Generation of Multilingual Ontology Lexica with M-ATOLL

A corpus-based approach for the induction of ontology lexica

Sebastian Josef Walter

Bielefeld University

Submitted in partial fulfillment of the requirements for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

October 2016

Abstract

There is an increasing interest in providing common web users with access to structured knowledge bases such as DBpedia, for example by means of question answering systems. All such question answering systems have in common that they have to map a natural language input, be it spoken or written, to a formal representation in order to extract the correct answer from the target knowledge base. This is also the case for systems which generate natural language text from a given knowledge base. The main challenge is how to map natural language (spoken or written) to structured data and vice versa. To this end, question answering systems require knowledge about how the vocabulary elements used in the available datasets are verbalized in natural language, covering different verbalization variants. Multilinguality of course increases the complexity of this challenge. In this thesis we introduce M-ATOLL, a framework for automatically inducing ontology lexica in multiple languages, to find such verbalization variants. We have instantiated the system for three languages, English, German and Spanish, by exploiting a set of language-specific dependency patterns for finding lexicalizations in text corpora. Additionally, we extended our framework to extract complex adjective lexicalizations with a machine-learning-based approach. M-ATOLL is the first open-source and multilingual approach for the generation of ontology lexica. In this thesis we present grammatical patterns for three different languages, on which the extraction of lexicalization relies. We provide an analysis of these patterns as well as a comparison with those proposed by other state-of-the-art systems. Additionally, we present a detailed evaluation comparing the different approaches with different settings on a publicly available goldstandard, and discuss their potential and limitations.

Printed on non-ageing paper `ISO 9706`

Acknowledgements

The work presented in this thesis was carried out in the Semantic Computing Group, headed by Prof. Dr. Philipp Cimiano, at the Faculty of Technology, Bielefeld University. This work was supported by the Cluster of Excellence Cognitive Interaction Technology *CITEC* (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

First of all I would like to thank my supervisor Prof. Dr. Philipp Cimiano for giving me the opportunity to work with him for more than 6 years. He guided me since my start as a master student and offered me the possibility to write a PhD in his group. Philipp Cimiano pushed me if necessary and I am very grateful for the many interesting and fruitful discussions we had.

I also want to say thank you to my external reviewer Prof. Dr. Paul Buitelaar from Galway, Ireland, for agreeing to review my thesis, as well as traveling to Bielefeld for the disputation.

Special thanks go to my colleagues Dr. Christina Unger and Dr. Matthias Hartung from our research group for the time they spent in order to give me valuable feedback for my thesis.

Last but not least I would like to thank my family, especially my parents, and my close friends for their unlimited support and encouragement during the last few years.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

CRF	Conditional Random Fields
EL	Entity Linking
ER	Entity Recognition
IE	Information Extraction
LOD	Linked Open Data
NEL	Named Entity Linking
NER	Named Entity Recognition
NLP	Natural Language Processing
OBIE	Ontology-based Information Extraction
OpenIE	Open Information Extraction
POS	Part of Speech
QA	Question Answering
RDF	Resource Description Framework
SVM	Support Vector Machine
URI	Uniformed Resource Identifiers
XML	eXtensible Markup Language

Chapter 1

Introduction

This chapter gives a short introduction to the challenge addressed in this thesis, and provides a formal task definition. We close this chapter with the research questions we address as well as a detailed list of contributions.

1.1 Motivation

In recent years the interest in natural language-based communication with technology devices, such as mobile phones, has increased. By "natural language-based communication" we refer to verbal or written communication without the need to manipulate a device with hands or gestures. Prominent examples of such systems on mobile phones are *Google Now*, *Apple's Siri* and *Samsung S Voice*. Also, automotive manufacturers increasingly develop techniques to enable the driver to communicate with the car, for example for altering the route in the navigation system, or calling a relative, using the own voice to execute the necessary commands (as example see the work by Kun et al. (2013) and Schalk and Gibbons (2013)).

In the research community these systems are investigated mainly under the topic of Question Answering (QA). The goal of QA is to provide an answer for a given question using text documents or a structured knowledge base. Especially QA systems that provide access to structured knowledge have recently attracted attention. A very prominent example for such a system is *Watson* from IBM, which is introduced in the work by Ferrucci et al. (2010). In the research community there are systems like *QAKiS* by Cabrio et al. (2012), *PowerAqua* by Lopez et al. (2012), *TBSL* by Unger et al. (2012), and *Bela* by Walter et al. (2012), which make use of structural databases in the background.

All these systems have in common that they have to map natural language input, be it spoken or written, to a formal representation in order to extract the correct answer from a database. This is also the case for systems which generate a natural language output for data from databases.

All systems have the same challenge to solve:

How to map from natural language (spoken or written) to a structured database and vice versa?

Consider the following natural language examples, with corresponding SPARQL queries.

(1.1) `How tall is Michael Jordan?`

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
SELECT DISTINCT ?num
WHERE {
        res:Michael_Jordan dbo:height ?num .
}
```

Here the term `tall` is represented by the property *dbo:height* in the SPARQL query.

Consider further the following example question:

(1.2) `In which country does the Nile start?`

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
SELECT DISTINCT ?uri
WHERE {
        res:Nile dbo:sourceCountry ?uri .
}
```

For this question the main challenge consists in mapping the phrases `country` and `does start` to the property *dbo:sourceCountry*.

Consider a third example:

(1.3) `Give me all Spanish films.`

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?uri
WHERE {
        ?uri rdf:type dbo:Film .
        ?uri dbo:country res:Spain .
}
```

This example is even more challenging than the previous examples. Besides creating the correct SPARQL syntax, it requires a mapping between `Spanish` and the property/resource combination *dbo:country res:Spain*. The complexity is given by the fact that the term `Spanish` has a meaning which is restricted to a certain resource. This means that the term `Spanish` can only be formalized using the property *dbo:country* with the restriction to the resource *res:Spain*.

These three examples highlight two interrelated challenges. The first challenge is to actually transform the natural language input to a formal representation. This has to be done by considering the vocabulary and structure of the given knowledge base. Also, and here the second challenge arises, terms and phrases from the natural language have to be mapped to the vocabulary used by the knowledge base. As presented in the previous examples there is often a mismatch between the input terms or phrases and the actual terms used in the structured database. We refer to this challenge as **lexical gap**. Imagine a user who wants to get an answer to the question `How high is the Mount Everest?` The systems mentioned above would presumably identify the terms `high` and `Mount Everest` as important and search the target databases for both terms. If the database contains a relation *high* with an entry for `Mount Everest`, the database will probably return the intended answer. The challenge involved in bridging the lexical gap becomes obvious in the moment when no direct mapping of both terms is possible. In this case lexicalizations of database relations (properties) are needed in order to map from the natural language input to the relations in the database. We regard a *lexicalization* as a relation between a lexical entry and some ontology entity which the lexical entry verbalizes. We refer to this relation as *Lex.* A lexicalization represents a $m : n$ relation as shown in Figure 1.1. The adjective lexical entry `high`, for instance, can refer to two properties, that is *dbo:elevation* and *dbo:height*, while the property *dbo:height* can be expressed by two lexical entries, `elevation` and `high`. The challenge involved in bridging the the lexical gap is to map the ontology terms on the right side, considering as context the data within the structured database, to the variants on the left side.

The complexity of the challenge increases when mapping input from different languages, such as German or Chinese, to a structured database in a different language, for example provided in English. This challenge can partly be solved by using machine translation techniques to map in real time from a term in German or Chinese to the correct English relation in a database.

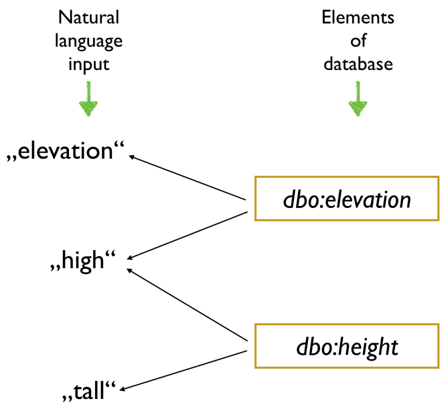


Figure 1.1: Illustration of the lexical gap.

However, this again fails if the difference between the translated input term and the name of the relation in the database is too big.

Through the rise of available structured datasets, in public but also in closed environments, the necessity to develop systems that bridge the lexical gap effectively is rising. Also the need for individual mappings to those datasets increases. Especially with the rise of publicly available structured datasets, like DBpedia and Freebase, not only domain specific datasets are available, but datasets covering a huge variety of different domains. Especially DBpedia, introduced by Auer et al. (2007), is now a widely used resource. According to Lehmann et al. (2015), the community behind DBpedia extracts multilingual knowledge for more than 111 languages from Wikipedia and stores it in a database using Semantic Web and Linked Data techniques. Those datasets have been used in the research community by question answering systems such as mentioned above.

As many question answering systems need to tackle the lexical gap in order to increase the coverage and performance, the need for systems finding *lexicalizations* is evident and has therefore gained interest in the research community. The need for lexicalizations in order to solve the lexical gap has been evaluated in detail in the survey by Hoeffner et al. (2016).

With the previously mentioned dataset, also an ontology, which defines the different relations in the structured database, is delivered. For the ontology provided with DBpedia, the community has mapped Wikipedia infoboxes from more than 27 language editions to one single ontology (see Lehmann et al. (2015)). The goal then is to close the lexical gap by finding all valid lexicalizations of a given ontology entity.

Formally speaking, the lexical gap in the context of an ontology can be described as a mismatch between a query term and the label of the corresponding ontology entity.

There are three existing approaches to solve the lexical gap:

(1.4) 1. Labels: Labels are often available in multiple languages (for example in DBpedia), but do not provide linguistic information, e.g. which part of speech (noun, verb, etc.) it represents. Relying on labels also does not produce the needed lexical variants.

2. Lexical Resources: Resources such as Wiktionary[1] or WordNet (Miller (1995)) provide a huge amount of lexical variants, as well as rich linguistic informations, but do not provide anchors to specific ontologies.

3. Ontology Lexica (Prévot et al. (2010)): Specify verbalization variants of ontology elements in a particular language. They provide not only different lexicalizations together with linguistic informations, but also the anchors to a specific knowledge base.

The creation of a lexicon for a given ontology is called *Ontology Lexicalization*. Traditionally, such ontology lexica are manually created by domain experts and speakers of the target language, leading to a huge effort to create lexica for multiple languages and domains. When updating the ontology or switching to another ontology, these ontology lexica need to be manually changed and updated.

There are certain requirements an ontology lexicon has to fulfill in order to bridge the lexical gap:

(1.5)
1. The lexicon should contain lexical variants and it has to be enriched with linguistic information, such as the part-of-speech, the syntactic frame, and the mapping between semantic and syntactic arguments.
2. The lexicon needs to be generated with as little manual effort as possible, e.g. is created automatically or with the help of crowd sourcing.
3. If possible it should incorporate already existing resources, or link to them.
4. It should serve different purposes. Further, it is important that the lexicon supports multilinguality and connects information between the different languages.

To solve the task of *Ontology Lexicalization* in a (semi-) automatic fashion, which we will formally introduce in the next section, we propose a framework called M-ATOLL (Multilingual, Automatic inducTion of OntoLogy Lexica), which produces multilingual lexica for example in order to support QA systems in addressing the lexical gap. The goal of this system is to learn as many instances as possible of the lexicalization relation that can potentially solve or at least ameliorate the lexical gap problem as it amounts essentially to learning different natural language variants of expressing some ontological entity.

M-ATOLL is a combination of three different approaches:

1. Label-based approach
2. Corpus-based approach
3. Machine-learning based approach to deal with adjectives as lexicalization for restriction classes

The *label-based approach* works both for classes and properties from a given ontology. It uses label information from the ontology for given classes and properties in order to extract correct

[1]See `https://www.wiktionary.org`

synsets from WordNet to create a list of lexical variants. This allows the challenge, illustrated in Example 1.1 on page 2, to be solved.

The *corpus-based approach* is the main approach of M-ATOLL. It uses data from a corresponding knowledge base in order to extract relevant sentences from a text corpus that are likely to express a given RDF triple $< s, p, o >$ for some given property p. Thus, this approach works only for ontology properties. The extracted sentences are dependency-parsed and converted to RDF. On this RDF data, handcrafted grammatical patterns are applied in order to extract lexical variants. A *grammatical pattern*, in the following sometimes only called *pattern*, matches a relation between two entities, as for example in the sentence `Barack Obama is married to Michelle Obama`. For the relation presented in this sentence, we have a pattern that extracts the relation `married to`. After applying a set of different handcrafted grammatical patterns on a bulk of parsed sentences, filtering strategies are used to add only valid lexicalizations to the final lexicon. A filtering strategy for example consists in adding only those relations to the final lexicon that were extracted with a minimum frequency of two.

With this approach, M-ATOLL is independent of the property labels, as we can create lexical entries also for relations with unusual or even missing names. For example, when dealing with a relation name such as *relation_1524*, with a similar label, it is almost impossible to create valid lexicalizations from the label alone. By exploiting a corpus based approach that searches for potential occurrences of the given relation, M-ATOLL creates lexicalizations in cases difficult even for humans. Hence, this approach can handle cases such as presented in Example 1.1 on page 2 and Example 1.2 on page 3.

The *adjective lexicalization approach* extracts for each given property all RDF triples and keeps those triples which contain at least one adjective in the object. Then, a machine learning-based model is applied on these extractions in order to determine which are valid lexicalizations and should be added to the final lexicon. This approach is used to extract more complex adjective relations, such as presented in Example 1.3.

With M-ATOLL we combine all three approaches presented in Enumeration 1.4 within one single framework. Additionally, M-ATOLL fulfills all the requirements on ontology lexicalization mentioned in Enumeration 1.5.

M-ATOLL currently supports German, Spanish and English for the corpus-based approach and English for the two other approaches. However, the latter two are designed in a way that they can be ported to other languages as well. In oder to serialize our created lexica we use the *lemon* model, which is now under consideration to be standardized under the W3C[2].

We instantiated our approach using the DBpedia dataset and ontology. This is a very commonly used dataset and corresponding ontology. Creating lexicalizations for this ontology therefore has a high potential impact. DBpedia is also a very broad ontology containing knowledge from many different domains. This gives us the possibility to design and evaluate a framework which is not limited to one domain but general enough to be easily transferred to other ontologies.Thus, our framework is independent of DBpedia and will work with other ontologies with a corresponding knowledge base.

[2]See http://www.w3.org/2016/05/ontolex/

1.2 Task definition

To illustrate the challenge that M-ATOLL addresses, consider the natural language question in Figure 1.2, given in English, German and Spanish together with a SPARQL query that retrieves the answers from DBpedia. In this example, the expressions `cities`, `Städte` and `ciudades` refer to the DBpedia class *dbo:City*. This mapping is straightforward in English, and could equally easily be established for German and Spanish by means of simple machine translation or a lookup in a multilingual lexicon. However, harder to automatically establish is the mapping of the expressions `have inhabitants`, `haben Einwohner` and `tienen habitantes` to the DBpedia property *dbo:populationTotal*, even when using existing lexical resources such as WordNet[3] or Wiktionary[4].

```
Which cities have more than 250 000 inhabitants?
Welche Städte haben mehr als 250 000 Einwohner?
¿Qué ciudades tienen más de 250 000 habitantes?
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?uri
WHERE {
  ?uri a dbo:City .
  ?uri dbo:populationTotal ?p .
  FILTER ( ?p > 250000 )
}
```

Figure 1.2: Example question in English, German and Spanish, together with a SPARQL query that retrieves the answers from DBpedia.

```
How tall is Michael Jordan?
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?size
WHERE {
  dbr:Michael_Jordan dbo:height ?size .
}
```

Figure 1.3: Example question in English `How tall is Michael Jordan` together with the corresponding SPARQL query

```
How tall is Mount Everest?
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?height
WHERE {
  dbr:Mount_Everest dbo:elevation ?height .
}
```

Figure 1.4: Example question in English `How tall is Mount Everest` together with the corresponding SPARQL query

[3]See https://wordnet.princeton.edu
[4]See https://en.wiktionary.org

Often, there are different lexical variants to refer to the same ontology element. For example, the adjective `tall` can refer to the property *dbo:height* in a context like `How tall is Michael Jordan` (see Figure 1.3), and it can refer to the property *dbo:elevation* in a context like `How tall is Mount Everest` (see Figure 1.4). This is usually not captured by ontology labels as the context-specific meaning is missing. For example, DBpedia contains only one English label for *dbo:elevation* (namely `elevation`) and *dbo:height* (namely `height`).

In this thesis we present an approach to extracting such mappings between natural language expressions and ontology elements automatically that can easily be ported across ontologies and languages. This task can be formally defined as follows:

> **Task: Ontology lexicalization**
>
> Given an ontology, a knowledge base and a text corpus, return all possible and meaningful lexicalizations for properties and classes of the given ontology.

For the purpose of this thesis we instantiate our approach for the DBpedia ontology. Since DBpedia covers a wide range of domains, we therefore avoid tailoring the approach towards a specific domain.

1.3 Research Questions

We will show that our approach satisfies the requirements presented in the previous section:

- The lexicon should contain lexical variants and it has to be enriched with linguistic information, such as the part-of-speech, the syntactic frame, and the mapping between semantic and syntactic arguments.
- The lexicon needs to be generated with as little manual effort as possible, e.g. is created automatically or with the help of crowd sourcing.
- If possible it should incorporate already existing resources, or link to them.
- It should serve different purposes. Further, it is important that the lexicon supports multilinguality and connects information between the different languages.

Additionally we want our system to outperform other automatic approaches (represented by a baseline described later in this thesis) with respect to F-measure.

This leads to the following research questions:

1. Do the results of the two main approaches, the label-based and corpus-based approach, complement each other?
2. How do different filtering strategies affect the quality of the created lexicon in terms of precision, recall and F-measure?
3. What are the advantages and disadvantages of M-ATOLL, compared to other state-of-the-art systems?
4. What is the coverage of our hand-crafted patterns with respect to a lexical evaluation over a gold standard? Are there frequent patterns not covered by the ones defined by us?

1.4 Contributions

Our main contribution to the scientific community is to present the first open source framework for the automatic extraction of multilingual ontology lexica. This approach relies on a combination of hand-crafted patterns with a state-of-the art machine learning approach, additionally incorporating knowledge from well known existing lexical resources, such as WordNet. The whole system is built to support multilinguality and uses exchangeable and adaptable patterns. In detail our contributions are as follows:

1. We develop the first open-source multilingual approach for the generation of ontology lexica.
2. We present grammatical patterns for three different languages for extracting lexicalizations, which are reusable beyond our framework. We provide an analysis of these patterns as well as a comparison with those proposed by others.

3. We give a detailed evaluation of different filtering strategies. These strategies are relevant to determine whether an extracted lexicalization should be added to the final lexicon or not. We evaluate four different strategies:

   (a) Frequency-based filtering strategy

   (b) Label-based filtering strategy

   (c) Property-based filtering strategy

   (d) Machine-learning-based filtering strategy

   For all strategies, we evaluate their impact in terms of precision, recall and F-measure and discuss the positive and negative aspects of each strategy. For the machine learning-based filtering strategy we present and evaluate each feature, all possible combinations of each feature as well as how the best feature combination behaves on a strict train/test split with previously unseen test data.

4. We present a novel machine learning based approach to extend M-ATOLL to adjective lexicalizations for restriction classes. We present each feature with examples and give a detailed analysis of each feature used, as well as the combination of different features.

5. We publish an open source implementation with extendable and adaptable patterns and an API for *lemon* lexica. The source code is available at:

   `https://github.com/ag-sc/matoll`

6. All resulting lexica are fully available at:

   `http://dblexipedia.org`

1.5 Publications

Parts of this thesis have been published earlier in the following papers:

2016 – S. Walter, C. Unger, P. Cimiano: *Automatic acquisition of adjective lexicalizations of restriction classes: a machine learning approach.* Journal on Data Semantics

2015 – S. Walter, C. Unger, P. Cimiano: *DBlexipedia: A nucleus for a multilingual lexical Semantic Web.* Proceedings of 3th International Workshop on NLP and DBpedia, co-located with the 14th International Semantic Web Conference (ISWC 2015)

2014 – S. Walter, C. Unger, P. Cimiano, B. Lanser: *Automatic acquisition of adjective lexicalizations of restriction classes.* Proceedings of 2nd International Workshop on NLP and DBpedia, co-located with the 13th International Semantic Web Conference (ISWC 2014)

– S. Walter, C. Unger, P. Cimiano: *M-ATOLL: A Framework for the Lexicalization of Ontologies in Multiple Languages.* Proceedings of the Semantic Web: 13th International Semantic Web Conference (ISWC 2014)

– S. Walter, C. Unger, P. Cimiano: *ATOLL - A framework for the automatic induction of ontology lexica.* Data and Knowledge Engineering, Volume 94, pp.148-162.

2013 – S. Walter, C. Unger, P. Cimiano: *A corpus-based approach for the induction of ontology lexica.* Proceedings of the 18th International Conference on Application of Natural Language to Information Systems (NLDB2013)

2012 – S. Walter, C. Unger, P. Cimiano, D. Bär: *Evaluation of a layered approach to question answering over linked data.* Proceedings of the 11th International Semantic Web Conference (ISWC 2012)

Throughout this thesis I use the personal pronoun *we*, because all of the above mentioned papers stem from close cooperation with Prof. Dr. P. Cimiano and Dr. C. Unger, among others. However, in all of these papers I contributed the most in terms of writing, design, implementation, experiments and evaluation.

1.6 Outline

The outline of this thesis is as follows:

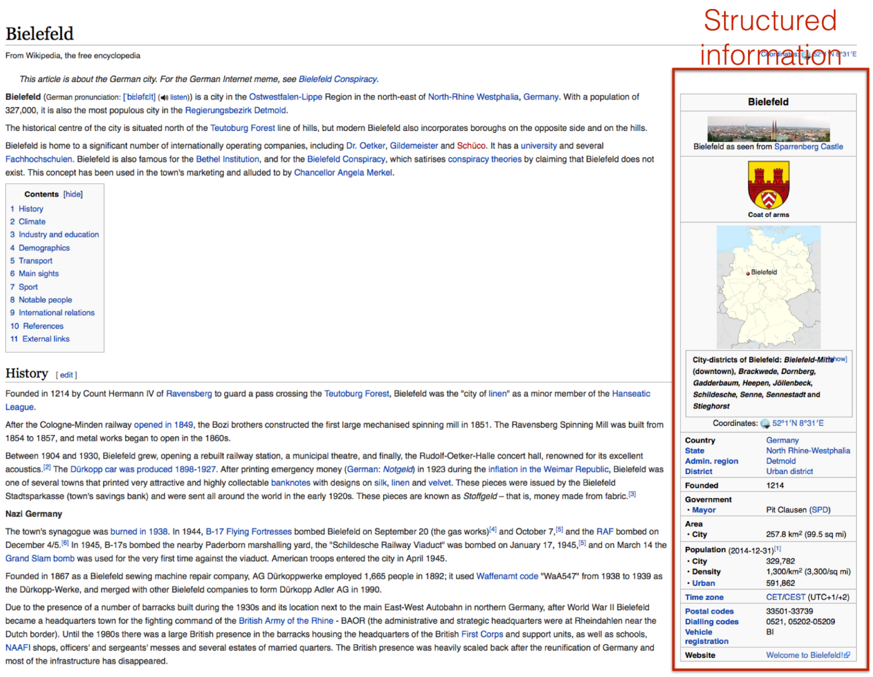
- Chapter 2 presents the foundations this thesis builds on. After presenting the concept of the Semantic Web and Linked Open Data in Sections 2.1.1 and 2.1.2, we present some constitutive ideas that build upon these concepts. In the second part of this chapter we introduce the task of ontology lexicalization (Section 2.2.1) and *lemon* (Section 2.2.2) in more detail..
- Chapter 3 contains an overview of the related work relevant for M-ATOLL.
- Chapter 4 presents the main approaches of our framework, namely the label-based as well as the corpus-based approach. This chapter concludes with a detailed evaluation of the results.
- Chapter 5 shows that the results can be improved with different filtering strategies. It also evaluates the impact of the different hand-crafted patterns and presents more details on the input corpus. We then present a quantitive analysis of the results of state-of-the-art systems, represented by our baseline, compared with the results from M-ATOLL
- Chapter 6 highlights that M-ATOLL, as introduced, does not cover a very important part of the lexicalization part, namely adjectives with restriction classes. Therefore in this chapter we present a machine learning based approach to tackle this problem and present a preliminary evaluation and discussion.
- Chapter 7 introduces *DBlexipedia*, where the resulting lemon lexica are made available for public.
- Chapter 8 closes this thesis with ideas for further work as well as a summary of the thesis.

In the appendix (see Section A.4), we give a brief overview of the implementation of M-ATOLL, as well as a description of how to actually start and use it.

Chapter 2

Foundations

In this chapter we introduce the foundations which this thesis builds on.



Bielefeld

From Wikipedia, the free encyclopedia

This article is about the German city. For the German Internet meme, see Bielefeld Conspiracy.

Bielefeld (German pronunciation: [ˈbiːləfɛlt] (listen)) is a city in the Ostwestfalen-Lippe Region in the north-east of North-Rhine Westphalia, Germany. With a population of 327,000, it is also the most populous city in the Regierungsbezirk Detmold.

The historical centre of the city is situated north of the Teutoburg Forest line of hills, but modern Bielefeld also incorporates boroughs on the opposite side and on the hills.

Bielefeld is home to a significant number of internationally operating companies, including Dr. Oetker, Gildemeister and Schüco. It has a university and several Fachhochschulen. Bielefeld is also famous for the Bethel Institution, and for the Bielefeld Conspiracy, which satirises conspiracy theories by claiming that Bielefeld does not exist. This concept has been used in the town's marketing and alluded to by Chancellor Angela Merkel.

Contents [hide]

History [edit]

Founded in 1214 by Count Hermann IV of Ravensberg to guard a pass crossing the Teutoburg Forest, Bielefeld was the "city of linen" as a minor member of the Hanseatic League.

After the Cologne-Minden railway opened in 1849, the Bozi brothers constructed the first large mechanised spinning mill in 1851. The Ravensberg Spinning Mill was built from 1854 to 1857, and metal works began to open in the 1860s.

Between 1904 and 1930, Bielefeld grew, opening a rebuilt railway station, a municipal theatre, and finally, the Rudolf-Oetker-Halle concert hall, renowned for its excellent acoustics.[2] The Dürkopp car was produced 1898-1927. After printing emergency money (German: *Notgeld*) in 1923 during the inflation in the Weimar Republic, Bielefeld was one of several towns that printed very attractive and highly collectable banknotes with designs on silk, linen and velvet. These pieces were issued by the Bielefeld Stadtsparkasse (town's savings bank) and were sent all around the world in the early 1920s. These pieces are known as *Stoffgeld* – that is, money made from fabric.[3]

Nazi Germany

The town's synagogue was burned in 1938. In 1944, B-17 Flying Fortresses bombed Bielefeld on September 20 (the gas works)[4] and October 7,[5] and the RAF bombed on December 4/5.[6] In 1945, B-17s bombed the nearby Paderborn marshalling yard, the "Schildesche Railway Viaduct" was bombed on January 17, 1945,[5] and on March 14 the Grand Slam bomb was used for the very first time against the viaduct. American troops entered the city in April 1945.

Founded in 1867 as a Bielefeld sewing machine repair company, AG Dürkoppwerke employed 1,665 people in 1892; it used Waffenamt code "WaA547" from 1938 to 1939 as the Dürkopp-Werke, and merged with other Bielefeld companies to form Dürkopp Adler AG in 1990.

Due to the presence of a number of barracks built during the 1930s and its location next to the main East-West Autobahn in northern Germany, after World War II Bielefeld became a headquarters town for the fighting command of the British Army of the Rhine - BAOR (the administrative and strategic headquarters were at Rheindahlen near the Dutch border). Until the 1980s there was a large British presence in the barracks housing the headquarters of the British First Corps and support units, as well as schools, NAAFI shops, officers' and sergeants' messes and several estates of married quarters. The British presence was heavily scaled back after the reunification of Germany and most of the infrastructure has disappeared.

Coordinates: 52°1′N 8°31′E

Bielefeld

Bielefeld as seen from Sparrenberg Castle

Coat of arms

Bielefeld

City-districts of Bielefeld: ***Bielefeld-Mitte*** (downtown), ***Brackwede***, ***Dornberg***, ***Gadderbaum***, ***Heepen***, ***Jöllenbeck***, ***Schildesche***, ***Senne***, ***Sennestadt*** and ***Stieghorst*** [show]

Coordinates: 52°1′N 8°31′E

Country	Germany
State	North Rhine-Westphalia
Admin. region	Detmold
District	Urban district
Founded	1214
Government	
• **Mayor**	Pit Clausen (SPD)
Area	
• **City**	257.8 km^2 (99.5 sq mi)
Population (2014-12-31)[1]	
• **City**	329,782
• **Density**	1,300/km^2 (3,300/sq mi)
• **Urban**	591,862
Time zone	CET/CEST (UTC+1/+2)
Postal codes	33501-33739
Dialling codes	0521, 05202-05209
Vehicle registration	BI
Website	Welcome to Bielefeld!

Figure 2.1: Excerpt from the Wikipedia article Bielefeld (https://en.wikipedia.org/w/index.php?title=Bielefeld&oldid=727115787) to show the loss of information in unstructured data

2.1 Semantic Web

In this section, we describe some fundamental techniques used in this work. All of these techniques were designed upon the idea of the Semantic Web.

2.1.1 Semantic Web

Berners-Lee et al. (2001) introduce the Semantic Web as an extension of the current web, where information is mapped to well-defined meaning in order to increase the cooperation between people and machines. In the `Semantic Web` the term `Semantic` emphasizes the fact that each used symbol has a counterpart in some model of the world. The goal of the `Semantic Web` is to move away from a web created for humans to a web designed to be used by machines. This means that all information has to be stored in some giant (distributed) database, so machines are able to access and use this data. However in reality, most information on the current Web are not easily interpretable and understandable for machines. For example in Wikipedia most information is stored in natural language text, while only the most important key facts are stored in a structured form in the so called info box of each article. This is displayed in Figure 2.1, where on the right side the structured data is available, while on the left side of the figure natural language text, which is traditionally not easily interpretable by machines, is presented.

The goal to extract structured information from text led to an own research field called Information Extraction, which will be discussed in Chapter 3.

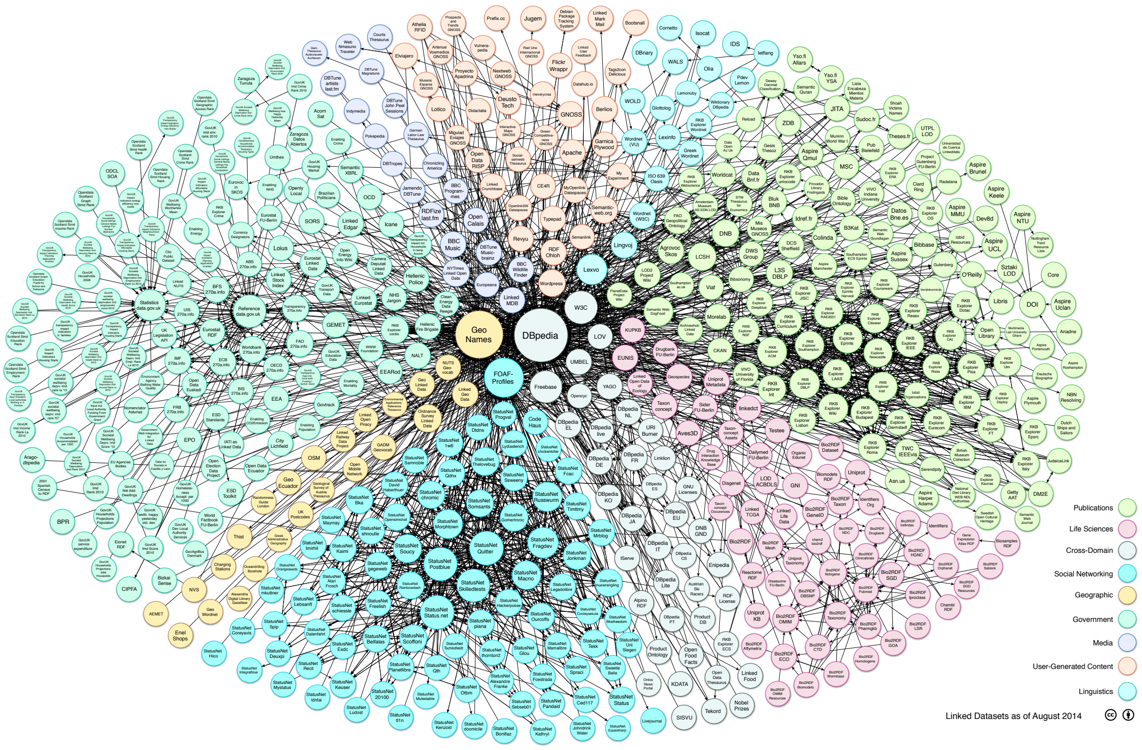


Figure 2.2: Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

2.1.2 Linked Open Data

Implementing the idea of the Semantic Web in a traditional, centralized way, would lead to a giant and unfeasible database, therefore a distributed architecture, where everyone can contribute, was chosen. The architecture which was selected to do so is Linked Data, which enables everyone to contribute to the Semantic Web by linking an own dataset to other datasets. As more and more datasets get publicly available, Linked Data is also often called by the term Linked Open Data, or LOD. The central idea behind LOD is that every resource in this network has a unique Uniformed Resource Identifier (URI) to which other resources are linked.

The concept of LOD was first introduced in Berners-Lee (2006) and since then more and more datasets, such as DBpedia, have adopted the so called Linked Data principles. Tim Berners-Lee proposed four Linked Data Principles:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful RDF information
4. Include RDF statements that link to other URIs so that they can discover related things

```
(http://www.uni-bielefeld.de/resource,
http://example.org/isLocatedIn,
http://bielefeld.de/resource)
```

Figure 2.3: Example for a triple with a resource as object

```
(http://www.uni-bielefeld.de/resource,
http://example.org/isLocatedIn,
'Bielefeld')
```

Figure 2.4: Example for a triple with a literal as object

The status of the so called LOD cloud, from the year 2014, is displayed in Figure 2.2. Newer visualizations are not available due to the fast increase of the LOD cloud and would therefore lead to even more dense pictures. The Figure visualizes different datasets and the connection between them with Datasets like DBpedia, Geo Names[1] and Foaf by Graves et al. (2007) as the center at the cloud. DBpedia, as presented in the work by Auer et al. (2007) and Kobilarov et al. (2009), describes itself as "A Nucleus for a Web of Open Data". It has a strong community which extract structured informations from Wikipedia articles, mainly from the info box of each article. It supports more than 100 languages and provides the ontology which is used in this thesis.

2.1.3 Resource Description Framework

A data model is necessary to incorporate data as linked open data. For this the Resource Description Framework, also called RDF, was developed. RDF is a data model which allows resources to be described on the web in a structured form. It is also a W3C standard[2] and was first presented in 2004. RDF supports the exchange of data on the web and describes information in a structured and query-able way. Resources, such as `Bielefeld University`, are identified with Uniformed Resource Identifiers (URI). The URI for the resource `Bielefeld University` for example is `http://www.uni-bielefeld.de/resource`. Some URIs might resolve in the browser, similar to an URL of a website, others might not. However as the URI is an identifier it does not necessarily need to be resolvable. The RDF data model builds on so called `triples` of the form *(subject, predicate, object)*, where the `subject` and `predicate` are resources, as described above. The `object` can either be a resource (see Figure 2.3), or a literal (see Figure 2.4), such as `Bielefeld`. On the position of the subject and the object, it is also possible to place blank nodes, which are not closer specified `items` without an URI. In this work we do not use blank nodes, as our goal is to have everything fully resolvable, in order to fulfill the LOD principle.

When combining different triples, they essentially represent a graph, where the edges represent the properties and the nodes of the graph are represented by the objects. An example is given in

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

[2] See `https://www.w3.org/RDF/`

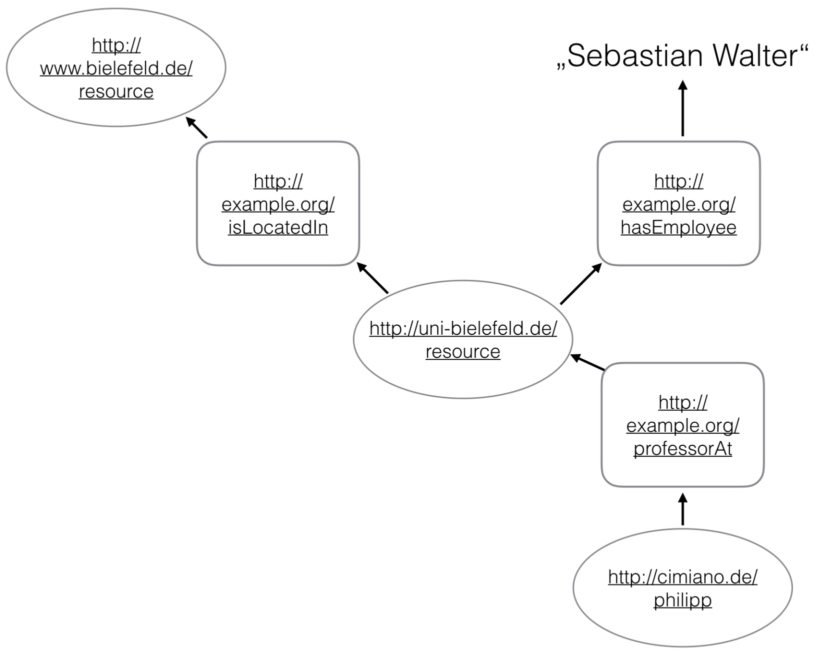

http://www.bielefeld.de/resource
„Sebastian Walter“
http://example.org/isLocatedIn
http://example.org/hasEmployee
http://uni-bielefeld.de/resource
http://example.org/professorAt
http://cimiano.de/philipp

Figure 2.5: RDF Graph example.

Figure 2.5.

When not presented as a visualized graph, RDF is serialized using the `N3`, `Turtle` or `RDF/XML` syntax. All three are presented in the following using the example from Figure 2.5:

1. Turtle:

```
<http://www.uni-bielefeld.de/resource> <http://example.org/isLocatedIn> <http://www.bielefeld.de/resource>.
<http://www.uni-bielefeld.de/resource> <http://example.org/hasEmployee> "Sebastian Walter".
<http://cimiano.org/philipp> <http://example.org/professorAt> <http://www.uni-bielefeld.de/resource>.
```

2. Abbreviation of Turtle syntax:

```
@prefix cimiano: <http://www.cimiano.org/>
@prefix unibi: <http://www.uni-bielefeld.de/>
@prefix biele: <http://www.bielefeld.de/>
@prefix example: <http://example.org/>
cimiano:philipp example:professorAt unibi:resource.
unibi:resource example:isLocatedIn biele:resource;
 example:hasEmployee "Sebastian Walter".
```

3. XML:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/199/02/22-rdf-syntax-ns#"
          xmlns:ex="http://example.org#"
<rdf:Description rdf:about="http://www.cimiano.org/philipp">
<ex:professorAt>
<rdf:Description rdf:about="http://www.uni-bielefeld.de/resource">
```

```
            </rdf:Description>
        </ex:professorAt>
    </rdf:Description>

    <rdf:Description rdf:about="http://www.uni-bielefeld.de/resource">
    <ex:isLocatedIn>
    <rdf:Description rdf:about="http://www.bielefeld.de/resource">
    <ex:name> Bielefeld</ex:name>
            </rdf:Description>
        </ex:sLocatedIn>
    <ex:hasEmployee>
    <rdf:Description>
    <ex:name> Sebastian Walter</ex:name>
            </rdf:Description>
        </ex:hasEmployee>
    </rdf:Description>
    </rdf:RDF>
```

All three examples represent the same information. In natural language this information could be read as follows: `The University of Bielefeld is located in Bielefeld. The University has an employee named Sebastian Walter. Philipp Cimiano is a Professor at Bielefeld University`.

