan Quemada and Gonzalo León and Yoëlle S. Maarek and Wolfgang Nejdl, editor, *Proceedings of the 18th international conference on World Wide Web (WWW '09)*, pages 631–640, New York, NY, USA, 2009. ACM.

[182] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[183] Sari Suomela and Jaana Kekäläinen. Ontology as a search-tool: a study of real users' query formulation with and without conceptual support. In David E. Losada and Juan M. Fernández-Luna, editors, *Proceedings of the 27th European conference on Advances in Information Retrieval Research (ECIR '05)*, volume 3408 of *Lecture Notes in Computer Science*, pages 315–329, Heidelberg, Germany, 2005. Springer.

[184] Naimdjon Takhirov, Fabien Duchateau, and Trond Aalberg. Linking FRBR entities to LOD through semantic matching. In Stefan Gradmann, Francesca Borri, Carlo Meghini, and Heiko Schuldt, editors, *Proceedings of the 15th international conference on Theory and Practice of Digital Libraries (TPDL '11)*, volume 6966 of *Lecture Notes in Computer Science*, pages 284–295, Heidelberg, Germany, 2011. Springer.

[185] Naimdjon Takhirov, Fabien Duchateau, and Trond Aalberg. An evidence-based verification approach to extract entities and relations for knowledge base population. 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, *Proceedings of the 11th international conference on The Semantic Web (ISWC'12) - Volume Part I*, volume 7649 of *Lecture Notes in Computer Science*, pages 575–590, Heidelberg, Germany, 2012. Springer.

[186] Naimdjon Takhirov, Fabien Duchateau, Trond Aalberg, and Ingeborg Sølvberg. An Integrated Approach for Large-scale Relation Extraction from the Web. In Yoshiharu Ishikawa, Jianzhong Li, Wei Wang, Rui Zhang, and Wenjie Zhang, editors, *Proceedings of the Web Technologies and Applications – 15th International Asia-Pacific Web Conference (APWeb '13)*, volume 7808 of *Lecture Notes in Computer Science*, pages 163–175, Heidelberg, Germany, 2013. Springer.

[187] Roy Tennant. *XML in Libraries*. Neal-Schuman Publishers, New York, 2002.

[188] The International Federation of Library Associations and Institutions. Functional Requirements for Bibliographic Records. *UBCIM Publications - New Series*, 19, 1998.

[189] Hayssam Traboulsi. Arabic named entity extraction: A local grammar-based approach. In *Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT '09)*, pages 139–143. IEEE, 2009.

[190] Thanh Tran, Philipp Cimiano, Sebastian Rudolph, and Rudi Studer. Ontology-based interpretation of keywords for semantic search. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *Proceedings of the 6th international the semantic web and 2nd Asian conference on Asian semantic web conference (ISWC '07 / ASWC '07)*, volume 4825 of *Lecture Notes in Computer Science*, pages 523–536, Heidelberg, Germany, 2007. Springer.

[191] Dionysios C. Tsichritzis and Frederick H. Lochovsky. *Data Models*. Prentice Hall Professional Technical Reference, 1982.

[192] Anne-Marie Vercoustre, James A. Thom, and Jovan Pehcevski. Entity ranking in Wikipedia. In Roger L. Wainwright and Hisham Haddad, editors, *Proceedings of*

the 2008 ACM Symposium on Applied Computing (SAC '08), pages 1101–1106, New York, NY, USA, 2008. ACM.

[193] Ellen M. Voorhees. The trec question answering track. *Journal of Natural Language Engineering*, 7(4):361–378, December 2001.

[194] Ellen M. Voorhees. TREC Experiment and Evaluation in Information Retrieval. In Lori P. Buckland Ellen M. Voorhees, editor, *Text REtrieval Conference*. National Institute of Standards and Technology, USA, 2005.

[195] Maja Žumer. FRBR: The End of the Road or a New Beginning. *ASIS&T Bulletin*, August/September, 2007.

[196] John N. Warfield. *A science of generic design: managing complexity through systems design*. Number v. 1 in Systems Inquiry Series. Intersystems Publications, 1990.

[197] Jakub Waszczuk, Katarzyna Glowinska, Agata Savary, and Adam Przepiórkowski. Tools and methodologies for annotating syntax and named entities in the national corpus of polish. In *International Multiconference on Computer Science and Information Technology (IMCSIT '10)*, pages 531–539. IEEE, 2010.

[198] Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from web sources. In Jan Paredaens and Dirk Van Gucht, editors, *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '10)*, pages 65–76, New York, NY, USA, 2010. ACM.

[199] Daya C. Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 36(3):306–323, June 2010.

[200] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Zhu. Probase: A probabilistic taxonomy for text understanding. In K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman, editors, *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12)*, pages 109–120, New York, NY, USA, 2012. ACM.

[201] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In Hans Uszkoreit, editor, *Proceedings of the 33rd annual meeting on*

Association for Computational Linguistics (ACL '95), pages 189–196, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[202] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, March 2003.

[203] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02)*, pages 473–480, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.