It is also possible to use a literal on the position of an object instead of resources. E.g. consider the following example in turtle syntax:

```
@prefix cimiano: <http://www.cimiano.org/>
@prefix example: <http://example.org/>
cimiano:philipp example:hasName "Philipp Cimiano".
```

The example presents a resource `http://cimiano.org#philipp` which has a property *hasName* with a string representation of the Name `Philipp Cimiano`.

In order to formalize the semantics behind RDF, for example to argue when a triple is true, we inherited the formalization introduced by Hitzler et al. (2007). As described before, triples are used to describe the relation between resources connected by a property. The interpretation of this relation consists of two different sets, namely IR, representing abstract resources and IP, representing abstract properties. We also need a function I_{EXT}, which describes how a specific pair of resources are connected with one specific property. This function shows that subject and object resources are not URIs, but representations of URIs or literals.

Therefore consider the following definitions: A simple interpretation $\mathcal{I}$ of a given RDF vocabulary $\mathcal{V}$ consists of:

- IR: a non-empty set of resources, also called the domain of discourse of $\mathcal{I}$
- IP: a set of properties. This might overlap with IR
- I_{EXT}: a function assigning each property a set of pairs from IR: $I_{EXT} : IP \rightarrow 2^{IR \times IR}$
- I_S: a function mapping URIs from $\mathcal{V}$ into the union of IR and IP, i.e. $I_S : \mathcal{V} \rightarrow IR \cup IP$

- I_L: a function from the typed literals from $\mathcal{V}$ into the set of IR resources
- LV: a particular subset of IR, called the set of literal values containing (at least) all untyped literals from $\mathcal{V}$.

After defining the sets IR, LP and LV we are now defining a function $\cdot^{\mathcal{I}}$, which maps all literals and URIs from the set $\mathcal{V}$ to the set of all possible resources and properties:

- every untyped literal "a" is mapped to a, or more formally: $(\text{“a”})^{\mathcal{I}} = a$
- every untyped literal carrying language information "a"@t is mapped to the pair $\langle a, t\rangle$, $i.e.$$(\text{“a”@t})^{\mathcal{I}} = \langle a, t\rangle$
- every typed literal is mapped to $I_L(l)$, formally $I^{\mathcal{I}} = I_L(l)$
- every URI u is mapped to $I_S(u)$, i.e. $u^{\mathcal{I}} = I_S(u)$

This function has to be interpreted as follows: All untyped literals are mapped to themselves, while typed literals are not bound to any semantic constraints. An untyped literal is something like a string or number, without any further information, while a typed literal has an additional information, such that it represents a date or a person etc. With these definitions we can now show if a given triple $\langle s, p, o\rangle$ is true or not. Therefore the function $\langle s, p, o\rangle \cdot^{\mathcal{I}}$, of a triple $\langle s, p, o\rangle$, is true if the following condition is true:

$$\langle s, p, o\rangle^{\mathcal{I}} = \begin{cases} true, & \text{if } \langle s^{\mathcal{I}}, o^{\mathcal{I}}\rangle \in I_{EXT}(p^{\mathcal{I}}) \\ false, & \text{otherwise} \end{cases}$$

Following this definition, a RDF-Graph is true if and only if all triples in this graph are true. Note that these definitions do not hold if the object of the given triple $\langle s, p, o\rangle$ is a blank node. However, for this work we are ignoring blank nodes.

RDF supports two concepts, which are needed for the following sections, namely the concept of *rdf:type*[3] and *rdf:Property*[4], where *rdf:* stands for the namespace http://www.w3.org/1999/02/22-rdf-syntax-ns#.

A namespace is a reserved attribute the goal of which is to shorten an URI and make it easily reusable.

rdf:Property describes the concept of an RDF property as a relation between subject resources and object resources. *rdf:type* is used to state that a resource is an instance of a certain type. For the previous example this means that the property *example:hasName* can be defined as follows:

```
example:hasName rdf:type rdf:Property .
```

This triple states that the property *example:hasName* is a property and can therefore be used in a triple $\langle s, p, o\rangle$ on the p position, as done above.

[3] http://www.w3.org/TR/rdf-schema/#ch_type
[4] http://www.w3.org/TR/rdf-schema/#ch_property

2.1.4 Ontologies

An extension of RDF is RDF schema, also called RDF(s). RDF(s) is a lightweight ontology language and adds some additional vocabulary to RDF, for example *rdfs:Class*, which defines a set of resources representing entities from the real world.

Consider the example of a class *example:Fruit*, containing apples and bananas as entities among others.

(2.1)
```
example:Fruit rdf:type rdfs:Class .
```

In this statement we defined that *example:Fruit* is of the type *rdfs:Class*. To define the type of a resource, we use the property *rdf:type*. This means that we now can add other resources to this class in order to combine different resources to a set of resources under the name *example:Fruit*.

Until now the class *example:Fruit* has no elements. We change this with the following statement:

(2.2)
```
resource:Apple rdf:type example:Fruit .
resource:Banana rdf:type example:Fruit .
```

With this statement we express the relation between the resources *resource:Apple* and *resource:Banana* and the class *example:Fruit*. The relation is now the following: Both resources are now elements of the class *example:Fruit* and can be retrieved as such.

With RDF(s) we are also able to define subclasses with the term *rdfs:subClassOf*. For example we could use our previously introduced class *example:Fruit* to express that this is a subclass of another class, for example *example:Plants*. This would look as follows:

(2.3)
```
example:Plant rdf:type rdfs:Class .
example:Fruit rdfs:subclassOf example:Plant .
```

In the first triple we defined *example:Plant* to be a class, in the second triple we then defined that our class *example:Fruit* is a subclass of *example:Plant*. This gives us now the possibility to infer that our resources *resource:Apple* and *resource:Banana* are not only *example:Fruits*, but also *example:Plants*.

With this we now have introduced a small and simple ontology. An ontology is a formal specification of a conceptualization in the sense of Gruber (1995).

However, even if for the purpose of this thesis the presented RDF/RDF(s) concepts are enough, and we do not use any reasoning on the input ontology, we would like to mention that there is a more powerful language to design ontologies. This language is called the Web Ontology Language (OWL), which was standardized by the W3C in 2004. The goal of OWL was to create a language, which on the one hand is very expressive, but on the other hand allows fast and efficient reasoning. There are three different types of OWL, namely OWL Full, OWL DL and OWL Lite, each with a different level of expressiveness for different purposes. The relation between the three languages and RDFS is visualized in Figure 2.6.

In this work we do not distinguish between the three languages in more detail. However for the sake of completeness, we are presenting shortly the advantages and disadvantages of these

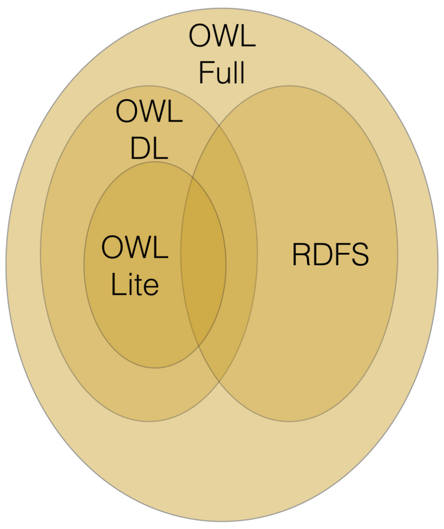


Figure 2.6: Relation between OWL Full, OWL DL, OWL Lite and RDFS

book	title	author	price
SW_Foundations	"Semantic Web Grundlagen"	Hitzler	27.99
Python_Machine_Learning	"Python Machine Learning"	Raschka	43.8
Avogadro_Corp	"Avogadro Corp: The Singularity Is Closer Than It Appears"	Hertling	9.62
Sherlock_Holmes	"Sherlock Holmes: The Complete Novels and Stories"	Doyle	0.99

Table 2.1: Example dataset, contains books and the most relevant information about them.

languages. *OWL Full* contains not only OWL DL and OWL Lite, but as the only one of the three also contains RDFS. It is very powerful, but it is not decidable. *OWL DL* contains OWL Lite and is a subset of OWL Full, in comparison to the former one, it is decidable and is supported by almost all software tools. The complexity is NExpTime in the worst case. *OWL Lite* is a subset of OWL DL and OWL Full, and is fully decidable, but not as powerful as the other two. Due to this fact the worst case complexity is only ExpTime.

An ontology based on OWL, also called OWL-Ontology, consists mainly of classes and properties, these can be used to create complex relationships between each other.

2.1.5 SPARQL

SPARQL is the abbreviation for *SPARQL Protocol And RDF Query Language*[5] and is a W3C recommendation since 2008 with an update to SPARQL 1.1 in 2013. In this section we explain some of the basic functions of SPARQL. We use SPARQL to serialize our patterns which are used in M-ATOLL by the corpus-based approach.

Imagine a database with four book entries as presented in Table 2.1. All four books have an URI, a title (as literal), an author(represented as URI) and a price. We assume that *example:*

[5] http://www.w3.org/TR/sparql11-overview/

subject	predicate	object
example:SW_Foundations	*example:title*	`"Semantic Web Grundlagen"`
example:SW_Foundations	*example:author*	*example:Hitzler*
example:SW_Foundations	*example:price*	`27.99`
example:Python_Machine_Learning	*example:title*	`"Python Machine Learning"`
example:Python_Machine_Learning	*example:author*	*example:Raschka*
example:Python_Machine_Learning	*example:price*	`43.8`
example:Avogadro_Corp	*example:title*	`"Avogadro Corp: The Singularity Is Closer Than It Appears"`
example:Avogadro_Corp	*example:author*	*example:Hertling*
example:Avogadro_Corp	*example:price*	`9.62`
example:Sherlock_Holmes	*example:title*	`"Sherlock Holmes: The Complete Novels and Stories"`
example:Sherlock_Holmes	*example:author*	*example:Doyle*
example:Sherlock_Holmes	*example:price*	`0.99`

Table 2.2: Example dataset about books. Equivalent to Table 2.1 on the previous page, but in triple format

stands for `http://example.org/` and the price is given as a float value, with other words, the literals in this column are from the type `http://www.w3.org/2001/XMLSchema#float`. Presented as a triple structure, this data would look as presented in Table 2.2. In the following we present some questions in natural language, as well as the corresponding SPARQL query.

(2.4) Give me all authors.

```
PREFIX example: <http://example.org/>
SELECT DISTINCT ?author WHERE {
    ?x example:author ?author .
}
```

(2.5) Give me all authors who wrote the book “Semantic Web Grundlagen”

```
PREFIX example: <http://example.org/>
SELECT DISTINCT ?author WHERE {
    ?x example:title "Semantic Web Grundlagen".
    ?x example:author ?author .
}
```

(2.6) Did the author Hertling write the book “Sherlock Holmes: The Complete Novels and Stories”?

```
PREFIX example: <http://example.org/>
ASK {
    ?x example:author example:Hertling .
    ?x example:title
    "Sherlock Holmes: The Complete Novels and Stories".
}
```

(2.7) Give me all book titles which cost more that 10.

```
PREFIX example: <http://example.org/>
SELECT ?title WHERE {
    ?x example:title ?title.
    ?x example:price ?price.
    FILTER (?price > 10)
}
```

In Example 2.4 on the facing page we ask for all authors presented in Table 2.1 on page 21. This is done with a simple *SELECT* query in SPARQL considering only the property *example:author*. Every object containing this specific property is returned.

In Example 2.5 on the facing page the question is a little bit more complex than the previous one, as it is asked for all authors who wrote the book `"Semantic Web Grundlagen"`. For this example we use again a *SELECT* query, this time with two triples. In the first triple, we select all items in the triple, which have the title `"Semantic Web Grundlagen"`. In the second triple, similar to the previous example, all triples containing the relation *example:author* are retrieved. The results of both triples are intersected on the position of the variable *?x*, which then yields only the (correct) answer *example:Hitzler*.

In Example 2.6, we show how to create queries returning only a boolean value (`true` or `false`) as answer. Instead of the *SELECT* as in the previous two examples, we use the *ASK* construct. In this case SPARQL checks if a given triple is true (how this is done see Section 2.1.3). In our example we have two given triples, the first returning all books by the author *example:Hertling* and the second returning the book with the title `"Sherlock Holmes: The Complete Novels and Stories`. If the intersected results (again intersected on the variable *?x*) lead to an answer, the query returns *true*, otherwise *false*.

In the final example (see Example 2.7) we use a *FILTER* operator in order to find all books with a price greater than 10. SPARQL supports comparison operators such as $<, =, >, <=, >=, ! =$, the comparison can be done on literals w.r.t. their natural order. It also supports numerical datatypes such as *xsd:int*, *xsd:dateTime* etc. For other types only the $=$ and $! =$ operators are defined and it is not possible to compare different types, e.g. it is not possible to compare strings with integers.

SPARQL also supports the possibility of optional triples within the query [6], as well as the possibility to join the results of two triples with the *UNION* operator[7], as well as many other operators.

An example for the usage of the *UNION* operator within a SPARQL query is given in Example 2.8 on the next page.

[6]See http://www.w3.org/TR/sparql11-query/#optionals
[7]See http://www.w3.org/TR/sparql11-query/#alternatives

(2.8) Give me all book titles by the authors Raschka and Hitzler

```
PREFIX example: <http://example.org/>
SELECT ?title WHERE {
    ?x example:title ?title .
    {?x example:author example:Hertling .}
    UNION
    {?x example:author example:Raschka .}
}
```

2.2 Natural Language Processing Methods

Natural Language Processing (NLP) studies the computer-supported processing and understanding of language. It is considered to be a subfield of Artificial Intelligence (AI). There are many practical applications for products based on NLP, such as dialogue, voice or assistance systems among others (see Martin and Jurafsky (2000)).

While the field of NLP is a wide research field, in this section we only present some concepts relevant to this thesis.

We first introduce the notion of *Lexicalization* and then give a small introduction into the main concepts of lemon, which we used to represent our final lexicon.

This section is closed with a small introduction of a dependency parser (MaltParser), as well as the CoNLL format.

2.2.1 Lexicalization

A *lexicalization* of an ontology element O (a class or property) in a language L comprises the following information:

- a lemma in L, the *canonical form*
- syntactic information, in particular:
  - the part of speech category (e.g. noun, verb, or adjective)
  - a subcategorization frame that specifies the required syntactic arguments
- semantic information, in particular:
  - a reference with respect to the ontology, i.e. O
  - a mapping of the corresponding semantic arguments to the syntactic arguments

As an example consider the DBpedia property *dbo:author*. Here are two possible English verbalizations:

- A lexicalization with the canonical form `write`. This lexicalization has part of speech *verb* and the subcategorization frame of a transitive verb taking two arguments: a subject a_1 and a direct object a_2. That is, the expression `write` occurs in a construction like a_1 `writes` a_2. It refers to the property *dbo:author*, which has two semantic arguments: the subject and object of a triple `<`a_2`,author,`a_1`>`. This already specifies the argument mapping: The syntactic subject corresponds to the object of the triple, and the syntactic object corresponds to the subject of the triple.

- A lexicalization with the canonical form `author`. This lexicalization has part of speech *noun* and the subcategorization frame of a relational noun taking two arguments: a copulative subject a_1 and a prepositional object a_2. That is, the expression `author` occurs in a construction like a_1 `is the author of` a_2. It refers to the property *dbo:author*, which has two semantic arguments: the subject and object of a triple `<`a_2`,author,`a_1`>`. As above, the argument mapping is such that the syntactic subject corresponds to the object of the triple, and the syntactic object corresponds to the subject of the triple.

Lexicalizations of classes are usually simpler. As an example consider the following two English lexicalizations of the DBpedia class *dbo:Star*:

- A lexicalization with the canonical form `star` and part of speech noun. It does not require any syntactic arguments. The semantic reference is the class *dbo:Star*.
- A lexicalization with the canonical form `sun` and part of speech noun. It also does not require any syntactic arguments and refers to the class *dbo:Star*.

We represent lexicalizations using *lemon* by McCrae et al. (2011), a framework for specifying machine-readable lexica in RDF, which allows to publish and share ontology lexicons as linked data. *Lemon* stays agnostic with respect to particular linguistic categories and formalisms, therefore parts of speech, syntactic frames and argument roles have to be imported from external linguistic vocabularies. M-ATOLL chooses this vocabulary to be *LexInfo* by Cimiano et al. (2011), as it is rich enough to define the different subcategorization frames we need. But nothing particular hinges on this choice; the lexicalizations could easily be adapted to any other ontology and could, in fact, be serialized in any other lexicon format as well.

2.2.2 Lemon

As mentioned earlier in this thesis, we use *lemon* to represent the extracted lexicalizations.

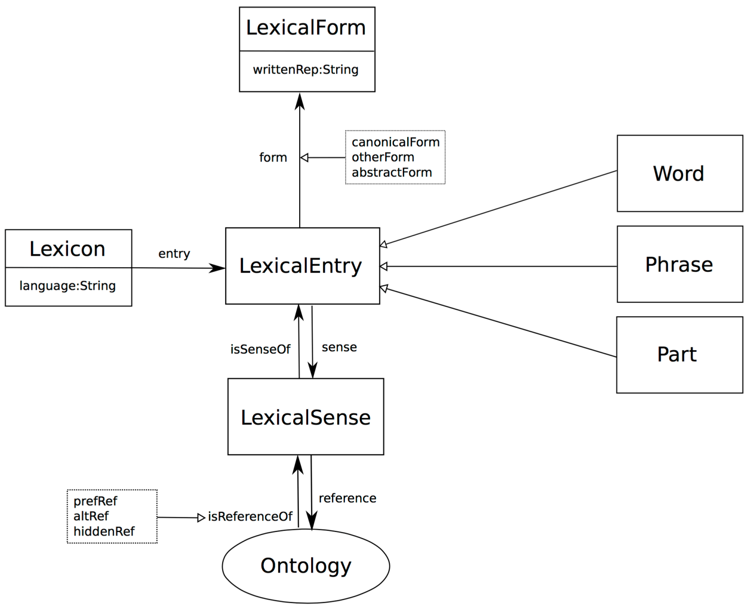


Figure 2.7: The lemon core as presented in `http://lemon-model.net/lemon-cookbook.pdf`

The *lemon core* is visualized in Figure 2.7 and consists of a set of relations and elements. The figure visualizes how the different elements and relations are connected to each other in order to define a lexical entry with all necessary core elements. The center of the Figure is the *LexicalEntry*, to whom all other items are connected to. On top of the Figure there is the *LexicalForm*, containing a written representation in form of a string. In case of the example above, this written form would be `'sun'` or `'star'`. On the bottom of the Figure each lexical entry is connected through a sense with the properties *isSenseOf* and *isReferenceOf* to a given ontology. In this thesis we implemented the lemon core models, with a few minor adjustments. The details are presented in Chapter A.4.2.

Expressing the above mentioned example lexicalizations *sun* and *write* in *lemon* yields the following RDF:

```
:sun a lemon:Word ;
  lexinfo:partOfSpeech lexinfo:commonNoun ;
  lemon:canonicalForm :sun_canonicalForm ;
  lemon:sense         :sun_sense .

:sun_canonicalForm lemon:writtenRep "sun"@en .

:sun_sense lemon:reference <http://dbpedia.org/ontology/Star> .

:write a lemon:Word ;
  lexinfo:partOfSpeech lexinfo:verb ;
  lemon:canonicalForm :write_canonicalForm ;
  lemon:synBehavior   :write_synBehavior ;
  lemon:sense         :write_sense .

:write_canonicalForm lemon:writtenRep "write"@en .

:write_synBehavior a lexinfo:TransitiveFrame ;
  lexinfo:subject      :a1 ;
  lexinfo:directObject :a2 .

:write_sense lemon:reference <http://dbpedia.org/ontology/author> ;
  lemon:subjOfProp     :a2 ;
  lemon:objOfProp      :a1 .
```

Note that there can be different lexicalizations with the same reference, as well as one lexicalization with different references. An example of the former are the lexicalizations `star` and `sun`:

```
:sun lemon:sense :sun_sense .
:sun_sense lemon:reference <http://dbpedia.org/ontology/Star> .

:star lemon:sense :star_sense .
:star_sense lemon:reference <http://dbpedia.org/ontology/Star> .
```

An example of the latter would be the above mentioned `tall`:

```
:tall lemon:sense :tall_sense1, :tall_sense2 .
:tall_sense1 lemon:reference <http://dbpedia.org/ontology/height> .
:tall_sense2 lemon:reference <http://dbpedia.org/ontology/elevation> .
```

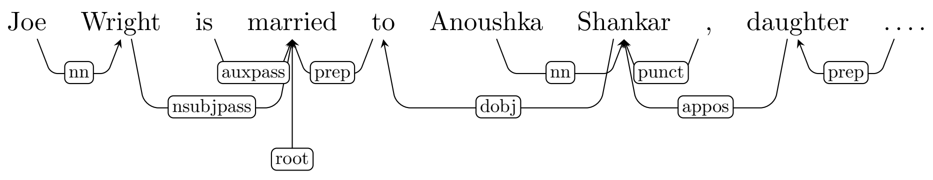


Figure 2.8: Dependency tree for the sentences *Joe Wright is married to Anoushka Shankar, daughter of Ravi Shankar and half-sister of Norah Jones.*

In addition, we enrich lexicalizations with provenance information specifying the source for that lexicalization, in particular the following information:

- *Frequency:* How often did the lemma occur in a given text corpus?
- *Pattern:* Which grammatical pattern matched

Moreover, for each lexicalization we store an example sentence, which does not only simplify debugging but makes a lexicalization more traceable throughout the whole process in M-ATOLL.

2.2.3 Dependency Parsing

Dependency parsing is a method to map a given sentences to its dependency structure. An example for such an input sentences is given in Example 2.9.

(2.9) Joe Wright is married to Anoushka Shankar, daughter of Ravi Shankar and half-sister of Norah Jones .

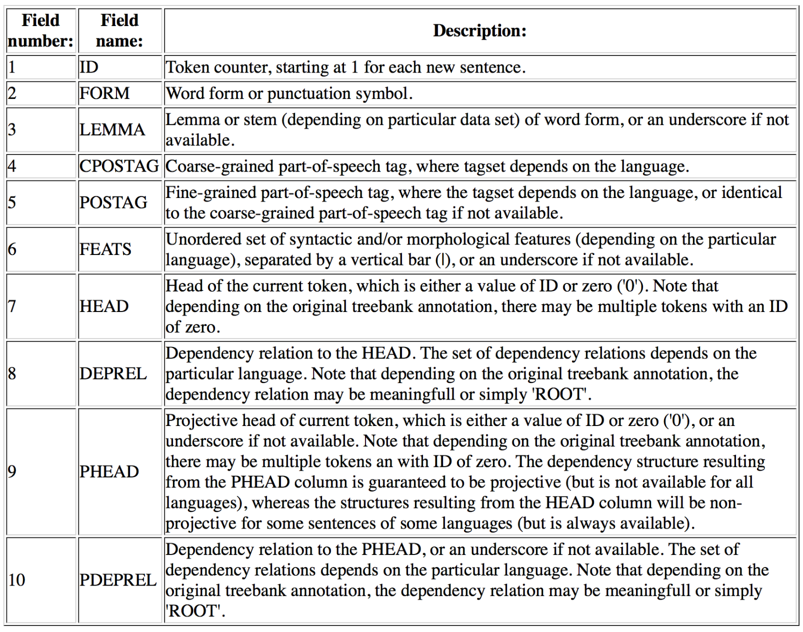
The goal of dependency parsing is to extract the functional structure of the sentence in a format that can be visualized as a dependency tree as presented in Figure 2.8. Dependency trees are directed trees. This example sentence was parsed with the MaltParser by Nivre et al. (2006). In difference to other parsers, such as presented in the work by Zettlemoyer and Collins (2005), which relay on a manually developed grammar, the MaltParser is a data-driven dependency parser, which means that it uses a statistical model to parse, which only works if previously enough training examples were given. Such models are usually trained on treebanks[8] as input data. Treebanks contain examples of sentences and their manually annotated syntactic tree.

Another example of such a data driven dependency parser is the Stanford Dependency Parser by de Marneffe et al. (2006).

The dependency tree from Figure 2.8 has to be read as follows:

The root node, or head, of the tree is the term `married`. This node has three child nodes, namely the words `Wright`, `is` and the preposition `to`. While `is` stands alone and the term `Writh` is only modified by the first name `Joe`, the preposition `to` connects the rest of the sentence `Anoushka Shankar, daughter of Ravi Shankar and half-sister of Norah Jones` with the root node `married`.

[8]see the Penn Treebank Project https://www.cis.upenn.edu/~treebank/



Field number:	Field name:	Description:
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma or stem (depending on particular data set) of word form, or an underscore if not available.
4	CPOSTAG	Coarse-grained part-of-speech tag, where tagset depends on the language.
5	POSTAG	Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.
6	FEATS	Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.
7	HEAD	Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningfull or simply 'ROOT'.
9	PHEAD	Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).
10	PDEPREL	Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningfull or simply 'ROOT'.

Figure 2.9: CoNLL format as presented here: `http://ilk.uvt.nl/conll/`

We use the grammatical information from the dependency tree in order to generate grammatical patterns which abstract of the actual words in the sentence.

Another way of displaying the output of the dependency parser is in the so called CoNLL format, a text serialization of the tree itself.

We use this format to transfer the parsed sentences into an RDF representation on which we then apply our grammatical patterns encoded as SPARQL queries. The RDF vocabulary, which we present later, is based on the vocabulary of the CoNLL format. The vocabulary of CoNLL is described in Figure 2.9, and the example tree from above is visualized in this format in Figure 2.10 on the next page.

Generally speaking the CoNLL format is a row and column based format. Each row represent the features of one word of the given sentences. Each column represents a different feature. The first row of Figure 2.10 represents the term `Joe`. The first column of this row represent the unique id of the term. The ids are assigned per sentence. The second column represents the term itself, in this case `Joe`. In the third column the lemmatized form of the term is given, e.g for the term `marries` the lemmatization `marry` would be displayed. For our English pared sentences we do not have any information on this position, as we create the lemmatization later with other resources. The next two columns represent different versions of the part-of-speech tag. While in this case both columns are equivalent, for our Spanish dependency parsed sentences the vocabulary in this columns differ. The second last column gives the id of the term to which the grammatical relation (last column) is pointing. The term `Joe` with the id *1* points as an *nn* towards the term with the id *2*, namely the term `Wright`. Another example from this figure is the term `to` with the id *5*

```
1 Joe _ NNP NNP _ 2 nn
2 Wright _ NNP NNP _ 4 nsubjpass
3 is _ VBZ VBZ _ 4 auxpass
4 married _ VBN VBN _ 0 null
5 to _ TO TO _ 4 prep
6 Anoushka _ NNP NNP _ 7 nn
7 Shankar _ NNP NNP _ 5 dobj
8 ,_ ,,_ 7 punct
9 daughter _ NN NN _ 7 appos
10 of _ IN IN _ 9 prep
11 Ravi _ NNP NNP _ 12 nn
12 Shankar _ NNP NNP _ 10 pobj
13 and _ CC CC _ 9 cc
14 half-sister _ NN NN _ 9 conj
15 of _ IN IN _ 9 prep
16 Norah _ NNP NNP _ 17 nn
17 Jones _ NNP NNP _ 15 pobj
18 . _ . . _ 4 punct
```

Figure 2.10: Example CoNLL representation for the sentence `Joe Wright is married to Anoushka Shankar, daughter of Ravi Shankar and half-sister of Norah Jones.`

which points as *prep* to the term `married` with the id *4*.

Chapter 3

Related Work

In this chapter we present related work for our framework M-ATOLL. The main related work comes from the field of Information Extraction (IE). Therefore, we give a short overview of the different tasks and areas of IE. While the general methodology of IE is very close to M-ATOLL, with our framework we solve a different, more specific task, namely the lexicalization of ontologies. However, as most systems use IE to solve this task, the lines between both tasks are oft blurred. We close this chapter with a discussion of related work from the field of Question Answering as well as related work from the area of lexicalization.

3.1 Information Extraction (IE)

The overall goal of IE is to extract relevant information from given textual data. Whether the information is relevant or not is defined by the task the IE system has to solve. Consider the MUC-dataset[1]. Using this dataset the relevant information to be extracted is data about terrorist attacks which are then filled into predefined templates. Other examples are the dataset from various shared tasks and conference series such as *BioCreative 2016*[2] where the relevant information are named entities from the chemical and biomedical domain, which have to be detected and extracted.

M-ATOLL uses many different techniques and ideas from the global field of IE. Therefore we present shortly the main areas of IE, namely Ontology-based Information Extraction, Open Information Extraction, Template- or Rule-based Information Extraction, Entity Recognition, Entity Linking, Coreference Resolution and Terminology Extraction.

3.1.1 Ontology-based Information Extraction

In Ontology-based Information Extraction (OBIE), the extraction process is driven by a given ontology. The basis of the process are the terms and concepts from the source ontology.

A good overview over this topic is given in the survey by Wimalasuriya and Dou (2010). After presenting different interpretations of the definition of an OBIE system, the authors present the common architecture of such systems and present some systems in more detail. That the topic of ontology-based information extraction is not only limited to research in natural language processing is shown in the work by Saggion et al. (2007), where an OBIE system was applied to a real world application in the business domain. The authors of this paper used 6 key economic indicators from Indian regions, extracted from Wikipedia pages, in order to evaluate their system. Overall, they gained an F-measure score of 81% on this evaluation.

A more recent approach is presented by Nebhi (2012), where tweets from Twitter are analyzed with a rule based system in order to recognize and disambiguate entities from the DBpedia and the Freebase ontologies. However, this paper also shows that the boundaries between the different areas of IE are very blurred and often a system can be categorized into multiple areas. The authors evaluated their system using a corpus of 115 short messages from BBC News, New York Times and The Times Twitter accounts. Overall, the system achieved a F-measure between 86% and 90%. However, the authors did not use a publicly available dataset but annotated the dataset themselves with DBpedia concepts.

3.1.2 Open Information Extraction (OpenIE)

A system is considered to be an Open Information Extraction system if structured relations are extracted, without specifying the structure of the pattern itself. Normally the extracted information is presented in a triple format, containing the left and right anchor, as well as the relation connecting these anchors. An example for this is the triple *(Alice;born in;Bielefeld)*, which represents the information extracted from the sentence `Alice was born in Bielefeld`. While this

[1]See http://www.itl.nist.gov/iaui/894.02/related_projects/muc/muc_data/muc_data_index.html
[2]See http://www.biocreative.org

notation resembles the triple notion in RDF, these two representations are not equivalent. An example for an OpenIE system is the *Stanford OpenIE* system by Angeli et al. (2015). This system reduces each input sentence to a set of entailed clauses. After reducing each clause to the most possible shortened sentence fragments, these fragments are sorted into OpenIE triples and returned as output. As raw data, the *Stanford OpenIE* system uses dependency-parsed sentences, but, differently from M-ATOLL, does not use any predefined grammatical patterns as we do, but uses all possible connections in the dependency tree. *Stanford OpenIE* was evaluated using the KBP Slot Filling task, performing with an F-measure of 28.3%.

Espresso (see Pantel and Pennacchiotti (2006)) employs a minimally supervised bootstrapping algorithm which, based on only a few seed instances of a relation, learns patterns that can be used to extract more instances. *Espresso* is thus comparable to our approach in the sense that both rely on a set of seed sentences to induce patterns. In our case, these are derived from a knowledge base, while in the case of *Espresso* they are manually annotated. Besides a difference in the overall task (relation extraction in the case of *Espresso* and ontology lexicalization in our case), one difference is that *Espresso* uses string-based patterns, while we use dependency paths, which constitutes a more principled approach to discarding modifiers and yielding more general patterns. A system that is similar to *Espresso* and uses dependency paths was proposed by Ittoo and Bouma (2010). A further difference is that *Espresso* leverages the web to find further occurrences of the seed instances. The corpus we use, Wikipedia, is bigger than the compared text corpora used in the evaluation by *Espresso*. But it would be very interesting to extend our approach to work with web data in order to overcome data sparseness, e.g. as in the work by Blohm and Cimiano (2007), in case there is not enough instance data or there are not enough seed sentences available in a given corpus to bootstrap the pattern acquisition process.

A very recent approach, falling into the category of **Template- or Rule-based Information Extraction** is the approach by Mahendra et al. (2011), in which the authors extract lexicalizations of DBpedia properties on the basis of a Wikipedia corpus, but, in contrast to our approach, do not consider the parse of a selected sentence, but the longest common substring between domain and range of the given property. The domain and range are normalized by means of DBpedia class labels, such as Person or Date.

Another very prominent example for this category is the system called *BOA* by Gerber and Ngomo (2011, 2012). Similar to the spirit of M-ATOLL, *BOA* relies on existing triples from a knowledge base, in particular DBpedia. *BOA* applies a recursive procedure, starting with extracting triples from linked data, then extracting natural language patterns from sentences. In the final step these extracted patterns are used to extract more instances, which can be used as new input. These instances are then added to the knowledge base. The main difference to our approach is that *BOA* relies on simple string-based generalization techniques to find lexicalization patterns. This makes it difficult, for example, to discard optional modifiers and thus can generate a high amount of noise, which has been corroborated by initial experiments in our lab on inducing patterns from the string context between two entities.

In order to compare our approach to other state-of-the-art systems, such as *BOA*, we implemented a baseline system that rests on the actual ideas of the later.

Other systems which extract relations from text corpora on the basis of predefined patterns are *ReVerb* by Fader et al. (2011), *ClausIE* by Corro and Gemulla (2013), Ravichandran and Hovy (2002), and Girju (2003), to name just a few. Very often the patterns used are quite restricted and overlap with the ones we defined. In contrast to M-ATOLL, they are usually defined as regular expressions, such as such as '$<$ NAME $>$ ($<$ BIRTHDATE $>$ $-$', and do not rely on grammatical information beyond part-of-speech tags.

A system supporting multilinguality is *WRPA* by Vila et al. (2010, 2013), which extracts English and Spanish relational paraphrases from the English and Spanish Wikipedia, respectively. Like other approaches, *WRPA* considers only the textual pattern between two anchor texts from Wikipedia, no parse structure. In difference to M-ATOLL, *WRPA* does not start with structured data from knowledge bases, but extracts entity pairs from structured information within Wikipedia. The authors apply distant learning and use these extracted pairs, similar to our approach, as anchor points for the candidate paraphrase extraction. The extraction itself is done by identifying the most common strings between similar given entity pairs. The idea is that only if a paraphrase occurs multiple times between similar entity pairs, it is a valid paraphrase candidate. We use a similar idea by adding lexical entries only to the final lexicon if they were extracted more than once. The authors then applied a decision tree on the paraphrase candidates in order to extract the final relation paraphrases. However, it is important to mention that the authors trained and applied this classifier only for the relation `authorship` in the Spanish language. This limitation allows the creation of a very effective classifier at the cost of reduced generality; within the framework of M-ATOLL, our goal is to keep the system as general as possible and avoid unnecessary dependencies towards certain domains.

An approach to extracting lexicalization patterns from corpora that is similar in spirit to our approach is *Wanderlust* (see Akbik and Broß (2009)), which relies on a dependency parser to find grammatical patterns in a given corpus—Wikipedia in their case as in ours. These patterns are generic and non-lexical and can be used to extract any semantic relation. However, *Wanderlust* also differs from our approach in one major aspect. We start from a given property and use instance data to find all different lexical variants of expressing one and the same property, while *Wanderlust* maps each dependency path to a different property (modulo some post processing to detect sub relations). *Wanderlust* is therefore not able to find different variants of expressing one and the same property, thus not allowing for semantic normalization across lexicalization patterns.

DIRT by Lin and Pantel (2001) relies on a similarity-based approach to group dependency paths, where two paths count as similar if they show a high degree of overlap in the nouns that appear at the argument positions of the paths. Such a similarity-based grouping of dependency paths could also be integrated into our approach, in order to find further paraphrases. The main difference to our approach is that *DIRT* does not rely on an existing knowledge base of instantiated triples to bootstrap the acquisition of patterns from textual data, thus being completely unsupervised. Given the fact that nowadays there are large knowledge bases such as Freebase and DBpedia, there is no reason why an approach should not exploit the available instances of a property or class to bootstrap the acquisition process.

A very similar approach to *DIRT* is the system *Snowball* proposed by Agichtein and Gravano

(2000).

The system *OLLIE* by Mausam et al. (2012) is a successor to *ReVerb* and the work presented by Wu and Weld (2010). The later two systems have two important weaknesses according to the authors of *OLLIE*. On the one hand, only verbs are considered for the relation extraction process. On the other hand, the context of the whole sentence is not considered. *OLLIE* however tries to solve both weaknesses by combining results from *ReVerb* with a corpus and dependency based approach. In the first step, *OLLIE* uses around 100.000 high quality tuples, provided by *Reverb*, in order to extract sentences from the web where both entities from the tuple occurs. After dependency parsing these sentences with the MaltParser, *OLLIE* extracts the most common syntactic patterns, connecting the entities of the given tuple. The most prominent syntactic patterns are then applied to new sentences in order to extract new relational tuples.

One approach that also incorporates dependency information is *PATTY* by Nakashole et al. (2012), a corpus-based approach to extract relational patterns together with semantic restrictions, such as [*SINGER*] *sings* [*SONG*]. Similar to M-ATOLL, entities from a given knowledge base, in this case YAGO, are used to extract all sentences from Wikipedia in which both the triple subject and triple object occur. Using the Stanford Parser, the authors of *PATTY* make sure that the syntactic subject is equivalent to the semantic subject and the syntactic object is equivalent to the semantic object. This is a limitation compared to M-ATOLL, as we consider all cases and then encode it in the grammatical dependency in our lexical entry. The shortest path on the dependency graph between both entities is then extracted as the relation. Finally, the pattern is generalized according to semantic types, e.g. into [*PERSON*] *loves* [*PERSON*].

Closely related to this approach is also *Sargraphs* by Krause et al. (2016), an approach to extract graphs that link semantic relations from knowledge bases with linguistic representations in natural language. Similar to M-ATOLL, *Sargraphs* is a corpus-based approach to extract lexicalizations from text. The main difference is that *Sargraphs* is not restricted to binary relations but targets more complex event-like structures, and that it uses the shortest path between two entities on a dependency graph to extract the lexicalization.

3.1.3 Entity Recognition

Entity Recognition (ER), often also called Named Entity Recognition (NER), is the task of identifying entities, such as *Barack Obama*, in a given natural language text. For example, consider the sentence `The President of the United States, Barack Obama, loves his wife`. In this sentence *President of the United States* and *Barack Obama* are entities, and the goal of a ER system is to find and mark those. In theory the term *wife* also represents an entity, however most ER systems do not recognize *wife* as a named entity.

One of the most famous tools in this area is the *Stanford Named Entity Recognizer*[3] by Finkel et al. (2005) in which linear chain Conditional Random Field (CRF) sequence models are used in combination with labeled training data to solve this task very accurately. CRFs were firstly pioneered by Lafferty et al. (2001), Sutton and McCallum (2006) and Sutton and McCallum (2010). On the CoNLL 2003 named entity recognition dataset the *Stanford Named Entity Recognizer*

[3] `http://nlp.stanford.edu/software/CRF-NER.shtml`

performed (depending on the configuration) with a F1 score of 85.51% (respectively 86.86%) and on the CMU Seminar Announcements dataset with a F1 score of 91.85% (respectively 92.29%)

However, there are and were many other NER tools as shown in the survey by Nadeau and Sekine (2007).

While traditionally NER tools are limited to "normal" text, especially when taking the context of multiple sentences into account, recently (see Ritter et al. (2011) or Derczynski et al. (2015)) NER tools have been developed which work on much shorter sentences such as tweets from Twitter.

An example of a tool in the field of Chemistry is *ChemSpot*[4] by Rocktäschel et al. (2012), a tool for named entity recognition as well as classification of chemical names in natural language text.

However, NER is not limited to the English language but finds an increasing interest in other languages and is an active research field in different languages, such as Arabic, as shown in the survey by Shaalan (2014).

3.1.4 Entity Linking

The task of Entity Linking (EL) consists in disambiguating named entities occurring in textual data by linking them to a knowledge base identifier. Many systems rely in the first step on entity recognition tools and then perform linking only for the recognized named entities.

As an example for EL, consider again the example sentence `The President of the United States, Barack Obama, loves his wife`. An EL tool with a structured database in the background such as DBpedia, should be able to identify *President of the United States* and *Barack Obama* and link it to the corresponding resources *http://dbpedia.org/resource/President_of_the_United_States* and *http://dbpedia.org/resource/Barack_Obama*.

An example for such a tool is *Weasel* by Tristram et al. (2015), which relies on a machine based learning approach combining different features. The authors of the paper claim that the PageRank (see Page et al. (1999)) feature is one of the most influential features and is a strong indicator to solve the task of entity linking - if a structured database such as DBpedia is given. On the AIDA/CoNLL dataset *Weasel* achieves an F-1 score of 0.58 and on the KORE50 dataset a F-1 score of 0.28.

Other prominent tools for EL are *Bablefy* by Moro et al. (2014) , *DBpedia Spotlight* by Mendes et al. (2011), *AGDISTIS* by Usbeck et al. (2014) and *FOX* by Speck and Ngomo (2014). While *FOX* considers itself as a NER tool, it also provides as result disambiguated and linked named entities, which area available for a user in various serialization formats. On the AIDA/CoNLL dataset, *DBpedia Spotlight* achieves an F-1 score of 0.3 and on the KORE50 dataset a F-1 score of 0.27, while *FOX* achieves a F-1 score of 0.37 and 0.25, respectively to the dataset

Generally, a distinction is made between local and global EL systems. While the local EL systems link entities only for once sentence, not considering any context, a global EL system considers context, for example multiple documents. This consideration has an influence on the overall linking process, because the linking in an early stage of a document limits the possibility to link to different entities later in the document.

[4]Source code is available at `https://github.com/rockt/ChemSpot`

3.1.5 Coreference Resolution

If two expressions refer to the same real-world entity, these two expressions are called *coreferential*. When two expressions are coreferential, one is usually a full form (the antecedent) and the other is an abbreviated form. The task of coreference resolution is to cluster mentions that refer to the same real world entity (and therefore are coreferential) within one text or across several texts. For example, consider the sentence `Alice said she will clean the room` where *Alice* and *she* co-refer as they denote the same entity. The goal of a coreference resolution system therefore is to detect that both refer to each other and represent the same entity. A more complex case is given if multiple sentences (or documents) are given and therefore the context increases the complexity.

The most prominent example is the *Stanford Deterministic Coreference Resolution System* presented by Lee et al. (2011) and Raghunathan et al. (2010). The modular system is based on different sieves building always on top of the output of the previous sieves. The more sieves are used to find a coreference, the lower the precision is. It is implemented as a simple, deterministic system and does not rely on machine learning or detailed semantic information. Instead it uses information from the Stanford Parser and is organized into six sieves. The first one, the `Exact Match`, links two entities if their canonical forms are equivalent; the second sieve, called `Precise Constructs`, considers entities to be linked if the mentions are related in different grammatical relations in the dependency tree such as appositive or relative pronoun. The next sieves are `Strict Head Matching`, `Variants of Strict Head`, `Relaxed Head Matching` and systematically add a pronoun decomposition. The *Stanford Deterministic Coreference Resolution System* was recently extended by Recasens et al. (2013) with a new unsupervised method for mining opaque pairs by adding additional 5 sieves to the original system. On the CoNLL 2011 Shared Task dev data set, the *Stanford Deterministic Coreference Resolution System* ranked top in the challenge with an average F1 score of 59.56%.

Another example is the *Berkeley Coreference Resolution System* presented by Durrett and Klein (2013) and Durrett et al. (2013). In the first step, the authors use similar rules as proposed as sieves by Lee et al. (2011). However, instead of applying these rules as filters, every mention matching at least one of these rules is extracted. This leads firstly to a high recall but lower precision. Afterwards, these mentions are ranked by a machine learning approach using seven different surface features, such as the mentioned type (e.g. if it is a nominal, proper or pronominal) or the complete string of the mention. The authors show in their evaluation that these simple surface features are enough to beat other state-of-the-art systems on the same dataset. More preciously on the CoNLL 2012 dataset the *Berkeley Coreference Resolution System* achieved an average F-1 score of 61.21% in the final test.

A third example is *BART - a Beautiful Anaphora Resolution Toolkit* presented by Versley et al. (2008a) and Versley et al. (2008b). Compared to other coreference resolution systems, the authors divide their tool into three categories, each intended to solve a different task of the coreference resolution pipeline, joining the result before the final output. For each of the three tasks (binding constraint detection, expletive identification and aliasing), an own kernel function is implemented and trained, resulting into a great performance for each task and therefore a good performance for the overall task.

As a final example, we would like to mention the system *Reconcile* by Stoyanov et al. (2010). The authors of this work claim that, while this system and *BART* are very similar, their system is more flexible. However, the authors do not explain their assumption and do not provide any evaluation to support their claim. *Reconcile* uses five steps (Preprocessing, Feature generation, Classification, Clustering and Scoring) in order to detect coreferential expressions. In the preprocessing step, various linguistic tools such as a tokenizer, a part-of-speech tagger, a syntactic parser etc. are used, leading into over 80 different features for the classification.

3.1.6 Terminology Extraction

The goal of *Terminology Extraction* is to extract relevant terms from a given text corpus. Sometimes this area of IE is also called *Term Recognition* or *Glossary Extraction*. It is a very important first step for the creation of domain ontologies or terminology bases as shown for example in the work by Bourigault and Jacquemin (1999) and Park et al. (2002).

In the work by Wong et al. (2007) the authors present an approach to use term extraction in order to learn domain ontologies. In this work an approach containing a series of base and derived measures for recognizing terms are used. The authors in particular introduce two new measures, namely the *domain prevalence* and the *domain tendency*, which was used in combination with four other measures to compute a weight called `Termhood`, ranking the extracted terms by linguistic evidence.

A more recent approach is *TBXTools* presented by Oliver González and Vàzquez Garcia (2015) which is a free and automatic terminology extraction tool based on linguistic and statistical methods. For the linguistic based terminology extraction the authors rely on predefined pattern such as *NN NN* or *JJ NN* applied on a POS-tagged corpus. For the statistical part of the terminology extraction the authors rely on the calculation of different n-grams sets.

Saffron[5] by Buitelaar and Eigner (2009) and Monaghan et al. (2010) is a knowledge extraction framework, which uses key-phrase extraction, entity linking, taxonomy extraction, expertise mining, and data visualization. Beside domains from Natural Language Processing as well as the Information Retrieval and the Semantic Web area, in this project finance data from 2007 and 2008 were used.

There have been many other works regarding terminology extraction such as in the work by Maynard and Ananiadou (1999, 2000, 2001), to mention a few.

3.2 Question Answering

Since 1999, there have been a series of question answering challenges[6] at the Text REtrieval Conference (TREC). For the last couple of years, a new QA challenge was defined, outside of TREC, targeting Question Answering over Linked Data (QALD)[7]. These challenges provide multilingual questions which have to be answered using DBpedia as knowledge base, see Cabrio et al. (2013),

[5]See `http://saffron.insight-centre.org`
[6]See `http://trec.nist.gov/data/qa.html`
[7]See `http://www.sc.cit-ec.uni-bielefeld.de/qald`

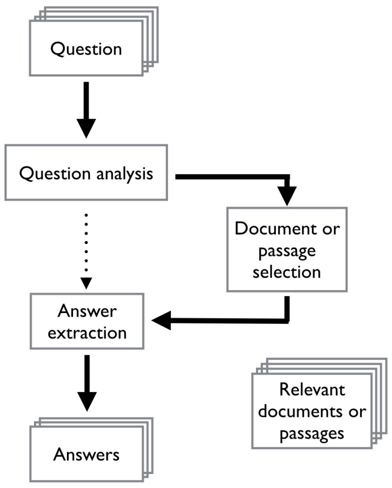


Figure 3.1: Generic QA system architecture as proposed by (Indurkhya and Damerau, 2010, p.489) in the chapter *Question Answering*

Cimiano et al. (2013) and Unger et al. (2014a). An extensive introduction to QALD is given in the work by Unger et al. (2014b). QALD is very relevant to this work, as one of the main problems in mapping from the question to a structured database is the lexical gap as shown in the work by Hoeffner et al. (2016). Therefore, many question answering systems rely on techniques from the semantic web, e.g. linked data, as described in the work by Lopez et al. (2011).

Generally speaking, Question Answering

> "can be defined as an automatic process capable of understanding questions formulated in natural language such as English and responding exactly with the requested information." (Indurkhya and Damerau, 2010, p.485)

The general architecture of a QA system is despited in Figure 3.1. In the first step, a question is given as input, which is analyzed in the second step. Based on this analysis, documents or passages are selected, from which hypotheses for answers are extracted. After ranking these hypotheses, the actual answer(s) is(are) presented.

Big parts of the research community focus on so called factoid QA systems, where the goal is to extract a concrete answer for a given question. Consider the question `Who is the wife of Barack Obama?`, here the factoid answer is `Michelle Obama`. A non factoid-question for example could ask to explain how to fix a certain airplane and involves more complex reasoning than factoid QA systems. As an example for a non-factoid QA system, consider the system proposed by Surdeanu et al. (2011). In this work the authors implement and evaluate a non-factoid QA system with a machine learning approach, using a large community-generated question-answer collection as input.

However, in this thesis we concentrate on factoid QA systems. There are typically two kinds of systems for factoid QA. The first kind of system uses unstructured data, such as documents as knowledge base, while the second kind uses structured databases as knowledge base. However, there are systems which combine both type of QA systems in order to maximize the coverage of the systems. Such a system was for example proposed in the work by Cucerzan and Agichtein (2005) and in the work by Usbeck et al. (2015).

QA is the task of mapping natural language to a structured database in order to retrieve a (hopefully) correct answer. Therefore, lexical resources such as presented in this work could lead to an increase in the performance of such systems. For example, in the work by Hakimov et al. (2015), the authors implemented a QA system based on CCG rules and showed in the evaluation that the accuracy could be improved by lexical entries provided by M-ATOLL. Similar results are shown in the work by Ravichandran and Hovy (2002) and Girju (2003), which both show that incorporating the extracted relations in a question answering system leads to an increase in the number of correctly answered questions.

There are also other question answering systems (see Hakimov et al. (2013) and He et al. (2014)), which use relation extracted by the system called *Patty*, which we described earlier, to solve the lexical gap.

Of course there are other ways to address this lexical gap. For example the system *Bela* by Walter et al. (2012) uses a layered approach to tackle this problem. After using lexical corpora, such as WordNet, the authors used in the final step the Explicit Semantic Analysis (ESA), proposed by Gabrilovich and Markovitch (2007), to map from the natural language input towards the structured data. While this layered approach showed to be very powerful, preliminary experiments showed that *Bela* can be improved by incorporating results from M-ATOLL.

Other prominent systems in the area of QALD are *QAKiS* by Cabrio et al. (2012), *PowerAqua* by Lopez et al. (2012) and *TBSL* by Unger et al. (2012), to name a few.

3.3 Lexicalization

There are some works which address the task M-ATOLL focus on, the task of finding lexicalizations of ontology entities.

One paper, called *Lexicalization of an ontology* by Peters et al. (2007), uses a similar approach to our label-based approach. The authors also use WordNet as main resource, however they do not only lexicalize properties and classes which had a given label (as we are doing), but also use the extracted label from the given URI when no ”official” label is given. While we rely on a LESK-inspired (see Lesk (1986)) selection strategy to select the correct synset, the authors of this paper use a semi-automatic approach where lexical candidates are presented and then selected by users to find the correct lexicalization for the given property and classes. This work already supports multilinguality.

There has been a substantial body of work on lexical acquisition from corpora, for example by Boguraev and Pustejovsky (1996); Pustejovsky (1992). One important task in lexical acquisition is the extraction of verb frames and subcategorization frames from corpora (see Preiss et al. (2007)). There has also been a lot of work on clustering adjectives by their semantics by Boleda Torrent and Alonso i Alemany (2003) and Chen and Chen (1994). Further work has addressed the extraction of hyponyms and hypernyms from corpora (see Hearst (1992)), and the identification of the particular meaning of adjectives in adjective-noun phrase compounds, e.g. relying on topic models (see Hartung and Frank (2011)), pattern based distributional semantic model (see Hartung and Frank (2010b)), and classification of adjectives to relational types (see Hartung and Frank (2010a)). Adjectives have also been exploited as attributes to cluster nouns as proposed by Almuhareb and Poesio (2004).

However, there has been little work so far on identifying the meaning of adjectives with respect to existing large ontologies such as DBpedia. For example in the work by Maillard and Clark (2015) the authors have proposed a tensor-based framework using skip-gram models to represent the meaning of adjectives. This approach induces a latent semantics from a corpus rather than trying to capture the semantics of an adjective with respect to existing ontologies.

In contrast, the work by Boleda Torrent et al. (2004) creates an ontology lexicon for Catalan, starting from a pre-defined list of adjectives, which were then classified into unary and binary (relational) adjectives.

Chapter 4

M-ATOLL: Framework and Experiments

This chapter introduces the reader to the two main approaches of our framework.

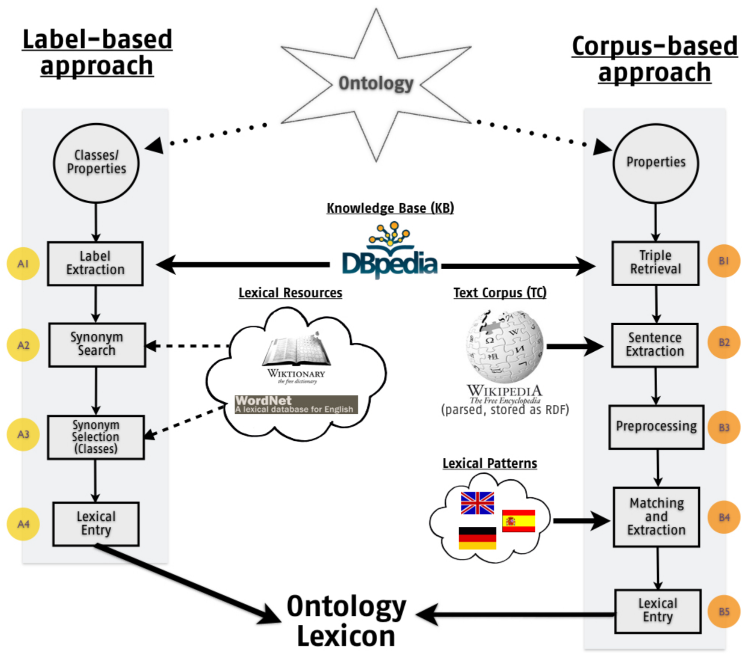


Figure 4.1: General Architecture of M-ATOLL

4.1 Overview

The architecture of M-ATOLL is depicted in Figure 4.1. It comprises two separate approaches: a *Label-based approach* and a *Corpus-based approach*. In all cases the input is an ontology and the output is a lexicon serialized in *lemon* format, lexicalizing the input ontology.

All approaches differ in the way this lexicon is induced. While the label-based approach generates lexical entries for both classes and properties based on the analysis of the labels and external linguistic resources, the corpus-based approach uses a text corpus to match predefined linguistic patterns (see Section 4.5) and thus retrieve candidate lexicalizations for ontology properties.

The label-based approach is so far implemented only for English, while the corpus-based approach supports English, German and Spanish (and is currently also ported to Japanese, see Lanser et al. (2016)).

The Label-based approach, marked with **A** in Figure 4.1, consists of four steps:

A1 For a given class or property, extract the corresponding label.

A2 For this label, retrieve synonyms from some lexical resources, e.g. WordNet or Wiktionary.

A3 Select the most relevant synonym in order to retrieve alternative lemmas.

A4 Create lexical entries based on the part-of-speech tags of those lemmas.

If no synonym is available (in step A2), the lexical entry is created based on the part-of-speech tag of the actual label.

The Corpus-based approach, marked with **B** in Figure 4.1, works as follows.

B1 For a given property p, extract all RDF triples.

B2 From the accompanying text corpus, extract all sentences that contain mentions of s and o.

B3 Preprocess the extracted sentences (e.g. marking of entities).

B4 Match the predefined linguistic patterns on those sentences.

B5 For every match, generate a lexical entry.

This approach does not work for classes as usually the necessary RDF triples are not available for classes.

For this approach our text corpus (for step B2) consists of 84,342,839 English, 37,701,496 German and 17,068,990 Spanish dependency parsed sentences from Wikipedia in the corresponding language. The URL to download these files is presented in Chapter 7.

4.2 Label-based Approach

The algorithm in pseudo code for this approach is presented in Algorithm 1. It takes an ontology as input, together with a knowledge base and a text corpus, and returns a lexicon for the ontology. For each class and property from the ontology, it collects lexicalization terms (line 3). To this end it first extracts the label of the class or property from the knowledge base (lines 5-7). Then, in addition, it retrieves the synsets of each label from WordNet (line 8). In the case of classes, a synonym selection is carried out (lines 9-10), before adding the labels of the possible synonym candidates as lexicalization terms. Finally, for each thus collected term a lexical entry is generated (lines 22-26). In the function *createLexicalEntry* we create a lexical entry in *lemon* format by mapping the URI, the part-of-speech as well as the term t to a predefined template.

Algorithm 1 Label-based approach for ontology lexicalization

```
 1: Input: Ontology $O$, Knowledge base $KB$, Text corpus $TC$
 2: Output: Lexicon

 3: $terms \leftarrow \{\}$
 4: for each property or class $uri \in O$ do
 5:     $labels \leftarrow \{\ l \mid \langle uri, \texttt{rdfs:label}, l\rangle \in KB\}$
 6:     for $l$ in $labels$ do
 7:         add $(l, uri)$ to $terms$
 8:         $synsets \leftarrow getSynsets(l, WordNet)$
 9:         if $uri$ is a class then
10:             $synonyms \leftarrow$ Algorithm 2 $(uri, synsets, KB, TC)$
11:         else
12:             for $s$ in $synsets$ do
13:                 $synonyms \leftarrow$ getLabelsforSynset(s)
14:             end for
15:         end if
16:         for $s$ in $synonyms$ do
17:             add $(s, uri)$ to $terms$
18:         end for
19:     end for
20: end for

21: $lexicon \leftarrow \{\}$
22: for $(t, uri)$ in $terms$ do
23:     $pos \leftarrow getPartOfSpeech(t)$
24:     $entry \leftarrow createLexicalEntry(uri, t, pos)$
25:     add $entry$ to $lexicon$
26: end for
27: return $lexicon$
```

For the DBpedia class *dbo:Activity*, for example, we retrieve its label `activity` as well as from WordNet the terms `action`, `body process`, `bodily function`, `bodily process`, `natural action`, `activeness` and `natural process`. The filled template, instantiated in Java for the class *dbo:Activity* with the label `activity` is presented in Section A.1 in Example A.1 on page 128.

In order to select relevant lexicalization candidates from the retrieved synonyms of class labels, we implemented a LESK-inspired (see Lesk (1986)) synonym selection algorithm, which is presented in Algorithm 2. First, we collect a word context for the input class. To this end, we retrieve 100 instances of this class from the knowledge base, together with their labels (lines 4-6). For each of these labels, we retrieve those sentences from the text corpus that contain this label (line 8 – for reasons of space and memory, we take only up 1000 sentences) and add each word in that sentence that is not a stop word to the context (lines 9-13). Then we use the thus collected word context in order to score how well each synonym fits that context. To this end, for each synonym we calculate the normalized Levenshtein distance to each of the context words (line 24) and average those values (line 27), considering only values greater 0. Finally, all synonyms with an average distance less than 0.5 are discarded (lines 28-30). This results into the creation of two entries, one for the canonical form *activity* and one for the canonical form *action*.

We are aware of the fact that our LESK-inspired synonym selection should be replaced by a more state-of-the-art word sense disambiguation step, such as with systems presented in the survey by Navigli (2009). However, for the purposes of this thesis, this simple synonym selection approach is sufficient as proof-of-concept.

Algorithm 2 Selection of synonyms for ontology classes

```
 1: Input: Class $uri$, $synsets$, Knowledge base $KB$, Text corpus $TC$
 2: Output: Disambiguated terms

 3: $context \leftarrow \{\,\}$
 4: $instances \leftarrow \{\, i \mid \langle i, \texttt{rdf:type}, uri \rangle \in KB \}$
 5: for first 100 elements $i$ in $instances$ do
 6:     $labels \leftarrow \{\, l \mid \langle i, \texttt{rdfs:label}, l \rangle \in KB \}$
 7:     for $l$ in $labels$ do
 8:         $sentences \leftarrow \{\, s \mid s$ is a sentence in $TC$ and contains $l \}$
 9:         for $w$ in $words(s)$ do
10:             if not $w$ a stop word then
11:                 add $w$ to $context$
12:             end if
13:         end for
14:     end for
15: end for

16: for $s$ in $synsets$ do
17:     $synonyms \leftarrow$ gettSynonymsForSynset(s)
18: end for
19: for $s$ in $synonyms$ do
20:     $distances \leftarrow \{\,\}$
21:     for $w$ in $context$ do
22:         $distance \leftarrow normalizedLevenshteinDistance(s, w)$
23:         if $distance > 0$ then
24:             add $distance$ to $distances$
25:         end if
26:     end for
27:     $distance \leftarrow average(distances)$
28:     if $distance < 0.6$ then
29:         remove $s$ from $synonyms$
30:     end if
31: end for
32: return $synonyms$
```

4.3 Corpus-based Approach

The corpus-based approach consists of two separate steps. The first step extracts for any property those sentences from a given text corpus that contain candidate lexicalizations of the property.

The sentences are then used as input to the second step that extracts those lexicalizations.

The sentence extraction is shown in Algorithm 3. Input are the properties of an ontology and a dependency-parsed text corpus; output is a set of preprocessed sentence parses in RDF format. First, for each property p in the ontology, all pairs of entities that are related via p are retrieved from the knowledge base (line 5). For each such entity pair, we then retrieve all sentences from the text corpus that contain the labels of both entities (lines 7-9). In our implementation we store the parsed text corpus in a Lucene index and always return the top k relevant sentences,[1] according to the internal Lucene score. The retrieved (already dependency parsed) sentences are then converted into a CoNLL-based RDF format[2] (line 11) that in the second step allows to declaratively define and match lexicalization patterns. An example of the RDF-Format is visualized in Figure 4.3 on page 51, it supports a large subset of the CoNLL vocabulary, namely *deprel*, *feats*, *form*, *head*, *postag*, *lemma* and *cpostag*. The last two items (*lemma* and *cpostag*) are not displayed in the given example. Additionally we enriched the format to encode the *id* of a sentence (in case multiple sentences are stored in one RDF file), as well as additional information, such as the subject and object of the property for which the sentence was retrieved.

Algorithm 3 Corpus-based approach for ontology lexicalization: Step 1 (sentence extraction)

```
 1: Input: Ontology $O$, Dependency-parsed text corpus $TC$
 2: Output: RDF dataset

 3: $dataset \leftarrow \{\}$
 4: for property $p \in O$ do
 5:     $entities \leftarrow \{ (e_1, e_2) \mid \langle e_1, p, e_2 \rangle \in KB\}$
 6:     for $(e_1, e_2)$ in $entities$ do
 7:         $l_1 \leftarrow$ label of $e_1$, i.e. $l$ such that $\langle e_1, \texttt{rdfs:label}, l \rangle \in KB$
 8:         $l_2 \leftarrow$ label of $e_2$, i.e. $l$ such that $\langle e_2, \texttt{rdfs:label}, l \rangle \in KB$
 9:         $sentences \leftarrow \{ s \mid s \in TC$ and $s$ contains $l_1$ and $s$ contains $l_2\}$
10:         for $s$ in $sentences$ do
11:             convert $s$ to RDF and add to $dataset$
12:         end for
13:     end for
14: end for

15: return $dataset$
```

The second step of the corpus-based approach, shown in Algorithm 4, is the heart of M-ATOLL. As input it takes an ontology and a list of parsed sentences in RDF format, as returned by Step 1; as output it returns the final lexicon. First, for each property p in the ontology, the list of corresponding sentences in RDF format is retrieved (line 5). Then a list of predefined, language-specific linguistic patterns is matched on each sentence. If a pattern matches, the corresponding lexical entry candidate is created.

The linguistic patterns currently used for each implemented language are described in more detail in Section 4.5. The patterns themselves are implemented as SPARQL queries over the CoNLL-based RDF format for dependency-parsed sentences.

Finally, the set of candidate lexical entries is filtered based on its frequency (line 19), in order

[1] Usually we limit k to 1000.

[2] Following the CoNLL specification at http://ilk.uvt.nl/conll/.

Algorithm 4 Corpus-based approach for ontology lexicalization: Step 2 (pattern-based extraction of lexicalizations)

```
1: Input: Ontology $O$, RDF dataset of parsed sentences
2: Output: Lexicon

3: $candidates \leftarrow \{\ \}$
4: for property $uri \in O$ do
5:     for $s \in sentences(uri, dataset)$ do
6:         for pattern $p$ in a predefined set of linguistic patterns do
7:             if $p$ matches $s$ then
8:                 $entry \leftarrow createLexicalEntry(uri, s, p)$
9:                 if not $entry \in candidates$ then
10:                    set $frequency(entry)$ to 1
11:                    add $entry$ to $candidates$
12:                else
13:                    increase $frequency(entry)$ by 1
14:                end if
15:            end if
16:        end for
17:    end for
18: end for

19: return $\{ entry \mid entry \in candidates$ and $frequency(entry) > 1\}$
```

to reduce noise in the final lexicon. Currently we accept all entries that occur more than once. However, in Chapter 5 we will evaluate different selection strategies and discuss the advantages and disadvantages compared to this frequency based selection strategy. The final lexicon is then serialized as *lemon* RDF.

4.4 Examples

We will now walk through the corpus-based approach by considering one object property, *dbo:spouse*, and one datatype property, *dbo:birthdate*, from the DBpedia ontology as examples, showing which English lexicalizations are extracted.

Object Property Example: *dbo:spouse*

First, all pairs of entities that are connected by the property *dbo:spouse* are retrieved together with their corresponding labels. This includes, for example, the following pairs and corresponding labels:

- ⟨ `Carl_Sagan` , `Lynn_Margulis` ⟩
  Carl Sagan *Lynn Margulis*
- ⟨ `Sulla` , `Valeria_(wife_of_Sulla)` ⟩
  Lucius Cornelia Sulla *Valeria*

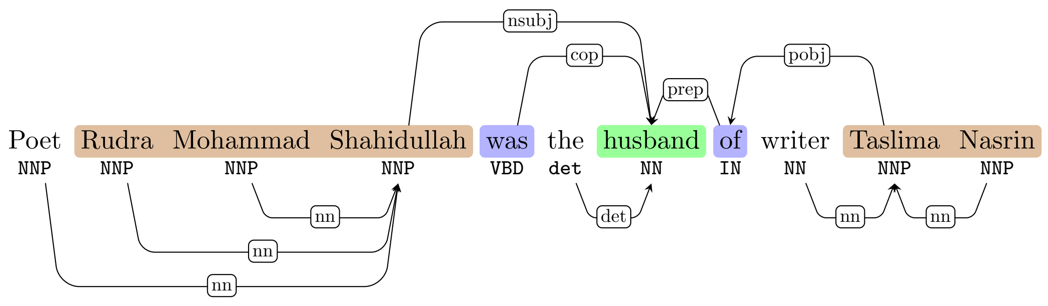


Figure 4.2: Dependency tree for the sentence `Poet Rudra Mohammad Shahidullah was the husband of writer Taslima Nasrin`.

- ⟨ `Taslima_Nasrin` , `Rudra_Mohammad_Shahidullah` ⟩
  Taslima Nasrin *Rudra Mohammad Shahidullah*

Since it happens that the texts in different language chapters of Wikipedia contain mentions of entity labels in other languages (e.g. the German and Spanish Wikipedia texts contain mentions of the English labels, and the English texts on German and Spanish entities contain mentions of their German and Spanish labels, respectively), M-ATOLL extracts all labels, irrespective of the language. For `spouse`, this yields 78,015 distinct entity pairs.

The text corpus we used is based on a dump of the English Wikipedia from April 2014. This corpus is dependency parsed using the MaltParser by Nivre et al. (2006) with a pre-trained English model provided by the MaltParser. All dependency parsed sentences are stored in CoNLL[3] format in a Lucene Index[4]. For each entity pair, we then search the sentence index for sentences that contain mentions of both their labels[5] (considering the top 1000 sentences according to the internal Lucene score). In the case of *dbo:spouse*, this retrieves 6,075 sentences, for example:

- `Lynn Margulis married astronomer Carl Sagan in 1957.`
- `Valeria was the fifth wife of Roman dictator Lucius Cornelius Sulla.`
- `Poet Rudra Mohammad Shahidullah was the husband of writer Taslima Nasrin.`

In the following we will concentrate on the third example sentence, `Poet Rudra Mohammad Shahidullah was the husband of writer Taslima Nasrin`. The structure of the dependency graph is shown in Figure 4.2 and a snippet of the corresponding RDF representation is shown in Figure 4.3 on the next page.

Next, for all defined linguistic patterns it is checked whether they match the RDF representation of the extracted sentences. Since those patterns are defined as SPARQL queries, this amounts to a simple query operation on the triple store. If there is a match, the lexical entry associated

[3]See `http://ufal.mff.cuni.cz/conll2009-st/task-description.html`
[4]See `https://lucene.apache.org`
[5]Currently only sentences with exact matches are considered; however to increase the recall it is possible to extend the matches to related anchor texts extracted from Wikipedia

```

<token:token3408_2>  <conll:cpostag>  "NNP" ;
<conll:deprel>      "nn" ;
<conll:feats>       "_" ;
<conll:form>        "rudra" ;
<conll:head>        <token:token3408_4> ;
<conll:postag>      "NNP" ;
<conll:wordnumber>  "2" ;
<own:partOf>        <class:class3408> .

<token:token3408_3>  <conll:cpostag>  "NNP" ;
<conll:deprel>      "nn" ;
<conll:feats>       "_" ;
<conll:form>        "mohammad" ;
<conll:head>        <token:token3408_4> ;
<conll:postag>      "NNP" ;
<conll:wordnumber>  "3" ;
<own:partOf>        <class:class3408> .


<token:token3408_4>  <conll:cpostag>  "NNP" ;
<conll:deprel>      "nsubj" ;
<conll:feats>       "_" ;
<conll:form>        "shahidullah" ;
<conll:head>        <token:token3408_7> ;
<conll:postag>      "NNP" ;
<conll:wordnumber>  "4" ;
<own:partOf>        <class:class3408> .

<token:token3408_5>  <conll:cpostag>  "VBD" ;
<conll:deprel>      "cop" ;
<conll:feats>       "_" ;
<conll:form>        "was" ;
<conll:head>        <token:token3408_7> ;
<conll:postag>      "VBD" ;
<conll:wordnumber>  "5" ;
<own:partOf>        <class:class3408> .

```

Figure 4.3: Example snippet of the RDF representation of the parsed sentence `Poet Rudra Mohammad Shahidullah was the husband of writer Taslima Nasrin`.

with the pattern is created. To do so we match the output from the SPARQL query to templates, which were predefined by us and are based on *lemon*. For the example sentence above the noun copulative pattern is matched, which leads to the following lexicalization:

- Canonical form: `husband`
- Part of speech: noun
- Subcategorization frame: NounPP
  Arguments:
  - copulative subject a_1
  - prepositional object a_2 with marker `of`
- Semantic reference: *dbo:spouse*
  Arguments:
  - subject a_1
  - object a_2

Instantiated in a Java template, this lexicalization is presented in Section A.1 in Example A.2 on page 129.

The noun copulative pattern is visualized as SPARQL query in Figure 4.4 on the next page.

For the *dbo:spouse* example, 133 different candidate lexicalizations are extracted. Considering only those with a frequency greater than 1 leaves 36 lexicalizations, among them the expected lexicalizations such as `married to`, `wife of` and `husband of`. In addition to those direct lexicalizations, M-ATOLL also extracts closely related lexicalizations, such as `widow of`, `divorce`, `engaged to` and `spend honeymoon with`.

Datatype Property Example: *dbo:birthDate*

The datatype property *dbo:birthDate* connects persons with dates, for example the following pair with corresponding labels:

- ⟨ `Andrei_Tarkovsky` , `1932-04-04` ⟩
  Andrei Tarkovsky *1932 04 04*

Collecting all distinct entity pairs connected in this way, a list of 962,610 pairs is retrieved.

Extracting mentions of these pairs from the sentence index is more challenging for datatype properties, especially for date literals, as the Wikipedia corpus does not contain tokens like `1932-04-04`. As a first approximation of proper date matching, we consider only the year component of dates, i.e. searching sentences that contain the labels `Andrei Tarkovsky` and `1932`. In the future we plan to replace this heuristic with HeidelTime[6] by Strötgen and Gertz (2015). This leads to 188,287 sentences, including the following examples:

- *Sun Yat-sen was born on 12 November 1866.*

[6]See http://heideltime.ifi.uni-heidelberg.de/heideltime/

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?lemma ?prefix ?a1 ?a2 ?prep WHERE{
{ ?y <conll:cpostag> "NN" . }
        UNION
{?y <conll:cpostag> "NNS" . }
        UNION
{?y <conll:cpostag> "NNP" . }
?y <conll:form> ?lemma .
OPTIONAL{
        ?lemma_nn <conll:head> ?y.
        ?lemma_nn <conll:form> ?prefix.
        ?lemma_nn <conll:deprel> "nn".
}
?verb <conll:head> ?y .
?verb <conll:deprel> "cop" .
?e1 <conll:head> ?y .
?e1 <conll:deprel> "nsubj" .
?p <conll:head> ?y .
?p <conll:deprel> "prep" .
?p <conll:form> ?prep .
?e2 <conll:head> ?p .
?e2 <conll:deprel> "pobj" .
?e1 <own:senseArg> ?a1.
?e2 <own:senseArg> ?a2.
}
```

Figure 4.4: SPARQL representation of the noun copulative pattern as visualized in Section A.3.1

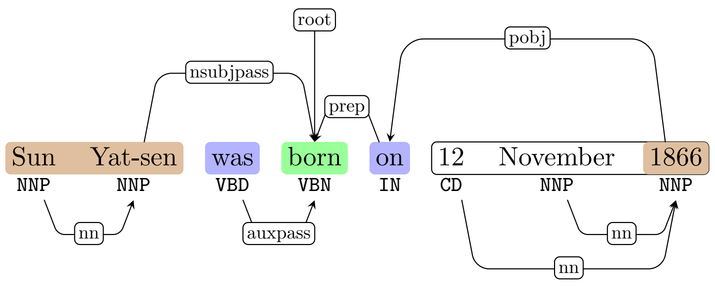

root
nsubjpass
prep
pobj
Sun Yat-sen was born on 12 November 1866
NNP NNP VBD VBN IN CD NNP NNP
nn
auxpass
nn
nn

Figure 4.5: Dependency tree for the sentence *Sun Yat-sen was born on 12 November 1866.*

- *David Edelstadt was born on 12 November 1986.*
- *Alessandro Volta, born in Como in 1745, invented the first true electrical battery, known as the voltaic pile.*

In Figure 4.5 the dependency graph for the first example sentence, `Sun Yat-sen was born on 12 November 1866`, is shown.

Matching the defined patterns with the RDF representation of all 188,287 sentences leads to 629 lexicalizations, 107 of which have a frequency of at least 2. The most prominent lexicalization is `born in`, which occurs 2,012 times, followed by `born on` with 41 occurrences. For the example sentence above, the predicative participle passive pattern matches, which leads to the lexicalization `born in`:

- Canonical form: `born`
- Part of speech: adjective
- Subcategorization frame: AdjectivePredicateFrame
  Arguments:
  - copulative subject a_1
  - prepositional object a_2 with marker `in`
- Semantic reference: *dbo:birthDate*
  Arguments:
  - subject a_1
  - object a_2

4.5 Language-specific lexicalization patterns

In this section we give examples for the linguistic patterns used in the approach described in Section 4.3, for all three languages: English, German, and Spanish. Overall we defined 7 patterns for English, 9 patterns for German, and 8 patterns for Spanish, covering noun, verb and adjective lexicalizations. A detailed visualization of the patterns together with an example parse is given in the appendix.

Our approach to creating relevant patterns is similar to the steps presented in Hearst (1992):

1. Start from a lexical relation, represented by a property in an ontology, e.g. *dbo:spouse.*
2. Create a list of verbalizations which represent this relation, e.g. `wife of`, `husband of`, `married to`.
3. Find sentences in the text corpus where these verbalizations occur.
4. Find commonalities between these sentences and extract the most common dependency pattern, which is still specific enough to represent the target relation.
5. Once the pattern is extracted, check if it also works for other relations of a similar type, e.g. for the property *dbo:successor.*

In the following we describe the patterns we found, following this procedure.

4.5.1 Noun patterns

The most frequent noun pattern, found in all three languages, are relational nouns with a prepositional object, such as `capital of` and `song by`. Such relational nouns can occur in two kind of grammatical relations: in copulative constructions as in (6), and in appositive constructions as in (7).

6. (a) *Brussels is the **capital of** Dutch-speaking Flanders and the **capital of** the European Union.*
   (b) *Kanazawa es la **capital de** la prefectura de Ishikawa.*
   (c) *Warschau ist seit 1596 die **Hauptstadt von** Polen.*
7. (a) *Arriving in Richmond, the **capital of** Virginia, on May 29, he was met by crowds at the railroad station.*
   (b) *Roma, **capital de** Italia, es la cuarta ciudad más poblada de la Unión Europea.*
   (c) *München, die **Hauptstadt von** Bayern, hat rund 1,5 Millionen Einwohner.*

In addition, relational nouns in German can be expressed with a possessive construct instead of a prepositional marker, both in copulative and in appositive constructions, as in the following examples:

8. (a) *Die Hafenstadt Amsterdam ist die **Hauptstadt** der Niederlande.*
   (b) *Warschau, die **Hauptstadt** Polens, liegt beidseitig am Strom der Weichsel.*

4.5.2 Verb patterns

The most obvious verb pattern for expressing relations is the transitive verb, as in the following examples:

9. (a) *Persson* ***developed*** *Minecraft and released it as a demo in 2009.*
   (b) *Persson* ***produjo*** *su primer juego a los 8 años.*
   (c) *Persson* ***entwickelte*** *Minecraft an seinem heimischen Rechner.*

Similarly, relations can be expressed as intransitive verbs with the object as part of a prepositional phrase, as in the following examples:

10. (a) *Nikola Tesla* ***died in*** *1943.*
    (b) *Benjamin Franklin* ***falleció en*** *Filadelfia.*
    (c) *Einstein* ***starb am*** *18. April 1955 im Alter von 76 Jahren.*

In order to capture different constructions in which transitive verbs occur, we also model passive structures, as in the following examples.

11. (a) *Instagram* ***was founded by*** *Kevin Systrom.*
    (b) *Instagram* ***wurde von*** *Kevin Systrom* ***gegründet.***

We did not introduce this pattern in Spanish because in these cases the dependency parser made a lot of errors; adding this pattern for Spanish would therefore result in more noise than useful lexicalizations.

The lexicalizations extracted from passive sentences are the same as those extracted from active sentences. To this end, the passive sentences are treated like their active counterparts (e.g. `Kevin Systrom founded Instagram`) by choosing the lemma of the verb participle as canonical form and by reversing the arguments (picking the syntactic subject, e.g. `Instagram`, as object, and the object marked by the preposition `by` as subject).

In German and Spanish we also find verbs that are inherently reflexive.

12. (a) *Die Golden Gate Bridge* ***befindet sich in*** *San Francisco.*
    (b) *Timbiriche* ***se desintegra en*** *1994.*

In Spanish, the same reflexive verbs also occur in passive constructions, as in the following example:

13. *En 1994, Timbiriche fué* ***desintegrado.***

4.5.3 Adjective patterns

Past participles of verbs, such as `born in` or `discovered in`, are treated as adjectival lexicalizations. The main reason is that the argument structure of their adjectival use corresponds directly to the argument structure of the lexicalized property, while the argument structure of their verbal counterpart in active constructions does not. For example, consider the property `discovered`, which relates a planet to the date it was discovered, as in the triple in (14a). The passive participle construction in (14b) comprises exactly those two arguments, while the active construction in (14c) is a construction with three arguments, one of which is not expressed in the RDF data.

14. (a) `<http://dbpedia.org/resource/Ceres_(dwarf_planet)>`
    `<http://dbpedia.org/ontology/discovered> "1801-01-01" .`

    (b) *Ceres was* ***discovered in*** *1801.*

    (c) *Giuseppe Piazzi* ***discovered*** *Ceres in 1801.*

Note that these cases differ from passive constructions with transitive verbs, like `Ceres was discovered by Piazzi`, where the active counterpart (`Piazzi discovered Ceres`) has the same argument structure.

Past participles in English and Spanish occur both in copulative constructions, as in the examples in (15), and in appositive constructions, as in the examples in (16).

15. (a) *Atri is* ***located in*** *Abruzzo, Italy.*

    (b) *Lancia es* ***fundado en*** *1906.*

    (c) *Armin Mueller-Stahl ist* ***verheiratet mit*** *Gabriele Scholz.*

16. (a) *Buena Vista Social Club is an album* ***published in*** *1997.*

    (b) *Buena Vista Social Club es un disco* ***publicado en*** *1997.*

In German the appositive constructions do not exist; instead they are expressed by means of a relative clause (e.g. `Buena Vista Social Club ist ein Album, das 1997 veröffentlicht wurde`).

4.6 Evaluation

We instantiated M-ATOLL for the DBpedia 2014 ontology and English, Spanish, and German Wikipedia dumps. In this section we present a detailed evaluation of the lexicalizations extracted by M-ATOLL. In the first part we evaluate the created lexical entries manually. After comparing the results to a goldstandard, we compare M-ATOLL to a baseline system that extracts surface patterns instead of more abstract grammatical patterns, in order to show that M-ATOLL can outperform other state-of-the-art systems. Additionally, we analyze the contribution of each pattern for each system to the overall lexicon. We conclude this section with a discussion and a manual evaluation of the two examples from Section 4.3: *dbo:spouse* and *dbo:birthDate*.

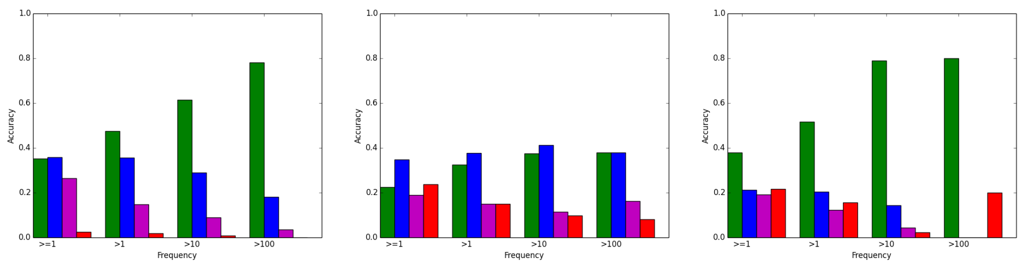
4.6.1 Manual Evaluation

For the first part of the evaluation we manually annotated a randomly chosen set of lexicalizations from all three languages. Three annotators were involved in this process, each annotator evaluated one language. For each pattern we separated the automatically extracted lexicalizations into four batches: The first batch contains all those that occurred only once, the second batch contains all those that occurred at least twice, the third batch contains all those that occurred more than ten times, and the fourth batch contains all those that occurred more than 100 times. From each batch we randomly chose 100 lexicalizations and annotated them along two dimensions: morphosyntactic correctness and semantic correctness. With respect to morphosyntactic correctness we distinguish two categories:

- The lexicalization is correct, i.e. the canonical form was correctly extracted and indeed belongs to the syntactic subcategorization frame that the pattern captures, the part-of-speech category is correct, possible prepositional markers are correct, and so on.
- The lexicalization contains an error. Examples would be missing verb particles (`kehren` instead of `zurückkehren` in German) or an incorrect matching of the pattern, for example extracting `belong since` from a sentence like `Hoogezand belongs to the principality Groningen since 1949` (instead of the correct `belong to`).

With respect to semantic correctness we distinguish three categories:

- The lexicalization constitutes a direct lexicalization of the property in question. An example is `wife of` as lexicalization of the property *dbo:spouse*.
- The lexicalization has a meaning that is semantically related to the property in question. An example would be `to divorce` as lexicalization for *dbo:spouse*. Often the decision when a lexicalization is semantically related is not clear-cut, but annotations within a language follow a consistent strategy to make this decision.
- The lexicalization is clearly not a lexicalization of the property in question, for example `residence of` as a lexicalization of the property *dbo:spouse*.



(a) English (b) German (c) Spanish

Figure 4.6: Overall results for the manual evaluation, with green representing morphologically and semantically correct entries, blue representing morphologically correct and semantically related entries, magenta representing semantically incorrect entries and red representing morphologically incorrect entries.

In Figure 4.6 we present the results of the manual evaluation for each language with respect to semantic correctness, showing that the quality of the extracted lexicon increases when considering only lexicalizations with a higher frequency. The x-axis separates the four frequency batches, the y-axis shows the percentage of lexicalizations that were annotated as morphologically and semantically correct, morphologically correct and semantically related, and morphologically or semantically incorrect. A clear trend is that with a higher frequency threshold, the number of correct lexicalizations increases, and the number of incorrect lexicalizations decreases. In the following we report the numbers for semantically and morphologically incorrect entries combined under the term `incorrect lexicalizations`. For English, the percent of correct lexicalizations increases from around 35% to almost 80%, while the percentage of incorrect lexicalizations decreases from around 27% to less than 5%. For Spanish, the percent of correct lexicalizations increases from around 38% to more than 80%, while the percentage of incorrect lexicalizations declines from around 42% to around 18%. For German, the results look slightly different: the increase of correct lexicalizations (from 22% to 38%) and decrease of incorrect lexicalizations (from around 43% to 25%) is less strong, while the decrease of the relative amount of related lexicalizations from around 43% to 25% is much higher. This observed difference could be an artifact of the fact that each of the authors annotated one language, possibly having a slightly different strategy for drawing the line between semantically related and semantically incorrect lexicalizations.

The general improvement of the extracted lexicon with higher frequency thresholds of course comes with a trade-off in terms of recall, which will be shown in the next section.

4.6.2 Automatic Evaluation

In this part we evaluate M-ATOLL with respect to a manually created English DBpedia lexicon by Unger et al. (2013). This lexicon contains 1,247 entries for 297 properties and 514 classes. As we do not have a similar goldstandard lexicon for German and Spanish in terms of coverage and quality, we restrict the automatic evaluation to English.

Note that the manually created goldstandard is not complete, i.e. does not contain lexicalizations of all properties. Therefore the evaluation is restricted to the subset covered by the goldstandard. The goldstandard we use was manually created independent of M-ATOLL and independent of the corpus M-ATOLL uses. It was not intended to be used as evaluation for M-ATOLL.

Moreover, M-ATOLL finds lexicalizations that count as correct but were missed during the creation of the manual lexicon. The reported results can therefore be considered a lower bound.

In order to gain insight into the potential of the approach M-ATOLL follows, we additionally compare the performance to the following baseline system, which mimics in a simplified way an ontology lexicalization approach as it is followed by, e.g., BOA by Gerber and Ngomo (2011, 2012):

> **Flat string:** The general workflow follows our corpus-based approach, as presented in Section 4.3, but instead of considering the dependency parse of sentences we only use the plain string. From this sentence string we extract all tokens that occur in between the entity mentions, keeping only nouns, adjectives and verbs. For instance, from the sentence `Barack Obama is the husband of Michelle Obama`, this baseline extracts `husband` as lexicalization, and from the sentence `Barack Obama is the first husband of Michelle Obama`, it would extract `first husband`.

Important to mention is that this baseline does not extract complete lexicalizations but only pairs of tokens and corresponding URIs. The evaluation therefore only considers canonical forms and references, and ignores all other lexical information. In the baseline also prepositions, for example as in `husband of`, and verbs in phrases, as in `is first husband of`, are removed. However, as said previously, we evaluate only on the lemma, and do not consider prepositions, therefore the results of the baseline is not biased in a negative way. Also the goldstandard does not contain phrases such as `is husband of`.

Evaluation Measures

For each property and class, we evaluate the automatically generated lexical entries by comparing them to the manually created lexical entries in terms of recall and precision at the lemma level. To this end, we determine how many of the goldstandard entries for a property are generated by our approach, where two entries count as the same lexicalization, if their lemma, part of speech and reference (URI of the property or class) coincide.

Thus macro and micro recall R_{macro} and R_{micro} for a set of properties P are defined as follows:

$$R_{macro}(P) = \frac{\sum_{p \in P} \frac{|entries_{auto}(p) \cap entries_{gold}(p)|}{|entries_{gold}(p)|}}{|P|} \tag{4.1}$$

Where $entries_{auto}(p)$ is the set of entries for the property p in the automatically constructed lexicon, while $entries_{gold}(p)$ is the set of entries for the property p in the manually constructed gold lexicon.

$$R_{micro}(P) = \frac{|entries_{auto} \cap entries_{gold}|}{|entries_{gold}|} \tag{4.2}$$

Similar to recall, macro and micro precision P_{macro} and P_{micro} are defined as:

$$P_{macro}(P) = \frac{\sum_{p \in P} \frac{|entries_{correct}(p)|}{|entries_{auto}(p)|}}{|P|} \tag{4.3}$$

Where $entries_{correct}(p)$ represents the set of correct lexical entries for a given property p.

$$P_{micro}(P) = \frac{|entries_{correct}|}{|entries_{auto}|} \tag{4.4}$$

The F-measure is calculated in both cases as the harmonic mean between recall and precision:

$$F_{macro}(P) = \frac{2 * R_{macro}(P) * P_{macro}(P)}{R_{macro}(P) + P_{macro}(P)} \tag{4.5}$$

$$F_{micro}(P) = \frac{2 * R_{micro}(P) * P_{micro}(P)}{R_{micro}(P) + P_{micro}(P)} \tag{4.6}$$

Evaluation of the Corpus-based Approach

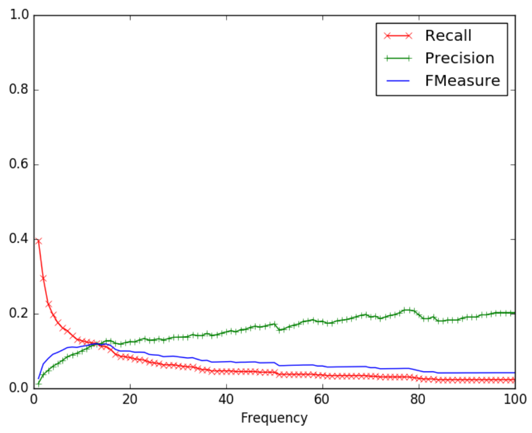
In Figure 4.7 on the following page we present the results for the corpus-based approach of M-ATOLL compared to the property entries of the goldstandard lexicon. For better comparison to the baseline later in this section, we evaluate once taking into account the part of speech and once ignoring it. All results are displayed as micro and macro recall, as well as the corresponding precision and F-measure values.

Overall, M-ATOLL finds around 40% of the entries of the goldstandard, with an F-measure slightly above 11% based on the micro measures, and around 38% based on the macro measures (both on recall and F-measure). Considering or ignoring the part of speech has almost no influence on the recall (around 3% on micro evaluation), which means that this is no serious source of errors.

In Figure 4.8 on page 63 we present results of the baseline system based on micro and macro measures, ignoring part-of-speech information. As expected, the baseline achieves a higher recall than M-ATOLL (cf. Figure 4.7 on the following page). However, it suffers greatly either in recall or precision, therefore M-ATOLL performs much better in terms of micro and macro F-measure.

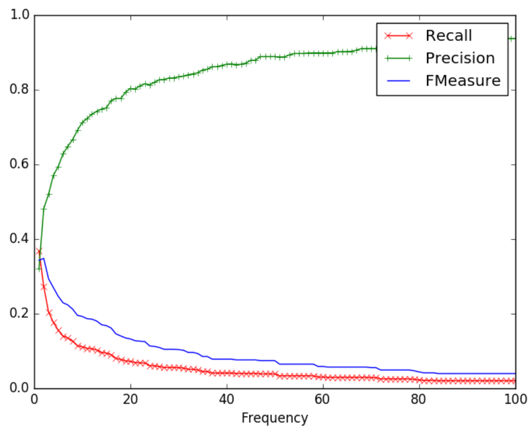
To put these results into perspective, we calculated the upper bound a system can achieve with respect to the goldstandard lexicon on the given corpus used as input. Starting with the text corpus, we generated a bag of words for each property by collecting all words appearing in sentences that contain mentions of the entities related by that property (only removing stop words). Then we check for each entry in the goldstandard lexicon whether the lemma appears in the relevant bag of words. This yields to an upper bound on recall of only 58%. That the achieved recall is lower is mainly due to the fact that the implemented patterns do not cover all grammatical constructions occurring in the text corpus, only the most common and reliable ones, as well as due to errors in the dependency parses.

Finally we reduced the goldstandard to those property entries that occurred in the bag of words collected for those properties, as these are the only entries that both M-ATOLL and the baseline system can possibly find. The results with respect to this reduced goldstandard are presented in Figure 4.9 on page 64. They show that neither the corpus-based approach of M-ATOLL nor the baseline system is able to extract all lexical entries. While the baseline can



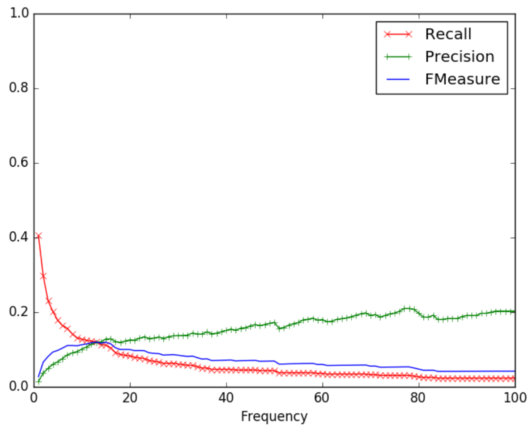
(a)

Micro measures (with POS)



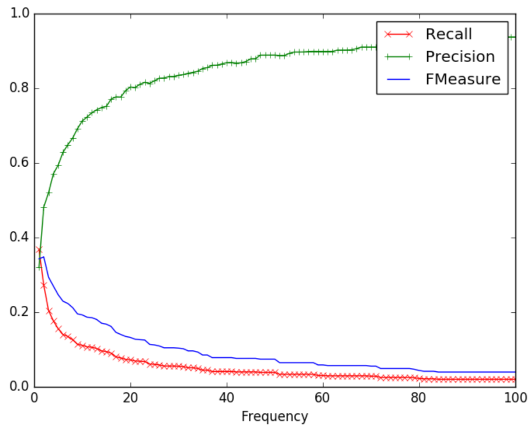
(b)

Macro measures (with POS)



(c)

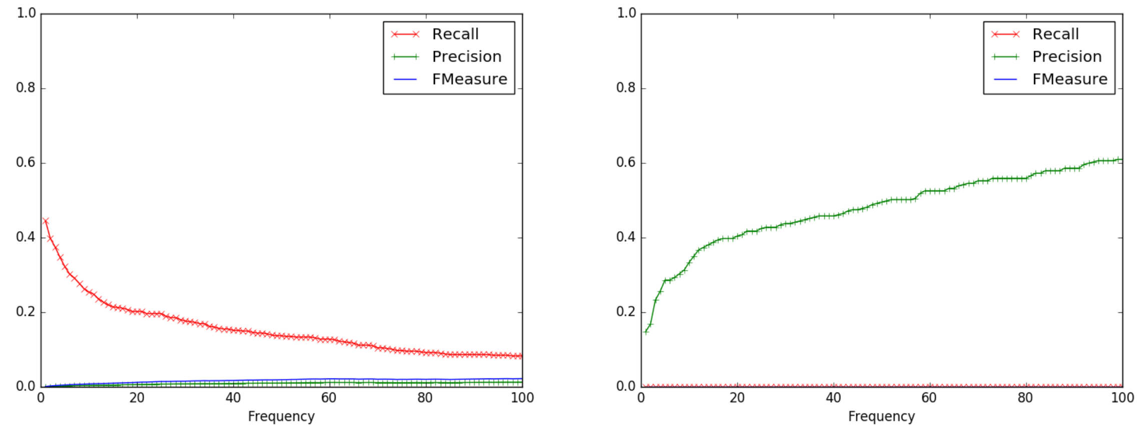
Micro measures (without POS)



(d)

Macro measures (without POS)

Figure 4.7: Results of the corpus-based approach of M-ATOLL compared to the property entries from the gold-standard lexicon, depending on the frequency threshold, once considering the part-of-speech information and once ignoring it.



(a) Micro (without POS)

(b) Macro (without POS)

Figure 4.8: Results for the flat-string baseline compared to the property entries from the goldstandard lexicon.

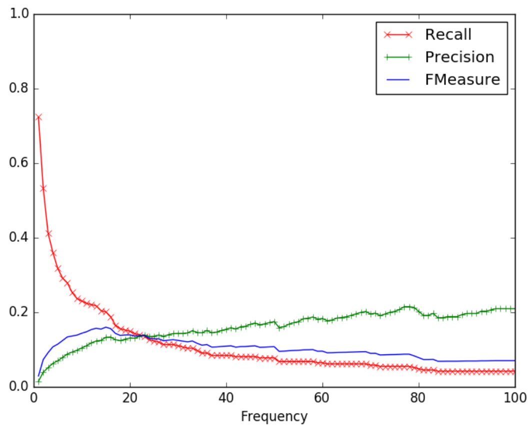
reach a micro recall of 80%, generally it suffers greatly either in recall or precision. M-ATOLL therefore clearly outperforms it in terms of F-measure, although it is able to find only around 72% of all lexicalizations. This shows that the lexicalization task benefits from linguistically grounded grammatical patterns.

For an better overview we present in Figure 4.10 on page 65 a direct comparison of the recall and F-measures from the corpus-based approach of M-ATOLL as well as the baseline.

Overall, the evaluation results show that accepting only lexicalizations above a certain frequency threshold has a big impact on the quality of the lexicon, exhibiting the expected trade-off between precision and recall. In fact, requiring a frequency greater than 1 often already reduces the number of incorrect lexicalizations considerably. Where the optimal threshold lies depends mainly on the application for which one wants to use the lexicon. The higher the threshold, the less lexicalizations are produced, but those that are produced are generally of high quality and are mostly direct lexicalizations, whereas a lower threshold leads to more lexicalizations, including semantically related ones (such as `to divorce` and `widow of` for *dbo:spouse*).

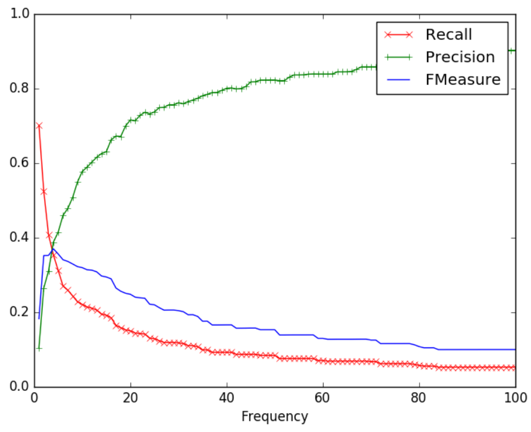
Evaluation of the Label-based Approach

In Table 4.1 on page 66 we present the results for the label-based approach in terms of micro and macro recall. In this case we do not consider precision, as all lexicalizations are created exactly once, therefore we currently have no strategy to assess the quality of those lexicalizations in order to filter out incorrect ones (something that can be done based on frequency in the corpus-based approach). We present results on the classes, on the properties, and on both together. In general, considering the part-of-speech information makes a difference for micro recall, especially for properties, but not for macro recall. The label-based approach reaches a macro recall of 12%



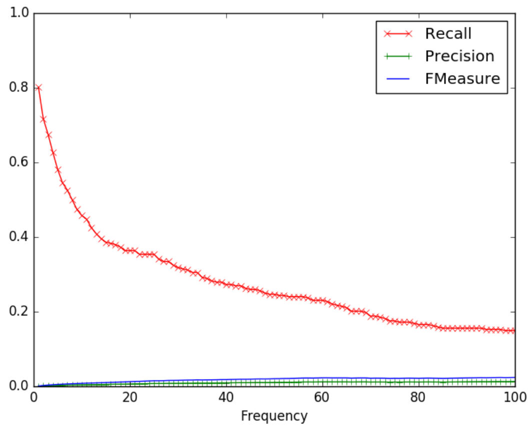
(a)

Corpus-based approach, micro measures



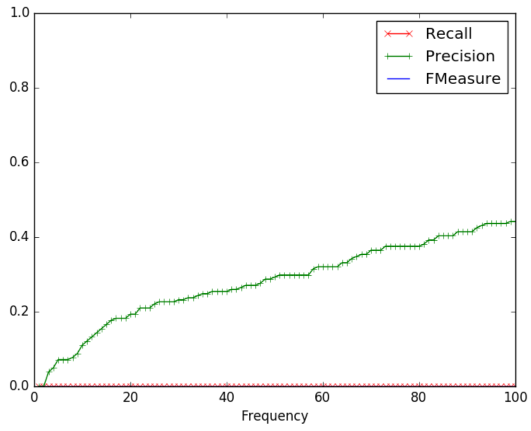
(b)

Corpus-based approach, macro measures



(c)

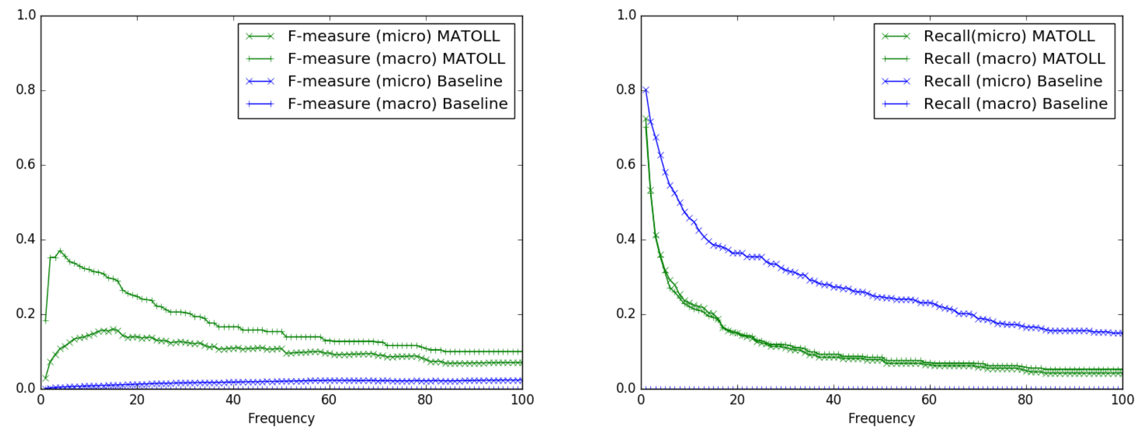
Baseline, micro measures



(d)

Baseline, macro measures

Figure 4.9: Results of the corpus-based approach as well as the flat-string baseline compared to the property entries from the reduced goldstandard, ignoring part-of-speech information.



(a) F-measure

(b) Recall

Figure 4.10: Comparison of results of the corpus-based approach and the flat-string baseline compared to the property entries from the reduced goldstandard, ignoring part-of-speech information.

for properties, of 46% for classes, and of 34% when considering both.

Finally, when combining the lexicalizations found by the label-based approach with those found by the corpus-based approach, M-ATOLL achieves a macro recall of 46% on the non-reduced goldstandard lexicon. As also observed with earlier versions of the system (see Walter et al. (2014a)), the label- and corpus-based approaches complement each other, so that a combination of both yields better results than either of them separately.

4.6.3 Influence of Each Pattern

In the following we analyze the contribution of each pattern to the overall lexicon. In doing so, we only consider the contribution in terms or recall, i.e. the number of lexicalizations found by these patterns, we do not assess the correctness of those lexicalizations.

For each pattern, we present the absolute number of how often it was used to generate a lexicalization, and its relative influence (a value between 0 and 1 specifying the portion of lexicalizations stemming from this pattern) – for English see Table 4.2 on the following page, for German see Table 4.3 on page 67, and for Spanish see Table 4.4 on page 67. The last three lines of Table 4.4 on page 67 are reported as 0.00 because we only consider two digits after the comma. Also, the absolute frequencies for German and Spanish are generally lower than those for English, as the Wikipedia dumps used as text corpus are smaller.

For English and Spanish the most influential pattern is the *Noun PP copulative* pattern (visualized in Sections A.3.1 on page 131 and A.3.3 on page 134), which matches sentences like `Alice is the wife of Bob`. For German the most influential pattern is the intransitive verb pattern (visualized in Figure A.3.2 on page 134). For a depiction of all patterns please refer to the appendix

	With POS		Without POS	
	R_{micro}	R_{macro}	R_{micro}	R_{macro}
Label-based approach				
Properties	0.12	0.12	0.12	0.12
Classes	0.43	0.46	0.43	0.46
Both	0.28	0.34	0.29	0.34
Combination of label- and corpus-based approach				
Both	0.45	0.46	0.69	0.46

Table 4.1: Recall of the label-based approach on properties, classes, and both, and of a combination of label- and corpus-based approaches on classes and properties.

Pattern	Absolute frequency	Relative influence
Noun with PP (copulative)	44,698	0.30
Intransitive verb with PP	34,242	0.23
Predicative participle (passive)	29,736	0.20
Noun with PP (appositive)	19,152	0.13
Transitive verb	9,174	0.06
Predicative participle (copulative)	4,395	0.03
Noun with possessive	3,188	0.02
Transitive verb (passive)	2,822	0.02

Table 4.2: Absolute frequency and relative influence of each English pattern.

in Section A.3.

4.6.4 Discussion

Finally we want to look closer at the lexicalizations found for the two running examples in Section 4.3: *dbo:spouse* and *dbo:birthDate*.

For the object property *dbo:spouse*, the lexicalizations from both the goldstandard and the approaches implemented by M-ATOLL are given in Table 4.5 on page 68, where lexicalizations found by one of the approaches are marked with a + and lexicalizations not found are marked with a -. For the corpus-based approach, we display only lexicalizations with a frequency of at least 10, and we include the actual frequency in brackets.

The combination of label- and corpus-based approach finds all lexicalizations that are also present in the goldstandard lexicon, and additional ones, such as `married to` and `consort of`,

Pattern	Absolute frequency	Relative influence
Intransitive verb with PP	10,269	0.27
Noun with PP (copulative)	8,351	0.22
Noun with PP (appositive)	6,483	0.17
Noun with possessive	6,035	0.16
Transitive verb	4,135	0.11
Transitive verb (passive)	886	0.02
Noun with possessive (appositive)	769	0.02
Predicative adjective	716	0.02
Reflexive transitive verb with PP	468	0.01

Table 4.3: Absolute frequency and relative influence of each German pattern.

Pattern	Absolute frequency	Relative influence
Noun with PP (copulative)	1,977	0.37
Intransitive verb with PP	1,483	0.28
Noun with PP (appositive)	668	0.12
Predicative participle (copulative)	505	0.09
Transitive verb	288	0.05
Noun with PP (copulative, with hop)	282	0.05
Reflexive transitive verb with PP	97	0.02
Predicative participle (passive)	22	0.00
Transitive verb (passive)	8	0.00
Transitive verb (reciprocal)	2	0.00

Table 4.4: Absolute frequency and relative influence of each Spanish pattern.

Lexicalization	Label-based	Corpus-based	goldstandard
`wife of`	-	+ (90)	+
`husband of`	-	+ (10)	+
`to marry`	-	+ (97)	+
`spouse`	+	-	+
`married to`	-	+ (82)	-
`widow of`	-	+ (17)	-
`consort of`	-	+ (12)	-
`to meet`	-	+ (10)	-
`married person`	+	-	-
`better half`	+	-	-
`partner`	+	-	-

Table 4.5: Lexicalizations extracted for the property *dbo:spouse*.

Lexicalization	Label-based	Corpus-based	goldstandard
`born in`	-	+ (2,012)	+
`born on`	-	+ (41)	-
`to die in`	-	+ (27)	-
`to bear`	-	+ (24)	-
`birth date`	+	-	-

Table 4.6: Lexicalizations extracted for the property *dbo:birthDate*.

and the not direct but related lexicalizations `widow of` and `meet`, as well as the synonym terms `married person`, `better half` and `partner` extracted from WordNet.

For the datatype property *dbo:birthDate*, the lexicalizations from both the goldstandard and the approaches implemented by M-ATOLL are given in Table 4.6. The goldstandard lexicon only contains the entry `born in`, which was also extracted by M-ATOLL (with a very high frequency). In addition, the corpus-based approach extracts `born on`, although the semantic distinction between using `in` (for years) and `on` (for dates) is currently lost. The lexicalization `birth date`, generated by the label-based approach, is not contained in the goldstandard, although it is a valid lexicalization. This kind of mismatches between M-ATOLL's results and the goldstandard lexicon are simply due to the fact that the latter was constructed manually and therefore is prone to lower recall (while having very high precision).

The lexicalization `to bear` is extracted from sentences containing `born`, whose dependency structure matches the passive transitive verb pattern. An example is `Richard Kell, Irish` `poet, born 1927`, which leads to the active verb version `to bear` with `Richard Kell` as direct object and `1927` as subject (erroneously). The lexicalization to `die in`, on the other hand, is syntactically correct but verbalizes exactly the opposite of the intended meaning. The problem here is rather one of entity matching. In a sentence like `Gilbert died in 1603`, for example, `Gilbert` would match a range of entities having the surname `Gilbert`, so it happens that someones birth date is mixed up with someone else's death date. Currently the entity matching step does not involve any entity disambiguation, which would be needed in order to solve this problem.

4.7 Conclusion

In this chapter we presented the core approach of the framework M-ATOLL, which, given an ontology, a corresponding knowledge base, and a text corpus, generates lexicalizations for ontology elements (classes and properties) in multiple languages. As proof of concept we implemented M-ATOLL for English, German and Spanish, and applied it to DBpedia, using Wikipedia as text corpus. M-ATOLL relies on a set of predefined grammatical patterns that are defined declaratively and thus are easy to extend and adapt. We were able to show that M-ATOLL produces good results for all three languages, being particularly useful when considering a semi-automatic scenario, in which automatically extracted lexicalization candidates are checked and, if necessary, corrected or discarded by a human engineer.

Chapter 5

Analysis & Discussion

In this chapter we present additional and detailed analyses of M-ATOLL's results, in order to get further insights in to the performance and limitations of the approach.

5.1 Analysis of different filtering strategies

In Section 4.6 we presented an automatic evaluation of M-ATOLL compared to a publicly available goldstandard. This evaluation was done using different frequency thresholds as a filter to decide whether to add an entry to the final lexicon. The results from Section 4.6 are shown again in this section, this time in tabular form in Table 5.1. This table shows how the recall results drop when adding entries with a higher frequency threshold. Already the drop from allowing all entries to be added to the final lexicon (frequency 1) and allowing only entries occurring at least twice (frequency 2) is more then 10%. Accepting only entries in the final lexicon which occurred five or more times furthermore halves the recall.

	Micro Recall		Macro Recall	
Frequency	with POS	without POS	with POS	without POS
1	0.40	0.41	0.37	0.37
2	0.29	0.30	0.27	0.27
3	0.23	0.23	0.20	0.20
4	0.20	0.20	0.18	0.18
5	0.18	0.18	0.16	0.16
6	0.16	0.16	0.14	0.14
7	0.15	0.16	0.13	0.13
8	0.14	0.14	0.13	0.13
9	0.13	0.13	0.11	0.11
10	0.13	0.13	0.11	0.11

Table 5.1: Recall of the corpus-based approach of M-ATOLL compared to the property entries from the goldstandard lexicon, depending on the frequency threshold, once considering the part-of-speech ("with POS") information and once ignoring it ("without POS").

However, the filtering strategy based on frequency is not the optimal solution, because it does not consider the amount of sentences which were available for a property, or the amount of overall extracted lexical entries for a given property. If for one property only one entry is extracted, it is more likely to be relevant than an entry for a property for which 10.000 entries were extracted. In the following, we try to find a better filtering strategy in order to minimize the drop in recall. We test three different strategies:

1. Lemma-based filtering strategy
2. Property-based filtering strategy
3. Machine-learning-based filtering strategy

Each of them will be explained in more detail below. The strategies are applied globally on the whole lexicon after it was created by M-ATOLL.

Entry	Lexicalization	Frequency	Property
e_1	`wife`	10	*dbo:spouse*
e_2	`husband`	5	*dbo:spouse*
e_3	`wife`	4	*dbo:relative*

Table 5.2: Example lexicon *Lex* to illustrate the *lemma_based_score* presented in Equation 5.1

5.1.1 Lemma-based filtering strategy

The idea behind the lemma-based filtering strategy is the following: When a lexical entry is extracted, it contains a canonical form. If this canonical form occurs only for this specific property, it is very likely to be the correct lexicalization. On the other hand if the canonical form also occurs for other properties, this likelihood decreases.

Formally we define the lemma-based filtering strategy as shown in Equation 5.1.

$$lemma\_based\_score(e, Lex) = \frac{freq(e)}{\sum_{e' \in \text{ Lex s. t. lemma(e')=lemma(e)}} freq(e')} \tag{5.1}$$

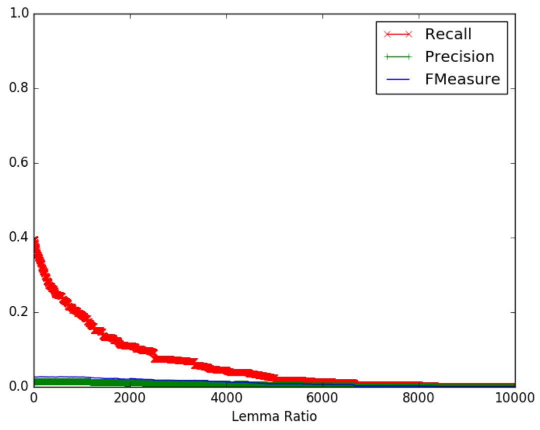
where $freq(e)$ represents the number of times that the lexical entry e has been extracted by our patterns from the lexicon Lex.

As an example consider the example lexicon Lex in Table 5.2. The scores for the three example entries are as follows:

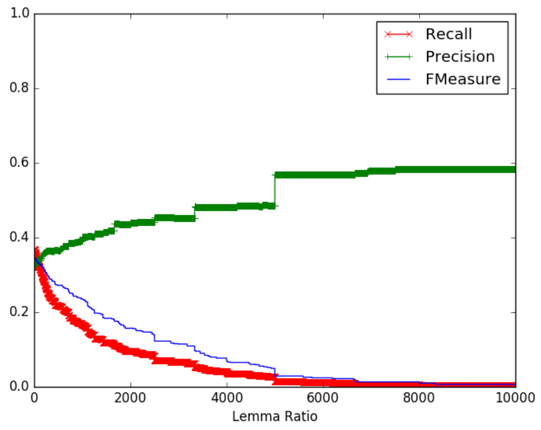
$lemma\_based\_score(\text{e\_1}, Lex) = \frac{10}{14} = 0.71$
$lemma\_based\_score(\text{e\_2}, Lex) = \frac{5}{5} = 1.0$
$lemma\_based\_score(\text{e\_3}, Lex) = \frac{4}{14} = 0.28$

The results are visualized in Figure 5.1 on the following page, where we present not only the recall, but also the precision and F-measure values. The threshold, deciding when to add an entry to the final lexicon or not, can be chosen in very small steps. Initial experiments showed that using steps of 0.1, this filtering strategy loses at its best recall around 5% compared to the best recall of the frequency-based filtering strategy. With a smaller step size of 0.01 we avoid this problem. For a better readability we scaled the values in Figure 5.1 on the next page between 1 and 10,000, this means that a 0.001 is represented as 1 in Figure 5.1 on the following page and a value of 0.01 is represented as 10.

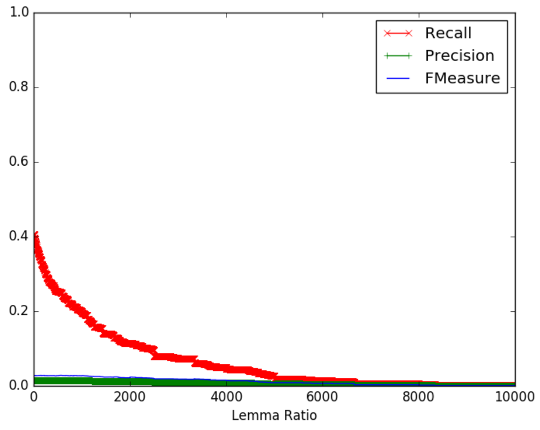
However, as visualized in Figure 5.1 on the next page, no significant improvement over the standard frequency-based strategy could be observed in terms of F-measure. The only advantage this strategy has over the frequency-based strategy is that a more fine grade selection, using very small step sizes, is possible. This could be used for example to optimize M-ATOLL for different use cases.



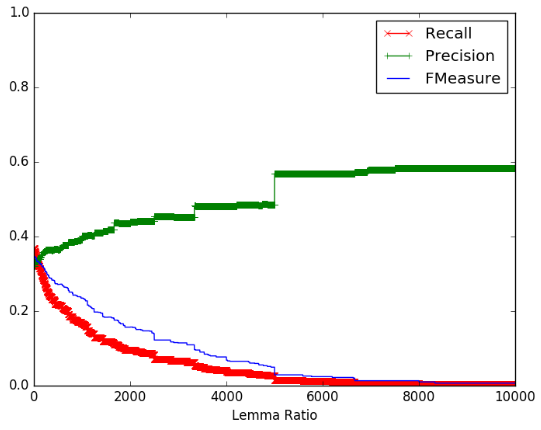
(a)
Micro measures (with POS)



(b)
Macro measures (with POS)



(c)
Micro measures (without POS)



(d)
Macro measures (without POS)

Figure 5.1: Results of the corpus-based approach of M-ATOLL compared to the property entries from the goldstandard lexicon, using the lemma-based filtering strategy, once considering the part-of-speech information and once ignoring it.

5.1.2 Property-based filtering strategy

After observing that the previously presented filtering strategy did not work as well as we expected we also consider a different strategy focusing not only on lemmas but on references. With this strategy we assume that not each entry has the same influence in the overall lexicon. The idea is as presented earlier: If for one property only 1 entry is extracted, it is more likely to be the correct entry than for a property with 10,000 extracted entries. This idea is formalized in Equation 5.2.

$$property\_based\_score(e, Lex) = \frac{freq(e)}{\sum_{e' \,\in\, \text{Lex s. t. reference(e')=reference(e)}} freq(e')} \tag{5.2}$$

Entry	Lexicalization	Frequency	Property
e_1	`married to`	12	*dbo:spouse*
e_2	`husband`	15	*dbo:spouse*
e_3	`written` by	4	*dbo:author*

Table 5.3: Example lexicon *Lex* to illustrate the *property_based_score* presented in Equation 5.2 on the preceding page

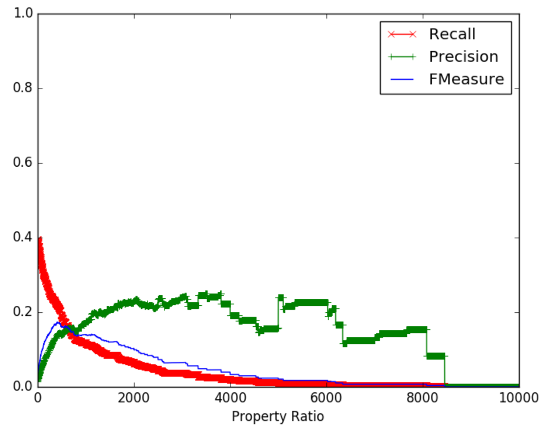
where $freq(e)$ represents the number of times that the lexical entry e has been extracted by our patterns from the lexicon Lex. As an example consider the example lexicon Lex in Table 5.3. The scores for the three example entries are as follows:

$property\_based\_score(\text{e\_1}, Lex) = \frac{12}{27} = 0.44$

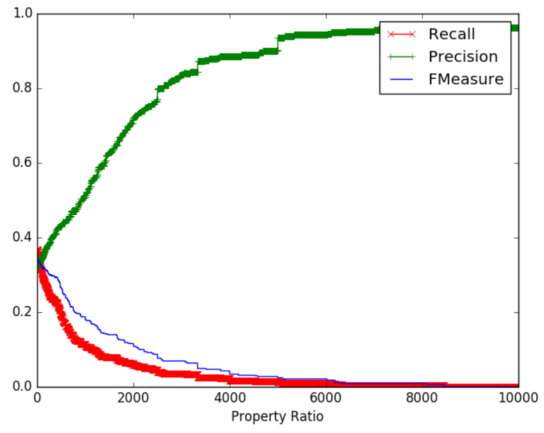
$property\_based\_score(\text{e\_2}, Lex) = \frac{15}{27} = 0.55$

$property\_based\_score(\text{e\_3}, Lex) = \frac{4}{4} = 1.0$

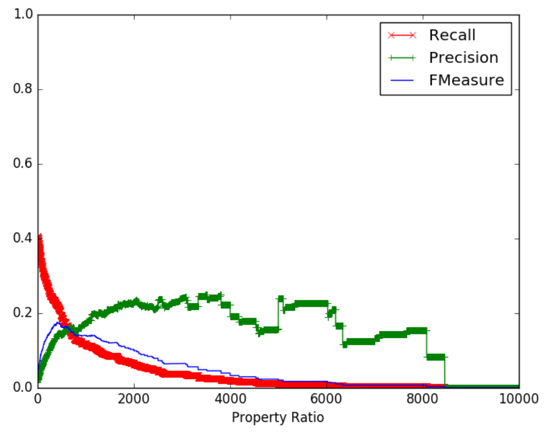
The results for this strategy are presented in Figure 5.2 on the following page.



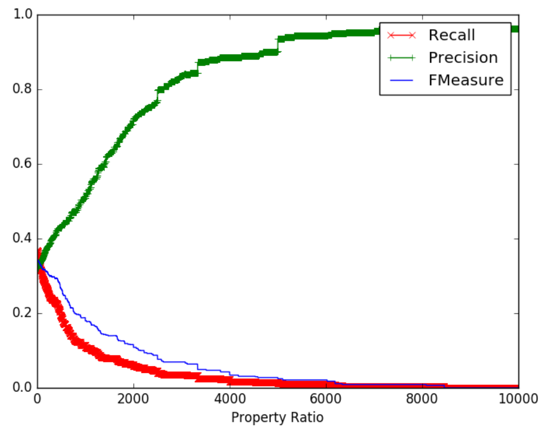
(a)
Micro measures (with POS)



(b)
Macro measures (with POS)



(c)
Micro measures (without POS)



(d)
Macro measures (without POS)

Figure 5.2: Results of the corpus-based approach of M-ATOLL compared to the property entries from the gold-standard lexicon, using the property-based filtering strategy, once considering the part-of-speech information and once ignoring it.

The results show clearly an improvement by around 16% on the micro evaluation compared to the results presented in Figure 5.1 on page 74. Compared to the original results presented in Figure 4.7 on page 62 the results improved by around 5% on F-measure on the micro-based evaluation. With the macro-based evaluation, the results are comparable to the original evaluation. Overall the results show that this strategy works better than the lemma-based filtering strategy and the frequency-based strategy, considering the micro-based evaluation.

5.1.3 Machine-learning-based filtering strategy

Finally, we consider a machine-learning-based filtering strategy. While the previous two strategies only covered one characteristic each, with the new strategy we hope to be able to include different characteristics to improve the performance presented above. To do so, we designed 12 different types of features, which are presented in the following:

- **Normalized Frequency by Lemma (NFL)**
  is as defined in Equation 5.1 on page 73 and represents the lemma-based filtering strategy.

- **Normalized Frequency by Property (NFP)**
  is as defined in Equation 5.2 on page 74 and represents the property-based filtering strategy.

- **Part-of-Speech Pattern (PP)**
  For a lexical entry we store the part-of-speech information. For each part-of-speech (e.g. adjective, verb, noun) which occurs in the lexicon we construct a Boolean feature recording whether the part-of-speech is used in the given entry or not.

- **SPARQL Pattern (SP)**
  For each lexical entry we store the information with which SPARQL pattern the entry was constructed. We construct a Boolean feature for each SPARQL pattern which was used in the overall lexicon, recording whether the SPARQL pattern was used in the generated entry or not.

- **Has Preposition (HP)**
  This feature is a Boolean feature and encodes if the given lexical entry has a preposition or not.

- **Preposition (PREP)**
  We collect each possible preposition from the overall lexicon and encode as Boolean feature which of the preposition occurred in the given lexical entry.

- **Frequency (FREQ)**
  This feature represents the frequency of the given lexical entry. However, we do not use the value itself in our machine learning approach, but rather encode the frequency ($freq$) as Boolean vector of different frequency bins in particular:

  $freq \leq 2$

  $freq \leq 5$

  $freq \leq 10$

  $freq \leq 20$

  $freq \leq 100$

  $freq \geq 100$

- **Type (TYPE)**
  This feature encodes as Boolean feature whether the property of the given lexical entry is a data property or an object property.

- **Domain Range (DR)**
  For this feature, the domains and ranges for all properties in the lexicon are extracted. The feature itself encodes, as Boolean feature, which domain and range is given for the property of the lexical entry.
- **Argument Mapping (ARG)**
  This feature encodes, again as Boolean feature, whether the subject of the property in the given lexical entry is also the syntactic subject in the sentence.
- **Lemma (LM)**
  For this feature, we extracted all lemmas from all entries in the lexicon and encode, as Boolean feature, which of the lemmas matches the lemma of the given lexical entry.
- **Bag of words (BOW)**
  For this feature, we build a set of all words from all sentences used to create the lexical entries. We removed all stopwords and then created a Boolean feature for each of the words, encoding which of the words appeared in the sentences for the given lexical entry.

To evaluate the influence of each feature, we choose four different classifiers: Random Forest, Ada Boost, Perceptron and Support Vector Classification(SVC), and evaluated their performance. All classifiers were used with the standard configuration provided by *scikit-learn*[1] and were not optimized for this use case. We used a standard 10-fold cross validation task; each fold was created by *scikit-learn*. Our dataset is balanced and contains 633 correct and 633 wrong entries, according to the evaluation with the goldstandard presented in Section 4.6.

The results of this experiment are presented in Table 5.4 on the next page. The feature with the best performance is *NFP* with an accuracy of 0.88 with the Random Forest classifier. The feature with the lowest impact is the *TYPE* feature with an accuracy of 0.35 in combination with the SVC classifier. Additionally, Table 5.4 on the facing page provides the average result for each feature over all four classifiers, showing that overall the feature *DR* with an average accuracy of 0.79 is the best. Overall, on average, the *TYPE* feature performs worse.

To evaluate the influence of each feature in combination with other features, we build the powerset from the set of all features, leading to 4,095 unique feature combinations. We tested these combinations with the four different classifiers. The best results are presented in Table 5.5 on the next page, ranked by decreasing accuracy: the best results were achieved by the Random Forest classifier with an accuracy of 0.92 followed by the Ada Boost classifier with an accuracy of 0.91; the third best classifier was the Perceptron classifier with an accuracy of 0.90 followed by the SVC classifier with an accuracy of 0.89. Table 5.6 on the facing page shows the top 10 feature combinations for the Random Forest. For the other classifiers, the top 10 feature combinations are presented in the Appendix, see Section A.2. The feature combination with the best result contains 7 out of 12 features, namely NFP, FREQ, HP, DR, LM, NFL and SP. All the presented feature combinations performed with an accuracy of over 90%.

Overall, the results show that our features are well defined enough in order to train a good classifier. But do they also improve the overall lexicon?

[1] http://scikit-learn.org/stable/index.html

	Random Forest	Ada Boost	Perceptron	SVC	$\overline{x}$
NFL	0.73	0.64	0.72	0.63	0.68
NFP	**0.88**	0.59	0.81	0.64	0.73
PP	0.51	0.54	0.51	0.51	0.51
SP	0.55	0.50	0.57	0.57	0.54
HP	0.52	0.50	0.52	0.52	0.51
PREP	0.62	0.51	0.61	0.61	0.58
FREQ	0.65	0.54	0.65	0.65	0.62
TYPE	0.40	**0.47**	**0.40**	**0.35**	0.40
DR	0.79	**0.78**	**0.81**	**0.81**	0.79
ARG	**0.40**	0.48	**0.40**	0.40	0.42
LM	0.74	0.71	0.71	0.62	0.69
BOW	0.68	0.74	0.67	0.73	0.70

Table 5.4: Accuracy for each single feature for all classifiers.

Classifier	Accuracy (best)
Random Forest	0.92
Ada Boost	0.91
Perceptron	0.90
SVC	0.89

Table 5.5: Accuracy for all classifiers.

Feature Combination	Accuracy
NFP, FREQ, HP, DR, LM, NFL, SP	0.9196
NFP, FREQ, PP, PREP, DR, LM, NFL	0.9156
NFP, PP, DR, LM, NFL, TYPE	0.9133
NFP, DR, LM, NFL	0.9133
NFP, FREQ, PP, PREP, DR, LM, NFL, TYPE, ARG	0.9118
NFP, FREQ, PP, PREP, DR, LM, NFL, ARG	0.9094
NFP, PP, DR, LM, NFL	0.9087
NFP, FREQ, PP, DR, LM, NFL, TYPE, ARG	0.9086
NFP, HP, PP, PREP, DR, LM, NFL, ARG, SP	0.9070
NFP, HP, PREP, DR, LM, NFL	0.9063

Table 5.6: Accuracy for top 10 feature combinations for the Random Forest Classifier.

Evaluation	Recall	Precision	F-Measure
Micro (with POS)	0.394	0.021	0.040
Micro (without POS)	0.400	0.021	0.041
Macro (with POS)	0.368	0.426	0.395
Macro (without POS)	0.368	0.426	0.395

Table 5.7: Results after applying the trained RandomForest model to the lexicon created by M-ATOLL.

To evaluate this question we took our complete dataset of 1266 annotated entries (633 correct and 633 wrong entries) and the Random Forest classifier with the best feature combination to train a model which was then applied to all lexical entries extracted by M-ATOLL. The input contained all 22,143 lexical entries, after applying the model the number was reduced to 13,719 lexical entries. Again we evaluated both micro and macro measures and also considered the part-of-speech (POS). The results are presented in Table 5.7.

The highest recall on the micro-based evaluation with a value of 0.4, meaning that non of the correct entries are missing. Remember that the highest recall we had initially (see beginning of this section) was also 0.4. Evaluating with the macro-based evaluation a F-measure of 0.395 was achieved. Overall, the results evaluated with the micro-based evaluation, considering the F-Measure, performs not as good as the original evaluation presented in Figure 4.7 on page 62. The results of the macro-based evaluation stayed the same. One reason that the results on the micro-based evaluation performs worse could be the fact that the wrong feature combination was chosen. We selected the same feature combination that proofed to be the best combination on a 10-fold cross validation task. In order to analysis this, we again took the powerset of all features to evaluate on the whole lexicon. However, as this is a very performance insensitive analysis, we reduced the combinations to combinations of those with four or less features in it. Because of the results presented in Table 5.6 on the previous page, where one of the top feature combinations contained only four features, but performed with an accuracy of 0.0036 less than the best combination, we assume that we still get similar results. This reduction still led to 793 unique feature combinations.

Evaluating with the micro-based evaluation the feature combination with the best result is using POS, LM and BOW. This combination leads to a F-measure of 21% (which is almost 10% higher than in the original evaluation) and a recall of 38%. Using the macro-based evaluation we achieved F-measure of 45%, improving this value by around 7%, with a recall of 36%. Both evaluations were done with consideration of the POS tag. For this evaluation we ignored the fact that our dataset we used to train the model is also contained in the data we used to actually evaluate the model. The reason is that the data to train the RandomForest model contains only a fraction of the whole dataset. The goal of this section was to give a proof-of-concept for our machine-learning-based filtering strategy; a more informative evaluation has to be done with a larger training and test dataset, with a clear separation between both datasets.

width	casualties	accommodate
format	wide	sublime
displacement	beam	reviewed
runtime	density	magnitude
long	length	weight
fuse	revenue	boil
tall	abbreviate	weigh

Table 5.8: 21 terms that are missed by M-ATOLL but occur in the corpus.

Overall, we presented an approach that is able to almost double the F-measure results (at least on the micro-based evaluation), without a major loss on recall.

5.2 Analysis of the corpus as well as the patterns

In this section we analyze the English corpus in more detail and try to explain the fact that our baseline, as introduced in Section 4.6.2, finds more lexicalizations than M-ATOLL. The general workflow of this baseline follows our corpus-based approach but instead of considering the dependency parse of sentences only plain strings are used. From this sentence string all tokens that occur in between the entity mentions are extracted, keeping only nouns, adjectives and verbs.

For the following analysis we use the same corpus, with marked entity mentions, and goldstandard as presented in Section 4.6. The goldstandard, reduced only to property entries, contains 605 entries with 366 unique terms (one term can be the lexicalization of multiple properties). The goldstandard we use was manually created independent of M-ATOLL and was not meant to be used as evaluation for M-ATOLL. From these 366 terms M-ATOLL does not find 100 terms. However, 79 of these 100 terms do not occur in our corpus and are neither extractable with our label-based approach. The other 21 terms do at least appear in the corpus. These terms are presented in Table 5.8.

But before starting a more detailed analysis on the reasons why these 21 terms were not extracted by M-ATOLL, we provide a closer look at the shortest path between two entity mentions in the corpus. As explained in the previous chapter, we handcrafted grammatical patterns in order to extract lexicalizations describing the relation between two entities. Often these hand-crafted patterns are equivalent to a shortest path between two given entity mentions. It is obvious that one of the reasons why these 21 terms were not extracted is that we missed a grammatical pattern.

To address this weakness, we extracted all shortest paths between the entities in the given corpus.

Overall, we were able to extract 3,028,019 shortest paths from our corpus, leading to 172,816 unique patterns. The top 10 patterns are presented in Table 5.9 on the following page. Besides the pattern itself, in this table we present the absolute frequency, the influence normalized over all

Shortest Path	Frequency	%	Sentence
appos	133,640	4.41	`The date set for the uprising was August 11, 1680`
nn prep pobj	42,622	1.41	`Joseph College in Hartford, Connecticut, in 1950`
pobj prep prep pobj	40,236	1.33	`The name was published by George Vancouver in 1798`
nn pobj prep prep pobj	40,312	1.33	`His third marriage was to Charlotte Savage in 1985`
nn	39,913	1.32	`The two ground stations were built by EADS Astrium`
nsubj prep pobj	37,750	1.25	`In 1972, at the age of 88 , Giovanni Bertone died`
pobj	37,241	1.23	`Frederick Charles Ferdinand died childless in 1809`
num	36,799	1.22	`He died at Pascagoula on August 5, 1853, aged 38`
conj	34,749	1.15	`Ford now encompasses two brands: Ford and Lincoln`
nn nn	34,154	1.13	`` The ‘‘ Gone with the Wind ’’ of Punk Rock Samplers ``

Table 5.9: Top 10 most frequent patterns, including absolute and relative frequency with an example sentence

3 million patterns as well as one example sentence with underlined entities. In the table we can see that only one pattern, namely *nsubj prep pobj*, which occurred 37,750 times, represents one of our hand-crafted patterns, namely the intransitive verb pattern. None of the other nine patterns contained a subject and an object dependency relation in the pattern. Many of the patterns were extracted from faulty sentences, such as `Joseph College in Hartford, Connecticut, in 1950`, where obviously the sentence was ripped apart during extraction.

The top 10 most frequent pattern also show that especially for data properties it is difficult to extract relevant sentences from the given corpus. In our approach we reduce properties with a date at the object position to the year in the date, which leads to correctly extracted sentences, but sometimes these sentences do not represent the relation. Consider for example the sentence `The name was published by George Vancouver in 1798` with its parse tree visualized in Figure 5.3 on the next page. We assume for this example that the property is *dbo:deathDate*, with the date *10 May 1798*, which we reduced to *1798*. While the sentence was therefore extracted correctly, the pattern between both terms, namely the year and the name of the person, do not represent a valid lexicalization of the relation. As we do not have a pattern for this case, this sentences is ignored.

While hand-crafted paths limit the number of sentences considered, this has an advantage over automatically extracting paths between two given entities: Concentrating on grammatically valid and hand-crafted patterns reduces the noise tremendously. We support the claim that considering all shortest paths between entities increases the noise with Table 5.10 on the facing page. In this table we show the top 20 shortest paths, none of them represent valid lexicalizations.

In Figure 5.4 on page 84, we show how the frequency of the unique patterns is distributed. The figure shows that from the 172,816 unique patterns, 103,640 occurred only once and are therefore probably noise.

In Figure 5.5 on page 85 we show how the frequency of the patterns is distributed in terms of

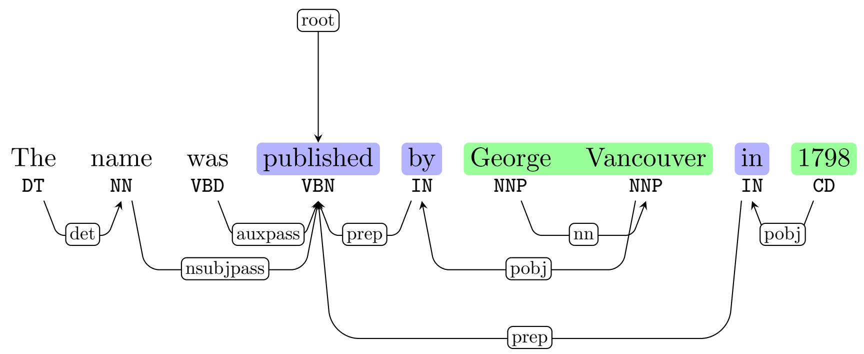


Figure 5.3: Dependency tree for the sentence *The name was published by George Vancouver in 1798.*

Lexicalizations	Frequency
`japan`	6,107
`canada`	6,071
`england`	5,866
`australia`	5,732
`pennsylvania`	5,431
`in a`	4,912
`2008`	4,883
`france`	4,423
`a in`	4,117
`india`	3,570
`london`	3,348
`scotland`	2,932
`2009`	2,752
`2010`	2,618
`york`	2,598
`in`	2,575
`spain`	2,502
`a in france`	2,485
`2012`	2,452
`and`	2,444

Table 5.10: Lexicalizations for the shortest paths including the frequency how often they occurred in the given corpus.

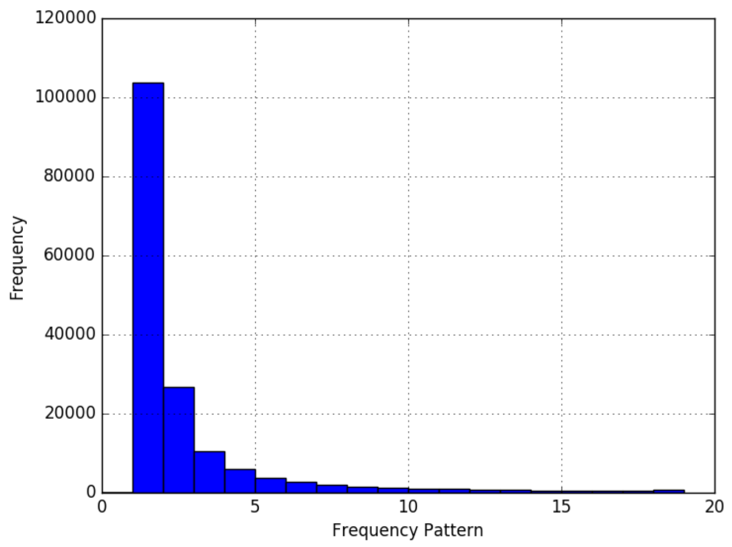


Figure 5.4: Power law of the frequency of the unique patterns, showing how many patterns occurred only once, how many only twice and so on.

their length. A pattern with length one is, for example, the pattern *nn*, while the pattern *pobj prep prep pobj* has the length four, and so on. Overall, 41 patterns had the length of one, for example representing a conjunction whereas 125,860 unique patterns had a length of 6 and more.

Let us now turn back to the question why the 21 terms mentioned in Table 5.8 were not extracted by M-ATOLL. The previous analysis justified that we can ignore all patterns with the length of one and all patterns that occurred only once. Additionally, we reduce the patterns to those which contain a subject and object dependency relation within the shortest path. This reduction leads to 562,471 shortest paths (21,557 unique paths). We present the most frequent paths in Table 5.11 on the next page.

After reviewing the results presented in Table 5.11 on the facing page, we decided to reduce these shortest paths furthermore to those paths which are also not longer than 5 elements and occur at least 1,000 times. This reduction reduced the number of shortest paths to 256,887, which are represented by 58 unique paths. For these paths we checked whether one of the 21 missing terms occurs or not. Overall we found 13 out of 21 terms in these paths. We show these 13 terms, including their frequency as well as the relevant path in Table 5.12 on page 86. For some terms we extracted several different paths, however we reduced this manually to the most likely ones.

10 of those paths are in fact variations of our intransitive verb pattern in M-ATOLL. We therefore adapted our intransitive pattern in the way that also an *nn* relation towards a subject or object can represent a given entity. When applying this new pattern on our corpus, however, we did not find the missing terms.

Actually, for 13 out of 21 terms we did find the terms as part of our lexicalization, but the lexicalization differ slightly from the one specified in the goldstandard as given in Table 5.8. A

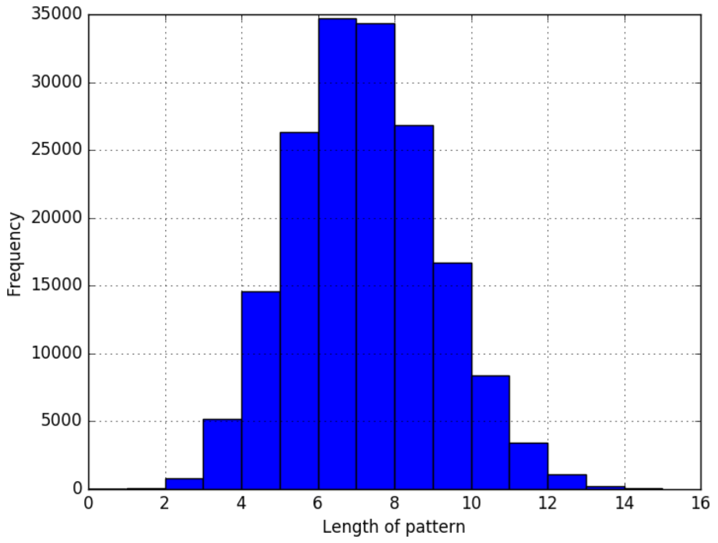


Figure 5.5: Power law of the lenght of the unique patterns, showing how many patterns have a length of one, how many have a length of two and so on.

Shortest Path	Frequency
nsubj prep pobj	37,750
nn nsubj prep pobj	24,741
nsubjpass prep pobj	17,457
nsubj prep pobj num	15,705
nsubj prep pobj prep pobj	13,674
nn nsubjpass prep pobj	12,661
nn nsubj prep pobj num	11,754
nsubj prep pobj nn	9,779
nsubjpass prep pobj num	9,512
nn nsubjpass prep pobj num	7,517

Table 5.11: Top 10 most frequent patterns after the first reduction.

Missing Term	Frequency	Shortest Path
`long`	225	nn nsubj prep pobj nn
`tall`	50	nn nsubj prep pobj nn
`format`	21	nn nsubj prep pobj nn
`wide`	15	nn nsubj prep pobj nn
`beam`	9	nn nsubj prep pobj
`weigh`	4	nn nsubj prep pobj nn
`fuse`	3	nn nsubj prep pobj nn
`revenue`	3	nn nsubj prep pobj
`weight`	2	nn nsubj prep pobj nn
`width`	1	nn nsubjpass prep pobj num
`runtime`	1	nn nsubjpass prep pobj nn
`length`	1	nn pobj prep nsubj
`density`	1	nn nsubj dobj prep pobj

Table 5.12: Terms not found by M-ATOLL including frequency and the relevant paths on a reduced corpus.

comparison of the results from M-ATOLL and the actual terms in the goldstandard are presented in Table 5.13 on the facing page.

Note that these results were already obtained by the original patterns.

The main reason for the difference is that we implemented M-ATOLL specifically to extract more of the context where the relation is embedded, e.g. *absolute magnitude of* and not only *magnitude*, because with this we can provide more accurate lexicalizations for a specific property. For example, consider the property *dbo:almaMater* for which M-ATOLL extracts the lexicalization `get degree from`, instead of `get from` without the grammatical object. However, this object `degree` is important for the overall meaning of the relation.

5.3 Comparison with other systems

In order to compare the results of M-ATOLL to other systems, we simulated systems, such as presented in the work by Mahendra et al. (2011) and Gerber and Ngomo (2011), by our baseline described in Section 4.6.2. Consider the Table 4.5 on page 68 from the previous chapter in which lexicalizations for the property *dbo:spouse* are presented. This table is reused in this section, extended by results from our baseline. The new table is now presented in Table 5.14 on the facing page. It shows that the baseline approach finds three lexicalizations, which are also found by M-ATOLL. Two of the three lexicalizations are also contained in the goldstandard. However, it is important to mention that the entries from the baseline did not contain any preposition. The top 10 most frequent lexicalizations for the property *dbo:spouse* from the baseline are presented in Table 5.15 on page 88.

Goldstandard	M-ATOLL
revenue	tax revenue from
length	full-length album by
density	have density in
wide	widen to
weigh	weight expert from
format	use format in
weight	weight expert from
abbreviate	abbreviated to
magnitude	absolute magnitude of
tall	tallest building in
fuse	refuse
reviewed	re-reviewed game in
long	longtime leader of

Table 5.13: Terms from the goldstandard with corresponding terms from M-ATOLL

Lexicalization	M-ATOLL	goldstandard	Baseline
wife of	+	+	$+^*$
husband of	+	+	$+^*$
to marry	+	+	-
spouse	+	+	-
married to	+	-	$+^*$
widow of	+	-	-
consort of	+	-	-
to meet	+	-	-
married person	+	-	-
better half	+	-	-
partner	+	-	-

Table 5.14: Comparison of the lexicalizations extracted for the property *dbo:spouse* from the different systems.

Lexicalization	Frequency
`wife`	488
`married`	279
`prince`	232
`was`	211
`daughter`	114
`is`	99
`husband`	97
`queen`	88
`duke`	60
`actress`	57

Table 5.15: Top 10 baseline results for the property *dbo:spouse*

To give more insights into the differences between the results of the baseline and those of M-ATOLL we present the results for further five properties: *dbo:birthDate* in Table 5.16 on the facing page, *dbo:elevation* in Table 5.17 on the next page, *dbo:crosses* in Table 5.18 on page 90, *dbo:locationCountry* in Table 5.19 on page 90 and *dbo:musicalArtist* in Table 5.20 on page 91.

For some properties, e.g. *dbo:elevation* the baseline approach clearly outperforms the results of M-ATOLL. However, this is due to the fact that for M-ATOLL it is currently difficult to match numerical values in a sentences, but might be revised by inserting a special preprocessing module, which can find and convert numerical values. For example, sometimes a height is given in DBpedia using the unit *foot*, but in the corresponding Wikipedia text the value is presented using the unit *km*, or vice versa.

With respect to other properties, e.g. *dbo:birthDate* or *dbo:musicalArtist*, M-ATOLL clearly outperforms the result of the baseline system.

Overall the presented tables show that M-ATOLL produces more lexical variants with less noise. Enriching the baseline with a machine learning based approach could minimize the noise within the baseline and put better verbalizations in a more prominent position.

5.4 Conclusion

In this chapter we presented three different analyses. In the first section we analyzed the impact of different filtering strategies on the quality of the resulting lexicon. The result showed that we could improve our results by choosing a different filtering strategy over the simple frequency-based filtering strategy. Especially for the machine-learning based filtering strategy we could improve the F-measure on the micro-based evaluation by 10% and on the macro-based evaluation by around 7%.

M-ATOLL	Frequency	Baseline	Frequency
born in	2042	born	87,206
born on	42	march	12,008
die in	36	january	11,857
bear	25	february	11,181
move in	10	may	11,048
born to	9	september	10,984
begin in	8	august	10,927
elected in	7	october	10,894
graduate in	7	april	10,863
return in	6	july	10,784

Table 5.16: Top 10 baseline and M-ATOLL results for the property *dbo:birthDate*

M-ATOLL	Frequency	Baseline	Frequency
change name in	2	is	411
have	2	elevation	261
born in	2	feet	150
described in	1	has	118
join organisation in	1	was	112
graduate in	1	route	66
die in	1	altitude	62
question use in	1	state	59
editor from	1	area	53
open concept in	1	covers	50

Table 5.17: Top 10 baseline and M-ATOLL results for the property *dbo:elevation*

M-ATOLL	Frequency	Baseline	Frequency
`bridge over`	3	`bridge`	47
`bridge across`	3	`is`	46
`cross`	2	`crosses`	17
`road bridge over`	2	`river`	17
`built across`	1	`road`	7
`short distance from`	1	`was`	7
`commence 1954 across`	1	`is bridge`	7
`span bridge over`	1	`is bridge crosses`	6
`cast-iron bridge over`	1	`spans`	5
`connection across`	1	`island`	4

Table 5.18: Top 10 baseline and M-ATOLL results for the property *dbo:crosses*

M-ATOLL	Frequency	Baseline	Frequency
`bank in`	9	`is`	1,223
`television station in`	7	`was`	555
`start telenovelum in`	6	`company`	321
`company in`	5	`based`	277
`trade union in`	5	`prefecture`	245
`established in`	4	`line`	160
`founded in`	4	`television`	149
`television network in`	4	`kanagawa`	142
`released in`	4	`released`	141
`return to`	3	`aichi`	135

Table 5.19: Top 10 baseline and M-ATOLL results for the property *dbo:locationCountry*

M-ATOLL	Frequency	Baseline	Frequency
`song by`	83	`is`	3,591
`single by`	72	`song`	1,731
`release`	46	`band`	798
`perform`	37	`was`	520
`single from`	23	`rock`	429
`write`	20	`recorded`	370
`studio album by`	14	`singer`	367
`record`	13	`album`	315
`album by`	8	`artist`	284
`cover`	7	`group`	282

Table 5.20: Top 10 baseline and M-ATOLL results for the property *dbo:musicalArtist*

In the second section we had a more detailed look at the given text corpus. We could show the superiority of M-ATOLL over approaches using only the shortest path between entities or only the flat string between two entities as a lexicalization. We could show that we did not find 21 terms which are found by the baseline, but also argue that we did not find them because of missing patterns, but because of different design choices between the lexica of M-ATOLL and the goldstandard.

In the last section we presented in more detail the differences in the results from the baseline and M-ATOLL based on six different properties.

Chapter 6

Extension: Adjective lexicalizations for restriction classes

After giving an example, where the current version of M-ATOLL fails, we introduce this challenge and provide a machine-learning based solution.

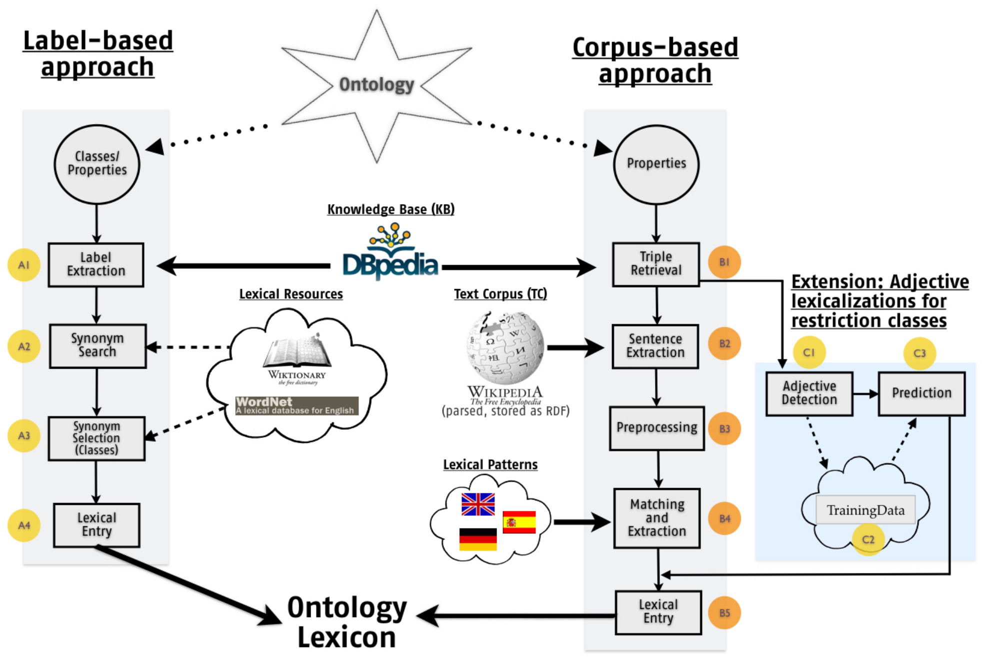


Figure 6.1: General Architecture of M-ATOLL, including the extension for adjective lexicalizations for restriction classes.

6.1 Introduction

The previous sections clearly showed that M-ATOLL is able to produce high quality lexical entries in order to solve the lexical gap for questions such as `Barack Obama married Michelle`. It does fail however on more complex adjective relations, as introduced in the first Section in Example 1.3 on page 3. However, those adjectives are an important piece of lexical knowledge.

For example, the 250 training and test questions of the QALD-4 Unger et al. (2014a) benchmark[1] for question answering over DBpedia contain 76 different adjectives. Most of these adjectives are gradable (e.g. high) or intersective (e.g. Australian). While the former cannot be directly represented in OWL (see McCrae et al. (2014)), the latter denote simple restriction classes involving nominals. For example, Danish denotes the class ∃ country.{Denmark}, female denotes the class ∃ gender.{Female}, and Catholic denotes the class ∃ religion.{Catholic_Church, Catholicism}.

Knowledge about such adjectives is crucial, for instance, when translating natural language questions such as 1 into SPARQL queries such as 2.

1. Which female Danish politicians are catholic?

2.
```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
SELECT DISTINCT ?x WHERE {
```

[1]http://www.sc.cit-ec.uni-bielefeld.de/qald/

Adjective	Corresponding part in SPARQL query
female	`?x dbo:gender res:Female .`
Danish	`?x dbo:country res:Denmark .`
catholic	`?x dbo:religion res:Catholic_Church .`
catholic	`?x dbo:religion res:Catholicism .`

Table 6.1: Mapping from adjectives in the question in 1 to the corresponding parts of the SPARQL query in 2.

```
    ?x rdf:type dbo:Politician .
    ?x dbo:country res:Denmark .
    ?x dbo:gender res:Female .
  { ?x dbo:religion res:Catholic_Church . }
  UNION
  { ?x dbo:religion res:Catholicism .} }
```

In this example, the required mappings of adjectives in the natural language question in 1 to the SPARQL query presented in 2 are as presented in Table 6.1. (Note that the semantics of the adjective catholic is represented by a union of two basic graph patterns.) These mappings are much harder to automatically find than, e.g., mapping the noun `politician` to the corresponding class *dbo:Politician*, in particular because the natural language string does not provide information on the required property.

Current approaches to learning lexicalizations, such our presented M-ATOLL, but also BOA by Gerber and Ngomo (2011) or WRPA by Vila et al. (2010), do not yet include methods for learning adjective lexicalizations. Therefore, the generated lexicons are necessarily incomplete and do not provide support when interpreting questions that include adjectives, such as the one above.

In this chapter, we propose a machine learning approach to the extraction of adjective lexicalizations from a given knowledge base, in our case DBpedia.

Our approach is presented in Figure 6.1, where it is marked with a **C**. It consists of the following steps:

C1 For a given property *p*, extract all RDF triples and keep those triples which contain at least one adjective in the object

C2 Training of a model (Random Forest) with the given training files.

C3 Using the model from the previous step, predict if a lexical entry for a given RDF triple should be serialized into a lexical entry.

Applying our approach to the whole of DBpedia 2014, we extract 15,380 adjective lexicalizations with an accuracy of 91.15 %. We provide the extracted lexicalizations in *lemon* format for free use:

https://github.com/ag-sc/matoll/public/june_2016/adjectives.ttl

The chapter is structured as follows. In Section 6.2 we introduce our machine learning approach to extracting adjective lexicalizations, describing in particular the training dataset and the features used. In Section 6.3 we outline our experiments and results, providing insights into the impact of each feature on the task. We then describe the resulting adjective lexicalization dataset in Section 6.5.

6.2 Method

The main intuition underlying our approach to learn adjective lexicalizations with respect to a knowledge base is that an inspection of the objects occurring with a particular property often suggests relevant lexicalization candidates. The DBpedia property *dbo:gender*, for example, occurs very often with resources *res:Male* and *res:Female* as objects. Thus, `male` and `female` are obvious lexicalization candidates for the restriction classes $\exists$ `gender.`{`Male`} and $\exists$ `gender.`{`Female`}, respectively. Similarly, the property *dbo:country* occurs with objects like *res:Denmark* and *res:Germany*, which have related adjective forms `Danish` and `German`, which are lexicalizations of the restriction classes $\exists$ `country.`{`Germany`} and $\exists$ `country.`{`Denmark`}, respectively.

In this section we explain in detail how our approach implements the extraction of adjectives from object occurrences and the decision which of the resulting candidates are valid lexicalizations.

6.2.1 Algorithm

Our approach comprises four steps, which are presented as pseudocode in Algorithm 5.

Algorithm 5 Approach to learning adjective lexicalizations.

```
1: for property $p$ do
2:    $objects(p) \leftarrow \{o \mid (\_, p, o) \in KB\}$
3:    for $o$ in $objects(p)$ do
4:       for terms $a$ contained in $label(o)$ do
5:          if $a$ is an adjective then
6:             $\vec{v} \leftarrow extractFeatures(a, p, o)$
7:             if $classifyAsLexicalization(\vec{v})$ then
8:                $createAdjectiveEntry(a, p, o)$
9:             end if
10:         end if
11:      end for
12:   end for
13: end for
```

First, for a given property p, all RDF triples $(\_, p, o)$ are retrieved from the knowledge base (see Algorithm 5 line 2). In particular, we are interested in the resource labels and literals appearing

```
1  :LE_<a>
2        a lemon:LexicalEntry;
3        lemon:canonicalForm :LE_<a>_canonicalForm ;
4        lemon:sense :LE_<a>_sense ;
5        lemon:synBehaviour :LE_<a>_synBehaviour_1 ;
6        lemon:synBehaviour :LE_<a>_synBehaviour_2 ;
7        lexinfo:partOSpeech lexinfo:adjective .
8
9  :LE_<a>_canonicalForm lemon:writtenRepresentation "<a>"@en ;
10
11 :LE_<a>_sense a lemon:Sense ;
12       lemon:reference LE_<a>_reference ;
13       lemon:isA _:b0 ;
14
15 :LE_<a>_reference a owl:Restriction ;
16       owl:onProperty <p> ;
17       owl:hasValue   <o> .
18
19 :LE_<a>_synBehaviour_1 a lexinfo:AdjectiveAttributiveFrame;
20       lexinfo:attributiveArg _:b0.
21
22 :LE_<a>_synBehaviour_2 a lexinfo:AdjectivePredicativeFrame;
23       lexinfo:copulativeSubject _:b0.
```

Figure 6.2: Template for adjective entry in *lemon* format, where `<a>` is a slot for the adjective form, `<p>` is a slot for the property, and `<o>` is a slot for the corresponding object.

at the object position of property p. From now on, for some object o, we refer to these labels and literals as *label(o)*.

Then we extract adjective forms from the object labels by checking for each term occurring in the label whether WordNet (Miller (1995)) and DBnary (Gilles (2015)) contain it as adjective, thereby building 3-tuples of the form (a, p, o), where a represents the adjective form found in the label of the object o, and p is the property with which o occurred (see Algorithm 5 lines 4 and 5). For example, from the object label Catholic Church the term *'catholic'* is extracted. These extracted 3-tuples represent the candidate lexicalizations.

Subsequently, we classify all 3-tuples (a, p, o) with respect to whether they constitute a valid adjective lexicalization or not. To this end, we train a classifier using the features described in Section 6.2.3 below. In case the 3-tuple is classified positively, we finally create a lexical entry in *lemon* format (see Algorithm 5 lines 7 and 8), using the template given in Figure 6.2.

In lines 1-7 of Figure 6.2, a resource of type `lemon:LexicalEntry` is created, where lines 3-7 specify the basic linguistic properties of this entry, such as its canonical form, its sense, its syntactic behavior and part of speech (`lexinfo:adjective` in our case). The canonical form indicates the written representation of the adjective (line 9), e.g. Catholic.

The property `lemon:sense` is used to express the meaning of the adjective with respect to the corresponding knowledge base by pointing to a restriction class (line 12), for example $\exists$ `religion.`{`Catholic_Church`}, which is defined in lines 15–17. In this case, `<p>` would be filled with the property *dbo:religion* and `<o>` would be filled with the object `Catholic_Church`.

Furthermore, each entry specifies two syntactic behaviors of the adjective:

- an `AdjectiveAttributiveFrame` (lines 19–20), capturing the attributive use of the adjective, as in *'the catholic politician'*.
- an `AdjectivePredicativeFrame` (lines 22–23), capturing the predicative use of the adjective, e.g. in a copula construction such as *'This politician is catholic'*.

The full entry for the adjective Catholic is shown in Figure 6.3.

We applied our approach to DBpedia 2014, processing every property and considering all objects of these properties. In the following sections we describe how the training data for the classifier has been created, and define the features used.

6.2.2 Training Dataset

Our training dataset consists of the five properties used by Walter et al. (2014b), i.e. *dbo:colour*, *dbo:architecturalStyle*, *dbo:geologicPeriod*, *dbo:militaryBranch* and *dbo:party*, together with eight additional properties that were randomly chosen to reduce the bias in our evaluation towards these manually selected properties. The resulting 12 DBpedia properties are shown in Table 6.2.

For each of these properties, we extracted all objects from DBpedia 2014 by means of the following SPARQL query:

```
SELECT ?o WHERE { _ <p> ?o . }
```

Given the resulting set $objects(p)$, we compute the frequency $freq(p, o)$ for each $o \in objects(p)$. For instance, for the property *dbo:religion*, there are 28,928 different objects; the top 10 most frequent ones are given in Table 6.3 together with the number of their occurrences. For the given properties, all unique 3-tuples (a, p, o) were extracted. The number of 3-tuples extracted for each of the considered properties is shown in Table 6.2; it represents the amount of candidate adjective entries for that property.

The authors of this paper then annotated all 3-tuples with respect to whether they represent a valid adjective lexicalization or not, resulting in 1,074 valid lexicalizations and 1,051 non-valid ones.

In the next section we describe the features used to describe each 3-tuple.

6.2.3 Features

All features are defined over 3-tuples (a, p, o), where a is an adjective that occurs in the label of an object o, and p is the property with which o occurs. To formally define the features, we introduce

```
:LE_Catholic
      a lemon:LexicalEntry;
      lemon:canonicalForm :LE_Catholic_canonicalForm;
      lemon:sense :LE_Catholic_sense;
      lemon:synBehaviour :LE_Catholic_synBehaviour_1;
      lemon:synBehaviour :LE_Catholic_synBehaviour_2;
      lexinfo:partOSpeech lexinfo:adjective.

:LE_Catholic_canonicalForm lemon:writtenRepresentation "Catholic"@en ;

:LE_Catholic_sense a lemon:Sense;
      lemon:reference LE_Catholic_reference;
      lemon:isA _:b0.

:LE_Catholic_reference a owl:Restriction;
      owl:onProperty <http://dbpedia.org/ontology/religion>;
      owl:hasValue   <http://dbpedia.org/resource/Catholic_Church>.

:LE_Catholic_synBehaviour_1 a lexinfo:AdjectiveAttributiveFrame;
      lexinfo:attributiveArg _:b0.

:LE_Catholic_synBehaviour_2 a lexinfo:AdjectivePredicativeFrame;
      lexinfo:copulativeSubject _:b0.
```

Figure 6.3: Lexical entry for the property *dbo:religion* with the object `Catholic_Church`.

Property (p)	Number of 3-tuples (a, p, o)
dbo:religion	427
dbo:birthPlace	402
dbo:ethnicity	326
dbo:operatedBy	322
dbo:architecturalStyle	257
dbo:gameEngine	132
dbo:campusType	117
dbo:colour	83
dbo:geologicPeriod	35
dbo:gender	15
dbo:associatedRocket	5
dbo:foundedBy	3
dbo:elevation	2

Table 6.2: Number of 3-tuples (a, p, o) for the annotated training properties.

Object	Frequency
`Catholic_Church`	6,422
`Islam`	1,959
`Christian`	1,042
`Baptists`	879
`Presbyterianism`	804
`Sunni_Islam`	781
`Hindu`	779
`Methodism`	772
`Episcopal_Church_(United_States)`	766

Table 6.3: Top 10 most frequent objects for the property *dbo:religion*.

the following functions, where *KB* denotes the underlying knowledge base (in our case DBpedia).

- The function *label*(o) returns the label of a given object o, e.g. for the resource `Catholic_Church` the label Catholic Church is returned. For literals, *label* simply returns the literal value unchanged.
- The function $\mathit{adjfreq}_{KB}(p)$ computes the number of objects o for a given property p that *'contain'* an adjective, i.e. for which an adjective is an element in the tokenized sequence of *label*(o):

$$\mathit{adjfreq}_{KB}(p) = \sum_{(s,p,o)\in KB} \begin{cases} 1, & \text{if } \mathit{label}(o) \text{ contains an adjective} \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

- The function $\mathit{adjfreq}_{KB}(p, a)$ returns the frequency of the particular adjective a occurring in objects of property p:

$$\mathit{adjfreq}_{KB}(p, a) = \sum_{(s,p,o)\in KB} \begin{cases} 1, & \text{if } \mathit{label}(o) \text{ contains } a \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

  For example, $\mathit{adjfreq}_{DBpedia}(\texttt{religion})$ is 12,720, i.e. 12,720 objects that occur as object of `religion` contain an adjective, and $\mathit{adjfreq}_{DBpedia}(\texttt{religion}, \text{Hindu})$ is 779, i.e. the adjective Hindu occurs in 779 of those objects.

- Another function we will use is $\mathit{freq}_{KB}(p, o)$, which returns the overall frequency of a given object o in all RDF triples with a given property p:

$$freq_{KB}(p, o) = \sum_{(s,p,o')\in KB} \begin{cases} 1, & \text{if } o' = o \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

  As an example consider the object `Greek_Orthodox_Church`, which occurs 174 times in the overall list of objects for the property `religion`, i.e. $\mathit{freq}_{DBpedia}(\texttt{religion}, \texttt{Greek\_Orthodox\_Church}) = 174$.

  Analogously, we will use a function $\mathit{pfreq}_{KB}(p, p')$ that captures the frequency of a pattern p' in an object of all RDF triples with a given property p. (see 6.2.3 and 6.2.3).

In the following, we describe the features we used.

Normalized Object Frequency (NOF)

The *Normalized Object Frequency* counts the number of times that an object o occurs in objects of the property p, divided by the number of objects that actually contain an adjective:

$$NOF_{KB}(p, o) = \frac{freq_{KB}(p, o)}{adjfreq_{KB}(p)} \tag{6.4}$$

To illustrate this, consider the property *dbo:religion* with the object Greek Orthodox Church, which occurs 174 times in *dbo:religion.* The total number of objects containing an adjective is 12,720, therefore $NOF_{DBpedia}(\texttt{dbo:religion}, \texttt{Greek Orthodox Church}) = \frac{174}{12{,}720} = 0.013$.

Normalized Adjective Frequency (NAF)

The *Normalized Adjective Frequency* counts the number of times a particular adjective occurs in objects of a given property p, normalized by the total number of adjectives:

$$NAF_{KB}(a,p) = \frac{adjfreq_{KB}(p,a)}{adjfreq_{KB}(p)} \tag{6.5}$$

As an example, consider the adjective Catholic. It appears 5,545 times in objects of the property *dbo:religion*, and the objects containing an adjective is 25,587 adjectives in total. Therefore, $NAF_{DBpedia}(\text{Catholic}, \texttt{dbo:religion}) = \frac{5{,}545}{25{,}587} = 0.21$. This means that out of all adjectives occurring in objects of the property *dbo:religion*, Catholic occurs in 21% of the cases.

Adjective Ratio (AR)

The *Adjective Ratio* computes the number of objects of a property p that contain an adjective, normalized with respect to the total number of objects:

$$AR_{KB}(p) = \frac{adjfreq_{KB}(p)}{|objects_{KB}(p)|} \tag{6.6}$$

For the property `religion`, for example, 12,720 adjectives are found in 25,587 objects. Therefore, $AR_{DBpedia}(\texttt{dbo:religion}) = \frac{12{,}720}{25{,}587} = 0.49$, which means that only roughly half of its objects contain an adjective.

Pattern Ratio (PR)

We transform the label of each object $o \in$ *objects*(p) into a pattern by replacing the corresponding adjective a by the string *ADJ*. For the property *dbo:religion*, for example, this yields the patterns shown in Table 6.4. Therefore, the function *pattern*(o), which is used below, returns for a given object o the corresponding pattern, e.g. for the object Greek Orthodox Church the pattern Greek *ADJ* Church is returned.

For each triple (a,p,o), we record the frequency of the pattern extracted from o normalized by the sum of the frequencies of all patterns appearing in objects of that property:

$$PR_{KB}(p,o) = \frac{pfreq_{KB}}{(p, pattern(o))} \sum_{o' \in objects(p)} pfreq_{KB}(p, pattern(o')) \tag{6.7}$$

The *Pattern Ratio* thus describes the number of occurrences of the pattern of a particular object compared to the sum of the occurrences of the patterns of all objects.

Part-of-Speech Pattern Ratio (POSPR)

Next we define an extension of the pattern ratio above, replacing all terms in an object label by their part-of-speech tag. The label Catholic Church, for example, would resolve into the pattern *JJ NN*. We refer to this pattern of an object o as *POSpattern(o)* and define the *Part-of-Speech Pattern Ratio* as follows:

Pattern	Frequency
ADJ Church	5,721
ADJ	2,570
ADJ Islam	772
Eastern *ADJ* Church	619
The Church of Jesus Christ of *ADJ* Saints	300
United *ADJ* Church	231
Greek *ADJ* Church	137
Serbian *ADJ* Church	129
ADJ Church of Canada	120
Southern *ADJ* Convention	106
Armenian *ADJ* Church	76
Dutch *ADJ* Church	72
ADJ Buddhism	70
Roman *ADJ* Kirk	64
ADJ Christianity	59
ADJ Religion	58
ADJ Judaism	57
ADJ Universalism	53
Seventh-day *ADJ* Church	51
Russian *ADJ* Church	42

Table 6.4: Top 20 most frequent patterns for the property *dbo:religion* together with their frequency.

$$POSPR_{KB}(p,o) = \frac{pfreq_{KB}(p, POSpattern(o))}{\sum_{o' \in objects(p)} pfreq_{KB}(p, POSpattern(o'))} \tag{6.8}$$

It thus describes the number of occurrences of the part-of-speech pattern of some object compared to the sum of occurrences of the part-of-speech patterns of all objects.

Position Features (AP, AFP, ALP)

This group of features encodes information about the position of the adjective a in the label of an object o.

For each 3-tuple (a, p, o), we encode the position at which a occurs in the label of o as a feature AP. For example, the adjective Catholic occurs in the label Catholic Church at position 1, while the adjective Orthodox occurs in Romanian Orthodox Church at position 2.

As specific cases, the Boolean feature *Adjective First Position (AFP)* indicates whether the adjective occurs as the first word in the label of the object, and the Boolean feature *Adjective Last Position (ALP)* indicates whether the adjective occurs as the last word in the label of the object.

Normalized Levenshtein Distance (NLD)

This feature computes the Normalized Levenshtein (see Levenshtein (1966)) distance between an adjective a and the label of an object o:

$$NLD(a,o) = \frac{LevenshteinDistance(a, label(o))}{max(length(a), length(label(o)))} \tag{6.9}$$

Character n-grams

For each adjective a in (a, p, o), we extract its character trigram and bigram suffixes. For example, for the adjective Hindu we extract the character trigram -ndu and the character bigram -du. For each such character n-gram, we construct one Boolean feature recording whether the n-gram occurs in the particular adjective a or not, yielding a total of 368 trigram and 160 bigram features for each adjective in our dataset.

Part-of-Speech Pattern (PP)

For each adjective a in (a, p, o), we extract the part-of-speech pattern $POSpattern(o)$. Overall, the data contains 428 different patterns. We include a feature for each of these 398 patterns indicating if the object o matches the given pattern.

Part-of-Speech Adjective Pattern (PAP)

For each adjective a in (a, p, o) we replace a in the label of o by the string *ADJ* and replace all other terms by their part-of-speech tag. For the adjective Catholic, for example, the object label Catholic Church is transformed into the pattern *ADJ* NN. We consider 398 patterns and as for the PP feature we include a feature for each of these 428 patterns indicating if the object o matches the given pattern.

6.3 Evaluation

In our experiments we test eight different classifiers: Decision Trees, Random Trees, Random Forests, Support Vector Machines with Sequential Minimal Optimization, Logistic Regression, Bayes Nets, Voted Perceptrons and Decision Tables, as implemented in WEKA 3.8 (Hall et al. (2009)). For each we perform an exhaustive search in the space of all feature groups to determine the best set of features for the task. As Random Forests achieved the best results, we report the results of the best feature set for this classifier. Then we compare the results for this feature set for all classifiers, see Section 6.3.2.

We use a ten-fold cross validation setting to perform the feature selection. We do not evaluate the selected features on additional, unseen data, but we regard our results as clear indicators for the features that work well on the task, given that the results are similar across classifiers, with 91.15% the top accuracy (Random Forest) and 82.68% (Decision Table) the lowest.

In addition, we perform a ten-fold cross validation to compute the accuracy of each feature in isolation, see Section 6.3.1.

The training data is as described in Section 6.2.2 and consists of the list of annotated 3-tuples (a, p, o). With 1,074 positive entries and 1,051 negative entries, the dataset is balanced. We use the cross validation strategy built in into WEKA, thus each triple (a, p, o) is contained in exactly one fold and triples sharing one element are not necessarily in the same fold.

6.3.1 Accuracy of Features in Isolation

We evaluated all 13 feature groups individually in a ten-fold cross validation setting. The results are presented in Table 6.5, starting with the feature group with the highest accuracy. Additionally, for each feature, the variance and standard deviation is given.

The feature with the highest impact is the *Normalized Adjective Frequency (NAF)*, reaching an accuracy of 83.48% with a standard deviation of 2.42, followed by the group of trigram features encoding the last three characters of an adjective, reaching an accuracy of 80.37% with a standard deviation of 2.49. Bigrams lead to an accuracy of 76.14% with a standard deviation of 2.91. The features with the lowest impact are the features which encode the position of the adjective, namely *AP*, *AFP* and *ALP*, with an accuracy of 59.52%, 57.93% and 54.21%, respectively. From these three, the *AP* feature is the best, as it encodes the direct position of the adjective and therefore indirectly subsumes the other two position features.

The adjective ratio *AR* was the main feature in our previous work (Walter et al. (2014b)), representing the *adjectivehood* of the objects encountered for a property. It reaches an accuracy of 75.67% and ranks highest when using a Decision Tree classifier.

6.3.2 Feature Analysis

In order to analyze the impact of the features also in combination with other features, we perform an exhaustive search over the possible feature combinations. This amounts to an evaluation of

Feature	Accuracy	Variance	Standard Deviation
NAF	83.48	5.85	2.42
Trigrams	80.37	6.2	2.49
Bigrams	76.14	8.46	2.91
PR	75.72	9.24	3.04
POSPR	75.72	9.24	3.04
AR	75.67	9.24	3.04
PAP	73.08	10.11	3.18
NOF	71.30	9.79	3.13
PP	71.29	8.52	2.92
NLD	70,73	11.35	3.37
AP	59.52	7.34	2.71
AFP	57.93	10.89	3.30
ALP	54.21	9.98	3.16

Table 6.5: Accuracy (in %) per feature.

8,192 feature combinations.[2] Each combination was separately evaluated using our ten-fold cross validation task, in order to determine the most effective feature combination. Table 6.6 reports the ten best combinations, all of which outperform each feature in isolation. The highest accuracy is 91.15%, achieved by a Random Forest classifier and using only 7 out of the 13 feature groups; the lowest accuracy is 54%.

In Table 6.7 we report the results achieved by all eight classifiers, once with the best feature combination (PR, ALP, Trigrams, AFP, AP, AR, see Table 6.6) and once using all features, showing that the Random Forest classifier achieves the best results in both cases.

It is interesting to highlight that the optimal feature combination for the Random Forest also represents the optimal feature combination for four out of the other seven classifiers (Decision Tree, Random Tree, Logistic Regression, Decision Table). The Bayes Net classifier showed no difference in accuracy, while the SVM with SMO and the Voted Perceptron yield a small increase of accuracy when using all features.

After evaluating all features (single and in combination) on a ten-fold cross validation task we decided to evaluate our approach on a strict train/test split setting. We randomly split our dataset in a balanced and equally sized training and test dataset. We used the Random Forest classifier with the best feature combination in order to train our model. Applying this model to the unseen test dataset lead to an accuracy 73.52%. This value is (as expected) lower as in the ten-fold cross validation task, but still shows the potential of our approach in combination with our features.

[2]Which can be downloaded at `http://sebastianwalter.org/dblexipedia/jods_arff_files.tar.gz` as `.arff` files.

Feature Combinations	Acc.
PR, ALP, Trigrams, AFP, AP, AR, NAF	91.15
PP, NOF, ALP, Trigrams, AR, NAF, POSPR, Bigrams	90.87
PP, PR, ALP, Trigrams, AP, AR, NAF, Bigrams	90.82
PP, PR, Trigrams, NAF, POSPR, Bigrams	90.78
PP, PR, ALP, Trigrams, AP, AR, NAF, POSPR	90.73
PP, NOF, ALP, Trigrams, AFP, AR, NAF, POSPR, Bigrams	90.72
PP, PR, Trigrams, AR, NAF, POSPR, Bigrams	90.68
PP, PR, NOF, ALP, Trigrams, AR, NAF, Bigrams	90.63
PP, Trigrams, AP, AR, NAF, POSPR, Bigrams	90.63
PP, PR, ALP, Trigrams, NAF	90.59

Table 6.6: Accuracy (in %) for the top 10 feature combinations.

Classifier	Accuracy (best)	Accuracy (all)
Random Forest	91.15	89.83
Decision Tree (J48)	87.29	86.63
Random Tree	86.68	81.97
SVM with SMO	85.64	86.50
Logistic Regression	85.27	83.95
Bayes Net	84.80	84.80
Voted Perceptron	83.62	84.09
Decision Table	82.68	81.60

Table 6.7: Accuracy (in %) for all classifiers, using the best feature combination (PR, ALP, Trigrams, AFP, AP, AR) as well as all features.

Property	Accuracy	Created entries
dbo:colour	84	71
dbo:gender	75	12
dbo:religion	80	271
dbo:governmentType	67	88
dbo:nationality	88	361
dbo:originalLanguage	100	15

Table 6.8: Manually evaluated accuracy (in %) on six example properties.

6.4 Discussion

In this section we discuss the results for six properties in more detail – three properties from the training set (*dbo:colour*, *dbo:gender*, *dbo:religion*) and three other properties (*dbo:governmentType*, *dbo:nationality*, *dbo:originalLanguage*) in order to verify the generalization of our approach to properties not seen during training.

Table 6.8 shows the accuracy and the number of created lexical entries for the above mentioned six properties. For each property, all lexical entries generated by our approach were manually evaluated, deciding whether they represent valid lexicalizations. The results show that our approach extracts reasonable lexicalizations, with an accuracy ranging from 67 % for `governmentType` to 100 % for `originalLanguage`.

In Table 6.9 we show examples of the extracted adjectives for each property from Table 6.8, together with the object from which the adjective was extracted.

An adjective that is frequently proposed by our method is the adjective united, occurring in object labels such as United States (with the property *dbo:governmentType*) or United Church of Christ (with the property *dbo:religion*). Such cases are difficult to automatically distinguish from cases where the lexicalizations are correct. In some cases it is, in fact, even hard to decide as a human whether the adjective is indeed a suitable lexicalization. For example, the property *dbo:nationality* often occurs with objects describing languages, such as French language and Kurdish languages.

We assume that we could improve the results by taking into account the range of a given property, thus learning that particular adjective lexicalizations make sense only for particular types of objects.

Another example that demonstrates the limitations of our approach is the property *dbo:birthPlace*. For this property a total of 2,519 lexical entries were extracted; some example lexicalizations together with the corresponding object label are presented in Table 6.10. The problem is that there is a high number of adjectives in the object labels but most of them should be filtered out – a case that the classifiers did not encounter clearly enough during training.

Due to the nature of our approach, no adjective lexicalizations are extracted for datatype

Property	Adjective	Object
dbo:colour	red	`Red`
dbo:colour	purple	`Purple`
dbo:colour	cyan	`Cyan`
dbo:colour	ivory	`Ivory_(color)`
dbo:gender	female	`Female`
dbo:gender	male	`Male`
dbo:gender	unisex	`Unisex`
dbo:gender	mixed-sex	`Mixed-sex_education`
dbo:religion	christian	`Saint_Thomas_Christian_churches`
dbo:religion	catholic	`Roman_Catholic_Diocese_of_Saskatoon`
dbo:religion	apostolic	`New_Apostolic_Church`
dbo:religion	orthodox	`Georgian_Orthodox_Church`
dbo:governmentType	constitutional	`Constitutional_monarchy`
dbo:governmentType	multi-party	`Multi-party_system`
dbo:governmentType	federation	`Federation`
dbo:governmentType	district	`District_municipality`
dbo:nationality	scottish	`Scottish_people`
dbo:nationality	portuguese	`Portuguese_people`
dbo:nationality	peruvian	`Peruvian_people`
dbo:nationality	sikh	`Sikh`
dbo:originalLanguage	latin	`Latin`
dbo:originalLanguage	hindi	`Hindi`
dbo:originalLanguage	greek	`Ancient_Greek`
dbo:originalLanguage	english	`English_language`

Table 6.9: Example adjectives extracted for the six example properties.

Lexicalization	Object Label
spanish	Spanish Empire
french	French Chad
worthy	Martyr Worthy
national	National City, California
little	Little Plumstead
green	Birches Green

Table 6.10: Example lexicalizations for the property *dbo:birthPlace* together with the object label from which the adjective was extracted.

properties with literals other than strings. For properties such as *dbo:birthDate*, for example, adjective lexicalizations such as young or old would be relevant but fall outside the scope of our approach.

6.5 Resulting Dataset

We applied the trained Random Forest model to all 2,796 properties in DBpedia 2014. This resulted in a *lemon* lexicon with 15,380 adjective entries, with an average of 14.2 entries per property.

We make three datasets available: the file which was used to train the Random Forest model (`adjective.arff`), the model itself (`adjective.model`), and the resulting *lemon* lexicon in Turtle syntax (`adjectivelexicon.ttl`). All three files can be found at the following location:

https://github.com/ag-sc/matoll/tree/master/public/june_2016/

6.6 Conclusion

In this chapter we presented an approach for extracting adjective lexicalizations with respect to an ontology by analyzing the labels of objects occurring in a given dataset. We showed that the features we selected for this task achieve a reasonable accuracy. We thus introduced a relatively low-cost and accurate method for inducing an adjective lexicon, which can be straightforwardly applied to other ontologies and languages.

Chapter 7

DBlexipedia

In this chapter we give an overview of *DBlexipedia*, a multilingual ontology lexicon for DBpedia created by M-ATOLL. We briefly show how it can serve as a nucleus for the multilingual lexical Semantic Web and explain where all necessary resources can be found.

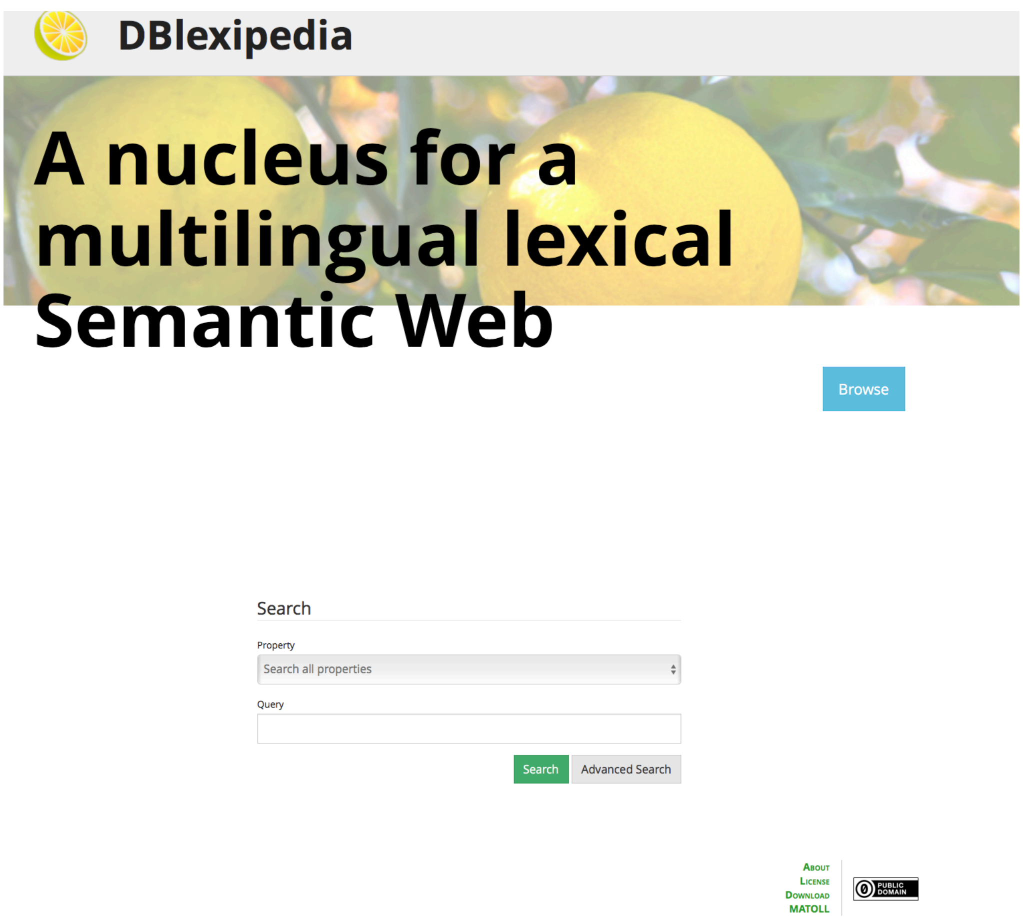


Figure 7.1: DBlexipedia: Landing page

7.1 Introduction

In this chapter we present *DBlexipedia*, a multilingual lexicon for DBpedia, covering English, german and Spanish lexicalizations created by M-ATOLL

It is available at:

http://dblexipedia.org/

Just like DBpedia provides a hub for Semantic Web datasets, this lexicon can provide a hub for the lexical Semantic Web, an ecosystem part of the linguistic linked data cloud in which ontology lexica are published, linked, and re-used across applications.

DBlexipedia is based on *YUZU*[1] by John P. McCrae, which is a micro-framework for publishing linked data. The website enables a user to browse through the lexical entries and supports search over them. The landing page of DBlexipedia is shown in Figure 7.1.

Here it is possible to search for certain values, for example for lexical entries for a particular property *dbo:spouse*[2] (see Figure 7.2), or to browse through the different entries. The browsing

[1]https://github.com/jmccrae/yuzu

[2]The results for this search can be found under: http://dblexipedia.org/search/?property=http%3A%2F%2Flemon-model.net%2Flemon%23reference&query=http%3A%2F%2Fdbpedia.org%2Fontology%2Fspouse

page is visualized in Figure 7.3.

In this chapter we present the numbers as presented in Walter et al. (2015), however, the next step is to update DBlexipedia to lexicalizations for the newest DBpedia version[3].

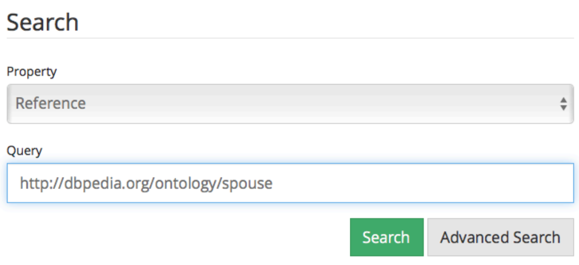
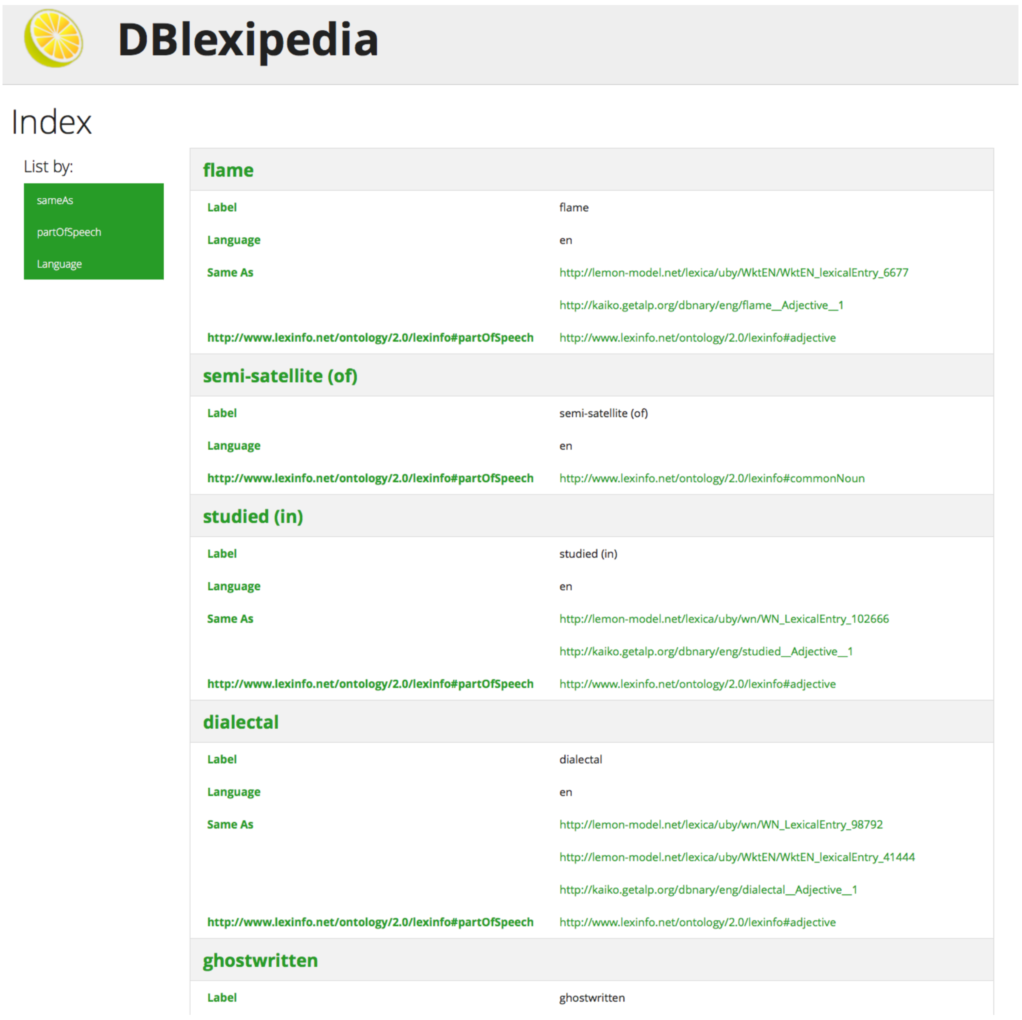


Figure 7.2: DBlexipedia: search for lexical entries with the property *dbo:spouse* as reference

[3]Seehttp://wiki.dbpedia.org/blog/yeah-we-did-it-again--new-2016-04-dbpedia-release



DBlexipedia

Index

List by:

sameAs

partOfSpeech

Language

flame

Label	flame
Language	en
Same As	http://lemon-model.net/lexica/uby/WktEN/WktEN_lexicalEntry_6677
	http://kaiko.getalp.org/dbnary/eng/flame__Adjective__1
http://www.lexinfo.net/ontology/2.0/lexinfo#partOfSpeech	http://www.lexinfo.net/ontology/2.0/lexinfo#adjective

semi-satellite (of)

Label	semi-satellite (of)
Language	en
http://www.lexinfo.net/ontology/2.0/lexinfo#partOfSpeech	http://www.lexinfo.net/ontology/2.0/lexinfo#commonNoun

studied (in)

Label	studied (in)
Language	en
Same As	http://lemon-model.net/lexica/uby/wn/WN_LexicalEntry_102666
	http://kaiko.getalp.org/dbnary/eng/studied__Adjective__1
http://www.lexinfo.net/ontology/2.0/lexinfo#partOfSpeech	http://www.lexinfo.net/ontology/2.0/lexinfo#adjective

dialectal

Label	dialectal
Language	en
Same As	http://lemon-model.net/lexica/uby/wn/WN_LexicalEntry_98792
	http://lemon-model.net/lexica/uby/WktEN/WktEN_lexicalEntry_41444
	http://kaiko.getalp.org/dbnary/eng/dialectal__Adjective__1
http://www.lexinfo.net/ontology/2.0/lexinfo#partOfSpeech	http://www.lexinfo.net/ontology/2.0/lexinfo#adjective

ghostwritten

Label	ghostwritten

Figure 7.3: DBlexipedia: Browsing the content

7.2 Dataset

The dataset we present in this thesis is the result of the above approaches and contains the entries in three languages: English, German, and Spanish. It covers lexicalizations for classes and properties for the DBpedia ontology namespace. Table 7.1 shows how many properties were lexicalized for each language. Overall, 574 different properties were lexicalized, but note that not every property is covered for each language. This is due to the fact that for some properties no sentences are found in the text corpus that match the predefined lexicalization patterns. For English, 567 properties were lexicalized, 224 by the corpus-based approach and 445 by the label-based approach. For German and Spanish, 145 and 50 properties, respectively, were lexicalized using the corpus-based approach (the label-based approach is currently not implemented for these languages).

Table 7.2 shows the number of entries generated by the different approaches for the different languages. Overall the published dataset contains 11,998 entries, but will be updated frequently.

It is important to mention that one lexical entry can lexicalize multiple properties. For example the lexicalization `wife` can describe the property *dbo:spouse* as well as the property *dbo:family*. Also one property can have multiple lexicalizations, for example the lexicalizations `married to` and `husband of` describe both the property *dbo:spouse*.

	Corpus-based	Label-based	Total
English	224	445	567
German	145	n.a.	145
Spanish	50	n.a.	50

Table 7.1: Number of lexicalized properties per approach and language.

	Corpus-based	Label-based	Total
English	4,456	4,092	8,548
German	3,320	n.a.	3,320
Spanish	130	n.a.	130

Table 7.2: Number of entries generated for each language for each approach.

Attached to the entries we also publish meta-data, in particular about provenance, specifying from which pattern an entry was created (for the corpus-based approach), with which frequency, and with which confidence (for the label-based approach). In the future we also intend to include example sentences for each entry (for the corpus-based approach) and the set of corresponding features which led to the entry (for the label-based approach).

Moreover, the lexical entries are linked to *dbnary*[4] and *lemon UBY*[5], based on their canonical form and the part of speech.

The whole dataset can also be downloaded at http://dblexipedia.org/public/all.nt.gz (as N-Triples). In addition to the dataset, the most current version of M-ATOLL is available at http://dblexipedia.org/public/MATOLL.jar.

7.3 Resources

With this thesis we publish also a series of resources which are necessary to run M-ATOLL and reproduce the published results. Currently, for server reasons, these resources are not published on http://dblexipedia.org, but will be moved there as soon as possible.

As dependency parsing of the whole Wikipedia corpus is very time consuming, we publish all three dependency-parsed text corpora. We used the Wikipedia dumps from April 2014 for the languages English, Spanish and German. Each line in the text files represent one dependency-parsed sentence in CoNLL format.

[4]See http://kaiko.getalp.org/about-dbnary/development/
[5]See http://www.lemon-model.net/lexica/uby/

- English:
  http://sebastianwalter.org/dblexipedia/raw/english_sentences.txt.tar.gz
- German:
  http://sebastianwalter.org/dblexipedia/raw/german_sentences.txt.tar.gz
- Spanish:
  http://sebastianwalter.org/dblexipedia/raw/spanish_sentences.txt.tar.gz

In addition, we make an already indexed version using Apache Lucene[6] available; the corresponding files are available here:

- English:
  http://sebastianwalter.org/dblexipedia/index/EnglishIndexReduced.tar.gz
- German:
  http://sebastianwalter.org/dblexipedia/index/GermanIndexReduced.tar.gz
- Spanish:
  http://sebastianwalter.org/dblexipedia/index/SpanishIndexReduced.tar.gz

Note, that for performance reasons we publish the reduced index files, containing only sentences which have 25 or less words in it. The raw files above contain all sentences.

We also publish the necessary data from the knowledge base, namely the entities which are connected by a property. For the property *dbo:spouse* an example data point looks as follows:

```
res:Abraham_Lincoln    Abraham Lincoln      Mary Todd Lincoln     res:Mary_Todd_Lincoln
```

containing the URI as well as the label of a resource pair. For all properties from the DBpedia 2014 ontology, for which data exists in the knowlegde base, the corresponding files can be downloaded at: http://sebastianwalter.org/dblexipedia/entities/ontology.tar.gz

We also publish indexed files (again as Lucene index), containing only those sentences that were used by M-ATOLL

- English:
  http://sebastianwalter.org/dblexipedia/input/input_EN.tar.gz
- German:
  http://sebastianwalter.org/dblexipedia/input/input_DE.tar.gz
- Spanish:
  http://sebastianwalter.org/dblexipedia/input/input_ES.tar.gz

Note that these files can be constructed by means of the previously mentioned resources. In the Appendix of this thesis (see Section A.4) we give a detailed description of the implementation of M-ATOLL and how to run it.

[6]Seehttp://lucene.apache.org/core/

7.4 Webservice

Additional to the DBlexipedia webpage, we publish the results of M-ATOLL as a REST-full web service. To explain the usage of this service consider the following examples with the corresponding service calls. The service takes as input a JSON file with arguments that specify attributes of the entries to be returned and returns the results also in JSON format. The possible arguments are as follows:

- **freq**
  Expects a integer value and checks if the frequency of an entry is equal or greater than this value.
- **lang**
  The default language is English (EN). The following languages are currently supported:
  - English: **EN**
  - German: **DE**
  - Spanish: **ES**
- **uri**
  Expects a reference such as `http://dbpedia.org/ontology/spouse` or `http://dbpedia.org/ontology/elevation`
- **pos**
  The following part-of-speech tags are supported:
  - Verb: **verb**
  - Noun: **noun**
  - Adjective: **adjective**
- **term**
  Expects a canonical form such as `wife` or `husband`.

1. - Give me all lexical entries with a frequency of 100 or more

   -
     ```
     curl -H "Content-type: application/json" \-X GET
     http://matoll.sebastianwalter.org/matoll -d
     '{"freq":"100"}'
     ```
2. - Give me all entries with the reference `http://dbpedia.org/ontology/spouse`

   -
     ```
     curl -H "Content-type: application/json" \-X GET
     http://matoll.sebastianwalter.org/matoll -d
     '{"uri":"http://dbpedia.org/ontology/spouse"}'
     ```
3. - Give me all verb entries

- `curl -H "Content-type: application/json" \-X GET http://matoll.sebastianwalter.org/matoll -d '{"pos":"verb"}'`

4. - Give me all lexical entries with the canonical form *village*
   - `curl -H "Content-type: application/json" \-X GET http://matoll.sebastianwalter.org/matoll -d '{"term":"village"}'`

5. - Give me all lexical entries with the canonical form *wife*, which occurred at least 5 times
   - `curl -H "Content-type: application/json" \-X GET http://matoll.sebastianwalter.org/matoll -d '{"freq":"5", "term":"wife"}'`

6. - Give me all entries for the canonical from *wife* for the property http://dbpedia.org/ontology/relative
   - `curl -H "Content-type: application/json" \-X GET http://matoll.sebastianwalter.org/matoll -d '{"uri":"http://dbpedia.org/ontology/relative", "term":"wife"}'`

7. - Give me all entries for the canonical form *wife* for the property http://dbpedia.org/ontology/spouse occurring at least 10 times
   - `curl -H "Content-type: application/json" \-X GET http://matoll.sebastianwalter.org/matoll -d '{"uri":"http://dbpedia.org/ontology/spouse", "term":"wife", "freq":"10"}'`

8. - Give me all entries for the property http://dbpedia.org/ontology/spouse in the German language
   - `curl -H "Content-type: application/json" \-X GET http://matoll.sebastianwalter.org/matoll -d '{"uri":"http://dbpedia.org/ontology/spouse", "lang":"DE"}'`

9. - Give me all entries for the property http://dbpedia.org/ontology/spouse in the Spanish language
   - `curl -H "Content-type: application/json" \-X GET http://matoll.sebastianwalter.org/matoll -d '{"uri":"http://dbpedia.org/ontology/spouse", "lang":"ES"}'`

To illustrate an example JSON output, consider the following call which returns all entries with the term `wife` and the frequency of 50 or greater for the English language for the property *dbo:spouse*:

```
curl -H "Content-type: application/json" \-X GET
http://matoll.sebastianwalter.org/matoll -d
'{"freq":"40", "term":"wife", "uri":"http://dbpedia.org/ontology/spouse"}'
```

The JSON output for this call looks as follows:

```
{
  "wife_1": {
    "Form": "wife",
    "Frame": "NounPPFrame",
    "Freq": 71,
    "Languages": "EN",
    "Obj": "http://lemon-model.net/lemon#subjOfProp",
    "Pattern": "SparqlPattern_EN_Noun_PP_appos",
    "Pos": "noun",
    "Prep": "of",
    "Subj": "http://lemon-model.net/lemon#objOfProp",
    "Uri": "http://dbpedia.org/property/spouse"
  }
}
```

Exactly one entry is returned with the canonical form `wife` and a `NounPPFrame`. Beside the information that the frequency of the entry is 71 (greater 50), also the information which preposition the entry contains is given among other information.

Chapter 8

Conclusion

With this chapter we conclude this thesis. We revisit the research questions raised in the introduction and elaborate on the four requirements an ontology lexicon has to fulfill. We close this chapter with an outlook on future work.

8.1 Conclusion

In this thesis we presented a framework comprising three different approaches to generate multilingual ontology lexica for a given ontology from a given text corpus.

8.1.1 Research Questions

In the beginning of this thesis (see Section 1.3 on page 9) we introduced the research questions we aimed to address in this work. In the following we want to summarize our main findings for each of these questions.

Do the results of the two main approaches, the label-based and corpus-based approach, complement each other?

Yes, the results complement each other. This is shown in Table 4.1 on page 66.

How do different filtering strategies affect the quality of the created lexicon in terms of precision, recall and F-measure?

Depending on the filtering strategy, the quality of the results differ. As baseline we used the standard frequency-based filtering strategy as proposed in Algorithm 4 on page 49. We evaluated three alternative strategies in Section 5.1. While the lemma-based filtering strategy did not improve the quality of the results, the property-based filtering strategy was able to increase the F-measure performance on the micro evaluation by 5% compared to the frequency-based filtering strategy. As the final strategy we tested a machine-learning based filtering strategy. After evaluating each feature independently and in combination with other features on different settings, with the best feature combination we could improve our F-measure results on the micro-based evaluation by 10% and on the macro-based evaluation by around 7%. Especially the last experiment showed that we can increase the quality of the lexicon by applying different filtering strategies without losing too much recall.

What are the advantages and disadvantages of M-ATOLL, compared to other state-of-the-art systems?

This question was answered throughout the whole thesis. The advantages are as follows:

- A combination of different approaches that complement each other. In particular an approach that is independent of property labels and therefore can be applied to extract lexicalizations in a multitude of languages.
- High quality grammatical patterns, which produce results with less noise than state-of-the-art systems that rely on the shortest path between entities or plain strings between entities.
- Generation of complex adjective lexicalizations for restriction class entries. This is currently not possible in other approaches.

- Support of multilinguality with three implemented languages: English, German and Spanish.

The disadvantages are as follows:

- Generation of hand-crafted patterns requires manual work. The whole system is thus not designed to be deployed in a fully unsupervised mode to a new language.
- Detection of literals, in particular numerical values, for datatype properties such as *dbo:evaluation* is challenging, but will be addressed in the future. This has a negative influence on the quality of the results.

What is the coverage of our hand-crafted patterns with respect to a lexical evaluation over a gold standard? Are there frequent patterns not covered by the ones defined by us?

In Section 4.6 we presented the coverage of each pattern on the final lexicon. For English and Spanish the most influential pattern was the *Noun PP copulative* pattern, which matches sentences like *Alice is the wife of Bob.* For German the most influential pattern is the intransitive verb pattern. Analyzed on the whole input corpus (only English), the most frequent pattern was *Intransitive verb with PP* (see Table 5.9 on page 82). The analysis also showed that 60% of all available patterns represented by the shortest path between two entities occur only once and therefore do not represent general enough patterns to be included in M-ATOLL. Ignoring those patterns and only considering patterns containing a subject and object dependency relation, we are still left with 21,557 pattern types, where the most frequent one represents abbreviations of our intransitive pattern. It is likely that that one of the 21,557 patterns could improve the coverage of M-ATOLL at the expense of potentially introducing further noise. Our evaluation showed that we already cover a lot of different lexical variants and also create high quality lexica.

8.1.2 Requirements

In the introduction (see Section 1.1 on page 2) we defined different requirements an ontology lexicon has to fulfill. In the following we elaborate on all four requirements and show that the lexica created by M-ATOLL fulfill those requirements.

The lexicon should contain lexical variants and it has to be enriched with linguistic information.

Throughout this thesis we showed that M-ATOLL creates lexica with different lexical variants - even more than presented in the goldstandard. See for example the verbalizations presented in Table 4.5 on page 68, Table 4.6 on page 68 and Table 5.14 on page 87, to mention a few. In addition, as introduced in Chapter 4 and Chapter 6, M-ATOLL extracts explicit linguistic information such as the part-of-speech, the syntactic frame and the mapping between semantic and syntactic arguments.

The lexicon needs to be generated with as little manual effort as possible, e.g. is created automatically or with the help of crowd sourcing.

We introduced M-ATOLL as an automatic approach to create ontology lexica. The results for the current languages are produced without the need for human input. However, we showed in the evaluation that it makes sense to use M-ATOLL in a semi-automatic fashion by asking a domain expert to remove incorrect lexical entries. In both modi, automatic and semi-automatic, the effort is much lower than creating all lexical entries, especially with this variance, manually.

If possible it should incorporate already existing resources, or link to them.

We reuse information from WordNet and Wiktionary in all our approaches. We also link to other resources such as *dbnary*[1] and *lemon UBY*[2].

It should serve different purposes. Further, it is important that the lexicon supports multilinguality and connects information between the different languages.

We support multilinguality by applying our methodology to three different languages: English, German and Spanish. Our approach relies on hand-crafted patterns which can be adapted and extended for other languages without the need for changing the framework itself. Our results can serve different purposes. This was shown in the evaluation in Section 4.6.1.

8.1.3 Conclusion

In this thesis we presented the framework M-ATOLL, which, given an ontology, a corresponding knowledge base, and a text corpus, generates lexicalizations for ontology elements (classes and properties) in multiple languages. As proof of concept we implemented M-ATOLL for English, German and Spanish, and applied it to DBpedia, using Wikipedia as text corpus. M-ATOLL relies on a set of predefined grammatical patterns, that are defined declaratively and thus are easy to extend and adapt. We were able to show that M-ATOLL produces good results for all three languages, being particularly useful when considering a semi-automatic scenario, in which automatically extracted lexicalization candidates are checked and, if necessary, corrected or discarded manually.

In this thesis we also presented an approach for extracting adjective lexicalizations with respect to an ontology by analyzing the labels of objects occurring in a given dataset. We showed that the features we selected for this task achieve a reasonable accuracy. We thus introduced a relatively low-cost and accurate method for inducing an adjective lexicon, which can be straightforwardly applied to other ontologies and languages.

We gave a deep analysis of the goldstandard in the context of the given text corpus as well as a detailed analysis of different filtering strategies for the main approach of M-ATOLL. Throughout the thesis we gathered different ideas to even further improve the coverage as well as quality of

[1]See http://kaiko.getalp.org/about-dbnary/development/
[2]See http://www.lemon-model.net/lexica/uby/

the results of our framework. We will close this thesis with presenting our ideas for future work in the following section.

8.2 Outlook

We present our ideas for the future work separately for each approach.

8.2.1 Label-based approach

To improve the coverage of the label-based approach, we plan to extend it with additional resources, such as *BabelNet* by Navigli and Ponzetto (2012). Additionally, the overall system would benefit from a detailed analysis of different word sense disambiguation strategies in order to choose the correct WordNet (or BabelNet) synset. However, the main focus will be to port this approach to other languages, in particular to German and Spanish.

8.2.2 Corpus-based approach

The main focus will be on improving the coverage for datatype properties. This requirers an sophisticated matching strategy for data types. Generally speaking, we want to improve the different matching of entity mentions and literals (such as dates and numerical measures) in sentences, in order to improve performance on datatype properties. As an example, consider the property *dbo:birthDate* with a `date` as data type. Currently, we reduce this date to the actual year ignoring the day and month. In the future we will replace this simple heuristic with *HeidelTime* by Strötgen and Gertz (2015). We will introduce a unit conversion to improve the mapping between the ontology knowledge base and the actual mention in the text. Preliminary analysis showed that the units in the knowledge base are not always the same as in our text corpus, therefore sometimes wrong sentences are retrieved or correct sentences are missed.

Moreover, M-ATOLL would benefit from a better sentence selection strategy for example using one of the coreference resolution systems presented in Section 3.1.5. In M-ATOLL we implemented a bunch of heuristics in order to resolve pronouns in a sentence to their antecedent, for example in `Cinderella is a poor girl but she fell in love with a prince`. Due to time constraints, however, we did not consider this strategy while evaluating our approach.

Currently we do not consider M-ATOLL in the context of an Open Information Extraction system. As recap, a system is considered to be an OIE system if structured relations are extracted, without specifying the structure of the pattern itself. We could use the extracted relations, including the data from the knowledge base, as seed instances in order to extract automatically more relevant patterns, without the need to hand-craft them. The idea is to apply our lexical entries to dependency parsed sentences. Whenever the relation as well as the subject and object of the corresponding triple is found, we save the dependency path between the two entities, describing this relation. After generalizing these paths, the most frequent paths could be considered as valid patterns, which could then be used as additional seed instances.

In the future we will also investigate the combination of M-ATOLL with other approaches e.g. *Patty* (which lacks linguistic information).

This approach already supports multilinguality, however, we still do not have a good goldstandard for Spanish and German. Therefore, one goal for the future is to create a goldstandard lexicon for both, Spanish and German, so that we can evaluate M-ATOLL properly on both languages.

8.2.3 Adjective approach

The main future work for this approach is to extend this approach to more languages. It would be nice if this can be done by reusing the features we already introduced in Section 6. For example, it would be interesting to see how well the trained model for the English language performs on other languages. With this analysis we could try to design this approach using only language-independent features, enabling the easy support for many more languages.

This approach would also benefit from a more extensive evaluation. Currently, we only provided some insights by discussing the results from a few properties as well as evaluating the impact of each feature.

Our ideas on how to improve M-ATOLL clearly show how interesting this research area is and how much potential our framework still provides for further research. Very important for us is to investigate further the impact of M-ATOLL lexica on NLP task, such as a QA system.

Appendix A

A.1 Appendix: Templates

A.1.1 Class Template

(A.1)
```
LexicalEntry entry = new LexicalEntry(Language.EN);
entry.setCanonicalForm("action");
Sense sense = new Sense();
Reference ref = new SimpleReference("http://dbpedia.org/ontology/Activity");
sense.setReference(ref);

Provenance provenance = new Provenance();
provenance.setFrequency(1);

sense.setReference(ref);

entry.setURI(lexicon.getBaseURI()+"LexicalEntry_action_as_WordnetClassEntry");

entry.setPOS("http://www.lexinfo.net/ontology/2.0/lexinfo#commonNoun");

SyntacticBehaviour behaviour = new SyntacticBehaviour();
behaviour.setFrame("http://www.lexinfo.net/ontology/2.0/lexinfo#NounPPFrame");

behaviour.add(new SyntacticArgument(
                "http://www.lexinfo.net/ontology/2.0/lexinfo#directObject",
                "object",
                null));
behaviour.add(new SyntacticArgument(
                "http://www.lexinfo.net/ontology/2.0/lexinfo#subject",
                "subject",
                null));

sense.addSenseArg(new SenseArgument(
                "http://lemon-model.net/lemon#subjOfProp",
                "subject"));
sense.addSenseArg(new SenseArgument(
                "http://lemon-model.net/lemon#objOfProp",
                "object"));

entry.addSyntacticBehaviour(behaviour,sense);

entry.addProvenance(provenance,sense);

lexicon.addEntry(entry);
```

A.1.2 Copulative Noun Template

(A.2)
```
LexicalEntry entry = new LexicalEntry(Language.EN);
entry.setPreposition(new Preposition(Language.EN,"of"));

Sense sense = new Sense();

Reference ref = new SimpleReference("http://dbpedia.org/ontology/spouse");
sense.setReference(ref);

Provenance provenance = new Provenance();
provenance.addPattern("copulative_noun_pp");
provenance.setFrequency(1);
provenance.addSentence(
        "Poet Rudra Mohammad Shahidullah was the husband of writer Taslima Nasrin");
```

```
SyntacticBehaviour behaviour = new SyntacticBehaviour();
behaviour.setFrame("http://www.lexinfo.net/ontology/2.0/lexinfo#NounPPFrame");
entry.setCanonicalForm("husband");
entry.setURI(lexicon.getBaseURI()+"LexicalEntry_husband_as_Noun_withPrep_of");
entry.setPOS("http://www.lexinfo.net/ontology/2.0/lexinfo#commonNoun");


if (a1.equals("http://lemon-model.net/lemon#subjOfProp")
&& a2.equals("http://lemon-model.net/lemon#objOfProp"))
{

        behaviour.add(new SyntacticArgument(
                        "http://www.lexinfo.net/ontology/2.0/lexinfo#prepositionalObject",
                        "object",
                        "of"));
        behaviour.add(new SyntacticArgument(
                        "http://www.lexinfo.net/ontology/2.0/lexinfo#copulativeArg",
                        "subject",
                        null));

        sense.addSenseArg(new SenseArgument(
                        "http://lemon-model.net/lemon#subjOfProp",
                        "subject"));
        sense.addSenseArg(new SenseArgument(
                        "http://lemon-model.net/lemon#objOfProp",
                        "object"));

        entry.addSyntacticBehaviour(behaviour,sense);
        entry.addProvenance(provenance,sense);
                   lexicon.addEntry(entry);
}

else if (a1.equals("http://lemon-model.net/lemon#objOfProp")
&& a2.equals("http://lemon-model.net/lemon#subjOfProp"))
{

        behaviour.add(new SyntacticArgument(
                        "http://www.lexinfo.net/ontology/2.0/lexinfo#adpositionalObject",
                        "subject",
                        "of"));
        behaviour.add(new SyntacticArgument(
                        "http://www.lexinfo.net/ontology/2.0/lexinfo#copulativeArg",
                        "object",
                        null));

        sense.addSenseArg(new SenseArgument(
                        "http://lemon-model.net/lemon#subjOfProp",
                        "object"));
        sense.addSenseArg(new SenseArgument(
                        "http://lemon-model.net/lemon#objOfProp",
                        "subject"));

        entry.addSyntacticBehaviour(behaviour,sense);
        entry.addProvenance(provenance,sense);
        lexicon.addEntry(entry);
}
```

A.2 Appendix: Feature Combinations

Feature Combination	Accuracy
NFP, FREC, HP, PP, PREP, DR, LM, NFL, ARG, SP	0.9079
NFP, FREC, PP, PREP, DR, LM, NFL, SP	0.9063
NFP, FREC, HP, PP, PREP, DR, LM, NFL, SP	0.9063
NFP, FREC, PP, PREP, DR, LM, NFL, ARG, SP	0.9055
NFP, FREC, HP, PREP, DR, LM, NFL, SP	0.9048
NFP, FREC, PP, DR, LM, NFL, SP	0.9008
NFP, PP, PREP, DR, LM, NFL, TYPE, ARG, SP	0.9001
NFP, FREC, HP, PP, DR, LM, NFL, SP	0.900
NFP, PREP, DR, LM, NFL, SP	0.8992
NFP, HP, PREP, DR, LM, NFL, SP	0.8992

Table A.1: Accuracy for top 10 feature combinations for the Ada Boost

Feature Combination	Accuracy
NFP, FREC, HP, PP, DR, LM, NFL, SP	0.8969
NFP, HP, PREP, DR, LM, NFL, ARG, BOW	0.8969
NFP, HP, PREP, DR, LM, NFL, SP	0.8968
NFP, FREC, PP, DR, LM, NFL, ARG	0.8959
NFP, FREC, HP, PP, DR, LM, ARG, SP	0.8959
NFP, FREC, PREP, DR, NFL, TYPE, ARG, SP, BOW	0.8947
NFP, FREC, PP, DR, NFL, SP, BOW	0.8946
NFP, HP, PP, DR, LM, NFL, SP	0.8945
NFP, HP, PP, PREP, DR, LM, NFL, ARG	0.8944
NFP, PP, DR, LM, NFL, ARG, SP	0.8936

Table A.2: Accuracy for top 10 feature combinations for the Perceptron.

Feature Combination	Accuracy
NFP, FREC, HP, PP, PREP, DR, LM, NFL, TYPE, ARG, SP	0.8922
NFP, FREC, HP, PP, PREP, DR, LM, NFL, ARG, SP	0.8915
NFP, FREC, PP, PREP, DR, LM, NFL, ARG, SP	0.8915
NFP, FREC, HP, PREP, DR, LM, NFL, TYPE, ARG, SP	0.8907
NFP, FREC, PREP, DR, LM, NFL, TYPE, ARG, SP	0.8907
NFP, FREC, PREP, DR, LM, NFL, ARG, SP	0.8907
NFP, FREC, HP, PP, PREP, DR, LM, NFL, ARG	0.8907
NFP, FREC, HP, PREP, DR, LM, NFL, ARG, SP	0.8907
NFP, FREC, DR, LM, NFL	0.8899
NFP, FREC, PP, PREP, DR, LM, NFL, SP	0.8891

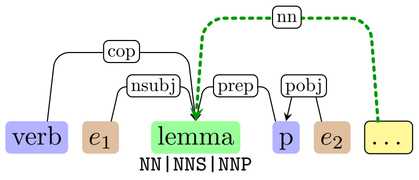
Table A.3: Accuracy (in %) for top 10 feature combinations for the SVC.

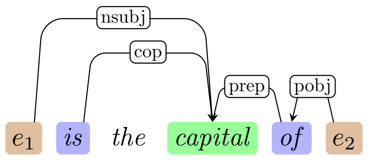
A.3 Appendix: Linguistic patterns

Note that the patterns looks slightly different across languages due to differences in the vocabulary and structure of the dependency parse.

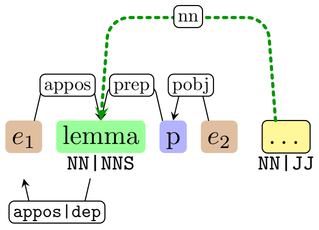
A.3.1 English patterns

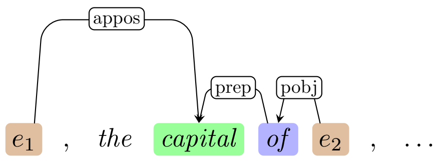
Noun with PP (copulative)



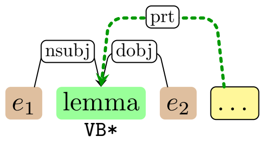


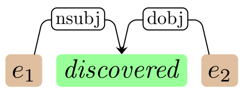
Noun with PP (appositive)



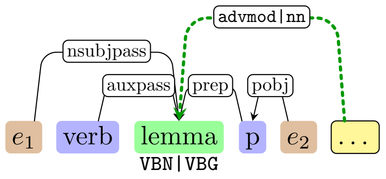


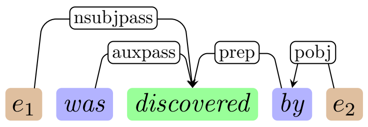
Transitive verb (active)



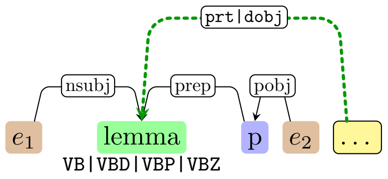


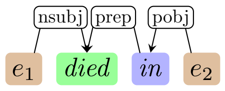
Transitive verb (passive)



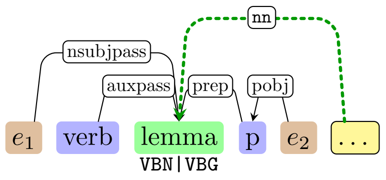


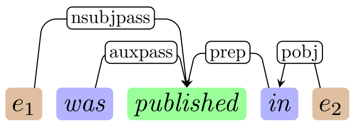
Intransitive verb with PP



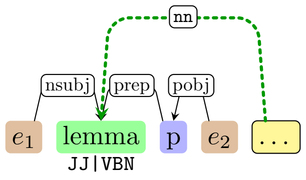


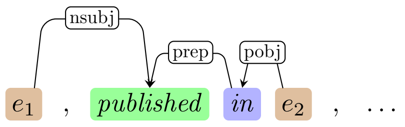
Past participle (copulative)





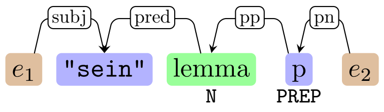
Past participle (appositive)

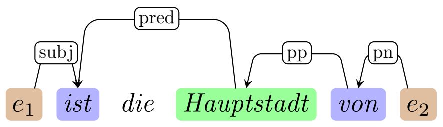




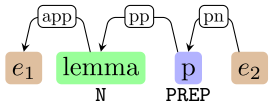
A.3.2 German patterns

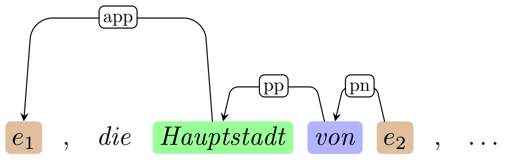
Noun with PP (copulative)



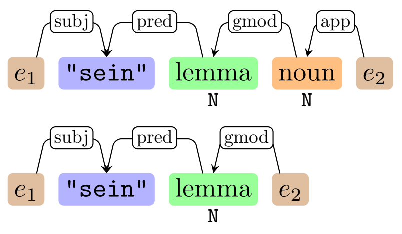


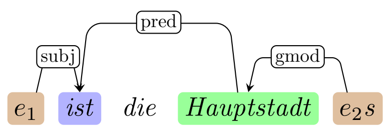
Noun with PP (appositive)



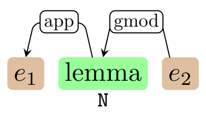


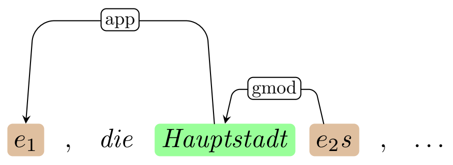
Noun with possessive (copulative)



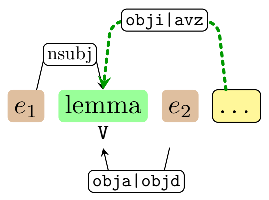


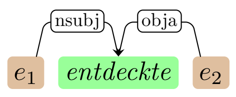
Noun with possessive (appositive)



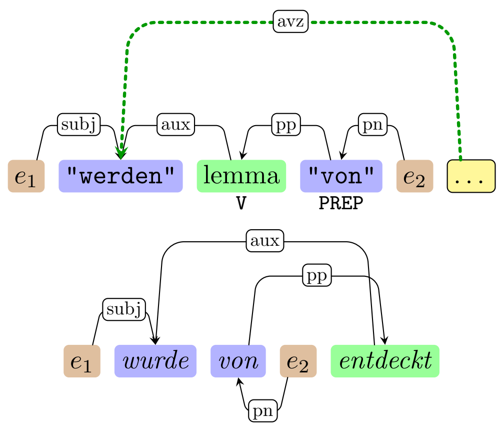


Transitive verb (active)

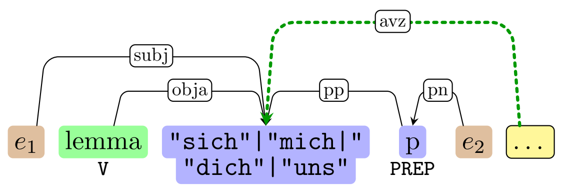


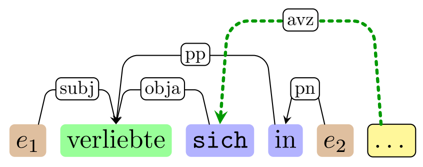


Transitive verb (passive)

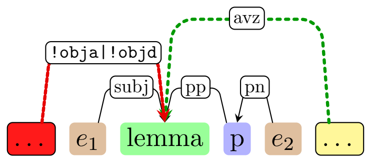


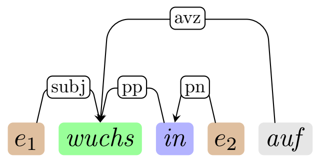
Reflexive transitive verb with PP



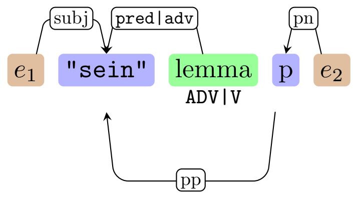


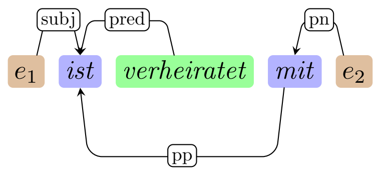
Intransitive verb with PP





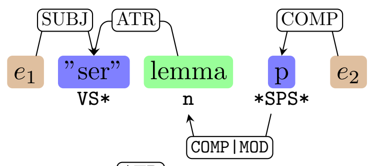
Past participle (copulative)

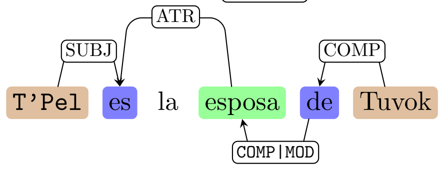




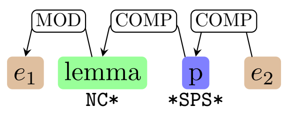
A.3.3 Spanish patterns

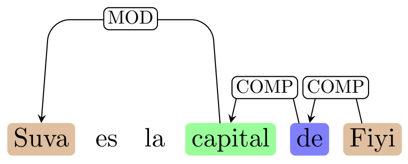
Noun with PP (copulative)

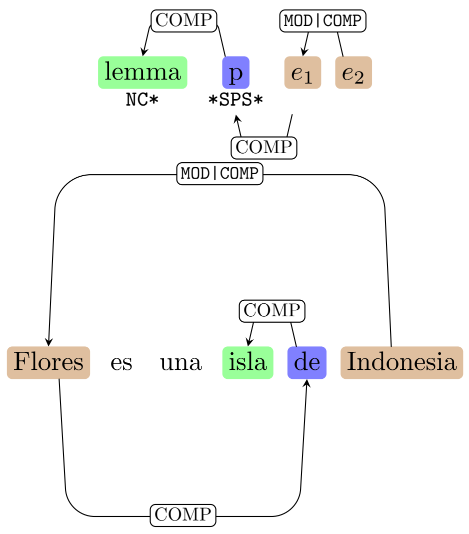




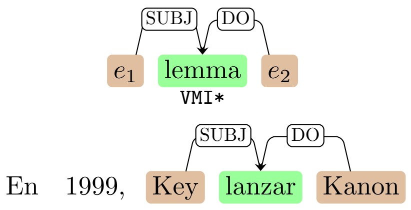
Noun with PP (appositive)



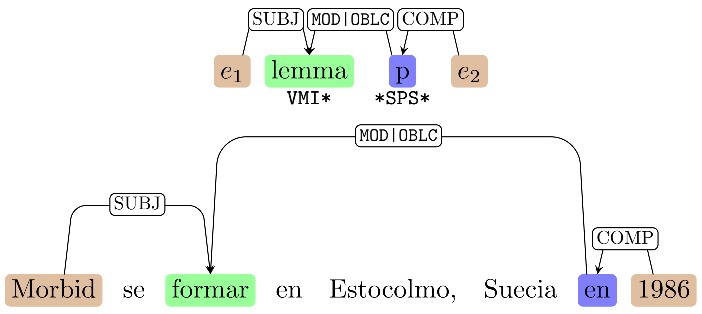


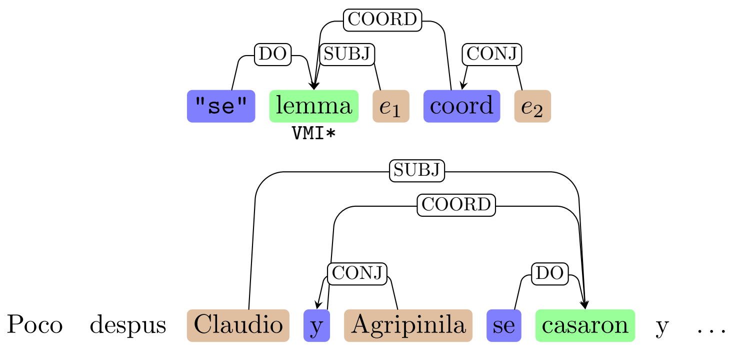


Transitive verb (active)

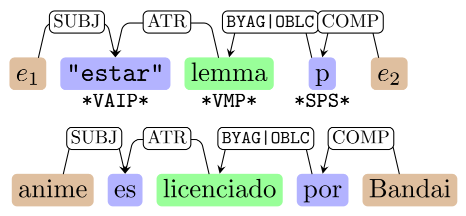


Reflexive transitive verb with PP

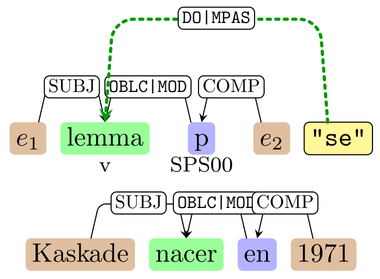




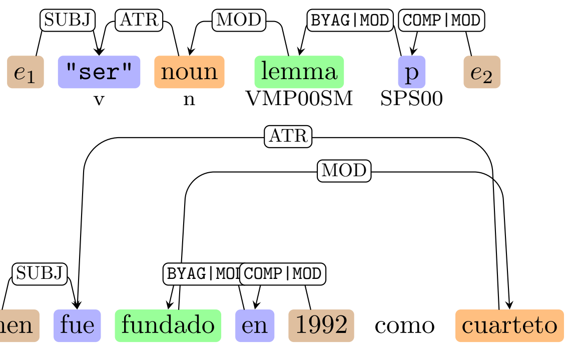
Reflexive transitive verb with PP (passive)



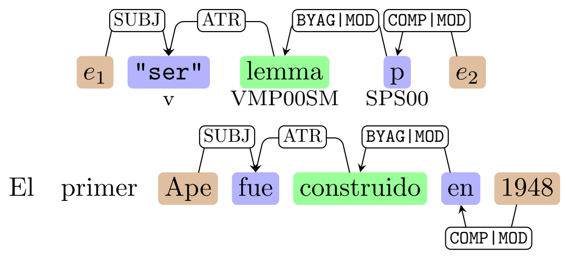
Intransitive verb with PP



Past participle (copulative)



Past participle (appositive)



A.4 Appendix: Implementation

A.4.1 External APIs

M-ATOLL uses the following external external APIs.

- WEKA by Hall et al. (2009) was used for the Random Forest experiments presented in Chapter 6.
- WordNet by Miller (1995) with an implementation from Finlayson (2014) was used in the Label-based approach presented in Chapter 4.
- Scikit-Learn by Pedregosa et al. (2011) and Buitinck et al. (2013) was used to evaluate the machine learning-based filtering strategy for M-ATOLL as presented in Chapter 5.
- The MaltParser by Nivre et al. (2006) with its pre-trained model for the English language was used to dependency parse the text corpus.
- For Spanish we used a Spanish MaltParser instance created by Marimon and Bel (2015) and Padró et al. (2013).
- TreeTagger by Schmid (1995, 1994) was used for the preprocessing of German sentences, which then were parsed with the ParzuParser by Sennrich et al. (2009, 2013).
- Lucene[1] by the Apache Software Foundation in order to store and provide a fast access to the parsed sentences.
- Jena API[2], also by the Apache Software Foundation, was used as core element in M-ATOLL to create, load and serialize lexical entries in *lemon* format.

A.4.2 lemon API

As mentioned in Chapter 2.2.2, we implemented an API for serializing lexicalizations in *lemon* format. Our implementation is presented in Figure A.1. We introduced some changes compared to the original *lemon core* which are highlighted by the green and red colors. For each sense M-ATOLL stores additional information in form of a provenance. A provenance contains the frequency of an entry among others as described in Chapter 4.

The API can be used to load, create, manipulate and serialize *lemon* lexica. In this section we present some examples of how to use the API. The current version can be downloaded http://dblexipedia.org/public/MATOLL.jar. It is also possible to build the API directly from the source code at https://github.com/ag-sc/matoll by running

```
mvn clean && mvn install
```

in the subfolder *matoll*. In the following we describe the core classes and their functions.

[1]See https://lucene.apache.org/core/

[2]See https://jena.apache.org

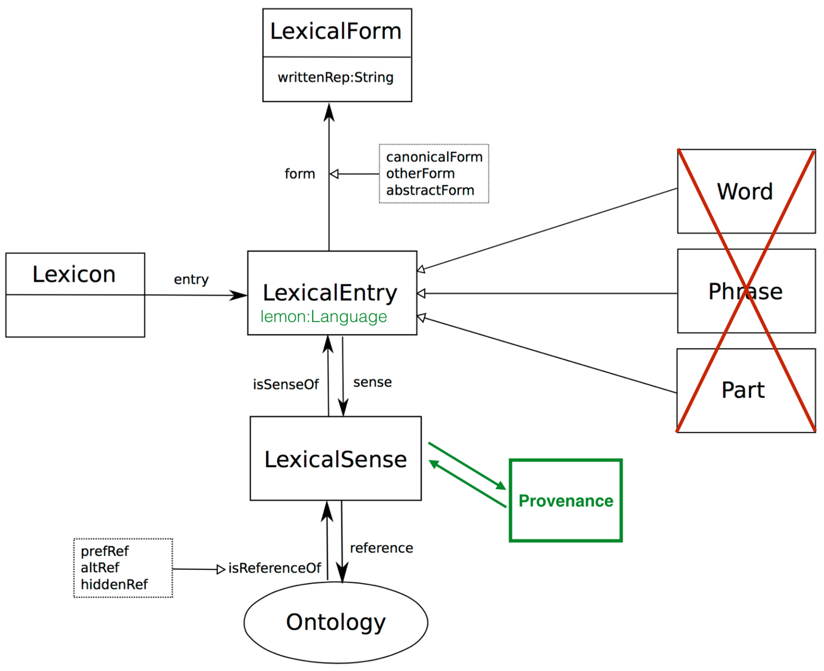


Figure A.1: The *lemon* core as implemented in this thesis

LexicalEntry

The class *LexicalEntry* is one of the core classes of the API. With this class *lemon* entries are generated.

```
LexicalEntry entry = new LexicalEntry(language);
```

initializes a lexical entry for a given language. Currently M-ATOLL supports four different languages:

```
public enum Language {
    EN, DE, ES, JA
}
```

For each entry the canonical form, the POS tag and the URI of the entry have to be specified, e.g. as follows:

```
setCanonicalForm("wife")
setPOS("http://www.lexinfo.net/ontology/2.0/lexinfo#commonNoun")
setURI("http://dblexipedia.org/LexicalEntry_wife_as_Noun_withPrep_of")
```

The URI serves as a unique identifier of the entry. Additionally, each lexical entry contains one or more senses, with a reference to the ontology element(s) that is verbalized. M-ATOLL implements two different types of references: simple reference pointing to a URI and references for restriction classes, as used by the adjectives in Section 6.

- Simple reference:

```
Sense sense = new Sense();
Reference ref = new SimpleReference
        ("http://dbpedia.org/ontology/spouse");
sense.setReference(ref);
```

- Reference for a restriction class

```
Sense sense = new Sense();
Reference ref = new Restriction
        (lexicon.getBaseURI()
        +"RestrictionClass_gender_female",
        "http://dbpedia.org/resource/female",
        "http://dbpedia.org/ontology/gender");
sense.setReference(ref);
```

While creating a lexical entry it is also possible to define the syntactic behaviour of this entry which is then bound to the sense.

```
SyntacticBehaviour behaviour  = new SyntacticBehaviour();
String lexinfo = "http://www.lexinfo.net/ontology/2.0/lexinfo#";
String \emph{lemon} = "http://lemon-model.net/lemon#";

behaviour.setFrame(lexinfo + "NounPossessiveFrame");

behaviour.add(new SyntacticArgument
        (lexinfo + "prepositionalObject",
        "object",
        preposition));

behaviour.add(new SyntacticArgument
        (lexinfo + "copulativeArg",
        "subject",
        null));

sense.addSenseArg(new SenseArgument(lemon + "subjOfProp","subject"));

sense.addSenseArg(new SenseArgument(lemon + "objOfProp","object"));

entry.addSyntacticBehaviour(behaviour,sense);
```

We use *lexinfo* as vocabulary for the frames, however, this can be simply changed to any other linguistic ontology.

By iterating over the pairs of SyntacticBehaviour and Sense, the URI of the lexicalized property can be retrieved.

```
for(Sense sense : entry.getSenseBehaviours().keySet()){
     Reference ref = sense.getReference();
     //for a simple reference
     if (ref instanceof de.citec.sc.matoll.core.SimpleReference)
                System.out.println(ref.getURI());

     //for a restriction class
     if (ref instanceof de.citec.sc.matoll.core.Restriction)
          System.out.println(refgetProperty());
}
```

Each lexical entry can also be linked to provenance information:

```
Provenance provenance = new Provenance();
provenance.setFrequency(1);
entry.addProvenance(provenance,sense);
```

Lexicon

If not empty, a lexicon consists of a set of lexical entries. A new lexical entry is added to the lexicon with

```
lexicon.addEntry(entry)
```

Based on the URI of the lexical entry it is automatically verified whether the entry already exists or not. If so, both are merged.

The algorithm for adding a lexical entry to an existing lexicon is presented in Algorithm 6 and works as follows: When an entry is added (called e_1 in in Algorithm 6), based on the unique URI of the entry (set during creation of the entry) it is validated whether the entry is already in the lexicon or not. If not, the entry is simply added to the lexicon (see line 3). If the entry is already in the lexicon, the entry in the lexicon is retrieved (line 5, called e_2 in Algorithm 6) and the sense of the e_1 is collected (line 6). If the sense of e_1 is the same as the sense of e_2 only the provenance information is updated, e.g. upgrading the frequency. If the sense of e_1 is not the same as the sense from e_2, also the sense of e_1 is added to e_2.

The lexicon class has some build-in functions to retrieve different lexical entries:

- *lexicon.getEntries()* retrieves all lexical entries
- *lexicon.getEntries('EN')* retrieves all lexical entries for a the English language
- *lexicon.getEntriesWithCanonicalForm("wife")* retrieves all lexical entries with a certain canonical form, e.g `wife`

Algorithm 6 How to add a lexical entry to an existing lexicon

```
Require: Lexicon, lexical entry
 1: for lexical entry $e\_1$ do
 2:     if $e\_1$ not in Lexicon then
 3:         add $e\_1$ to Lexicon
 4:     else
 5:         $e\_2 \leftarrow getLexicalEntry$(e_1)
 6:         $sense \leftarrow getSense$(e_1)
 7:         if sense not in $e\_2$ then
 8:             add $e\_1$ to Lexicon
 9:         else
10:             $provenance1 \leftarrow getProvenance$(e_1, $sense$)
11:             $provenance2 \leftarrow getProvenance$(e_2, $sense$)
12:             Update provenance2 with provenance1
13:         end if
14:     end if
15: end for
16: return Lexicon
```

- *lexicon.getEntriesForReference(”http://dbpedia.org/ontology/spouse”)* retrieves all lexical entries for a certain property, e.g. *dbo:spouse*

Additionally some other helpful functions are available:

- *lexicon.getPrepositions()* returns all prepositions occurring in lexical entries, e.g. `to`, `for`, `while`, etc.
- *lexicon.getReferences()* returns all references (e.g. *dbo:spouse*) in a given lexicon.
- *lexicon.size()* returns the number of lexical entries. Note that as one entry can lexicalize multiple properties (i.e. have multiple references), it is possible to have more references than lexical entries. And because several entries can lexicalize the same property, it is also possible to have less references than entries.
- *lexicon.setBaseURI(http://localhost:8080/)* sets the baseURI for the lexicon to `http://localhost:8080`. If this function is not used, the baseURI is automatically set to `http://dblexipedia.org/`

LexiconLoader

The lexiconloader allows to load a lexicon from a file:

```
LexiconLoader loader = new LexiconLoader();
Lexicon lexicon = loader.loadFromFile("example.ttl");
```

Afterwards this lexicon can be used as described above.

LexiconSerialization

With the help of the class *LexiconSerialization* a lexicon is written to an RDF file.

```
LexiconSerialization serializer = new LexiconSerialization(true);
Model model = ModelFactory.createDefaultModel();

serializer.serialize(lexicon, model);

FileOutputStream out = new FileOutputStream(new File("lexicon.ttl"));
RDFDataMgr.write(out, model, RDFFormat.TURTLE) ;
out.close();
```

When initializing the LexiconSerialization

```
LexiconSerialization serializer = new LexiconSerialization(true);
```

the Boolean argument enables or disables the function to automatically remove those lexical entries where the canonical form is a stopword.

A.4.3 Running the Corpus-based Approach

Preprocessing the text corpus

As explained in Chapter 4, the corpus-based approach takes a parsed text corpus, converted into RDF, as input. The source code for this step is stored in the subproject *SentencePreprocessing*.

Assuming a dependency-parsed corpus exists and is stored in a Lucene index, the following steps have to be carried out in order to create the necessary input for M-ATOLL: First it is necessary to retrieve the entities from a given knowledge base. There are two possible ways to do so. You can either use M-ATOLL by providing a SPARQL endpoint, such as http://dbpedia.org/sparql, and uncomment the following line in the *Process.java* file

```
de.citec.sc.sentence.preprocessing.
sparql.Resources.retrieveEntities(properties,
        folderToSaveResourcesSentences, endpoint, language);
```

However, as some endpoints only return a limited amount of entities per query, it is advised to do this offline by downloading the files and creating the entity files manually. In the folder *PythonSkript* an example how to create the necessary files is given.

If all files are prepared the whole process can be enabled by

```
mvn clean && mvn install
mvn exec:java
        -Dexec.mainClass=
                "de.citec.sc.sentence.preprocessing.process.Process"
        -Dexec.args="config.xml"
```

with a given config file, such as the follows:

```
1 <Config>
2    <Language>DE</Language>
3    <Index>PathToIndex</Index>
4    <Input>dbpedia_2014.owl</Input>
5    <Output>PathToOutput</Output>
6    <Endpoint>http://dbpedia.org/sparql</Endpoint>
7    <WithSentences>True</WithSentences>
8    <AdditionalOutput>True</AdditionalOutput>
9 </Config>
```

line 7 When uncommenting the above mention line in the source-code and setting the Boolean value of this line to *False*, only the entities are retrieved. When setting the Boolean value of this line to *True* the sentences are extracted, transformed to RDF and saved.

line 8 When setting the Boolean value to *True*, a human readable version of the parsed sentences is written, additionally to the transformed sentences in RDF. This is only advised for debugging M-ATOLL and should be set to *False* in normal use.

After completing the extraction process, the generated files are ready to be used as input to M-ATOLL.

Lexicalizing an ontology

The easiest way to run the ontology lexicalization is using maven with the following commands:

```
mvn clean && mvn install
mvn exec:java
        -Dexec.mainClass="de.citec.sc.matoll.process.Matoll"
        -Dexec.args=
        "--mode=train /path/to/inputFiles/ /path/to/config.xml"
```

where */path/to/inputFiles/* stands for the path where the corpus files from the preprocessing step are stored.

The following is an example for the config.xml

```
<Config>
   <Language>EN</Language>
   <Coreference>False</Coreference>
   <GoldStandardLexicon>../lexica/dbpedia_en.rdf</GoldStandardLexicon>
   <OutputLexicon>dbpedia2014Full_new.lex</OutputLexicon>
   <Output>dbpedia2014.eval</Output>
   <NumLexItems>10000</NumLexItems>
   <RemoveStopwords>True</RemoveStopwords>
   <BaseURI>http://localhost:8080/</BaseURI>
</Config>
```

A.4.4 Running the Label-based Approach

This approach is started with the following maven command:

```
mvn clean && mvn install
mvn exec:java
        -Dexec.mainClass="de.citec.sc.matoll.LabelApproach.Process"
```

It automatically runs the label-based approach for the properties and classes as well as the adjective approach for restriction classes. To do so, only the path to the ontology and the path to the Lucene index have to be included. The RandomForest model for the adjective approach is automatically built and used.

A.4.5 Porting across Domains and Languages

In order to port M-ATOLL to other domains or languages, some adjustments are necessary. The changes to port M-ATOLL to another domain are the following:

1. Pick an ontology and a corresponding knowledge base that cover the target domain.
2. Compile a text corpus that matches the target domain and parse it with one of the included dependency parsers (MaltParser for English and Spanish, ParZu for German). If another dependency parser is used, the patterns have to be adapted to the vocabulary and parse structures of that parser.

Porting to another language requires the following steps:

1. Compile a text corpus in the target language and parse it with a dependency parser for that language.
2. Where necessary, adapt the preprocessing steps, in particular for matching named entity mentions in parsed sentences.
3. Design dependency patterns for the target language, on the basis of the vocabulary and parse structures of the dependency parser.

Bibliography

Agichtein, E. and L. Gravano (2000), “Snowball: Extracting relations from large plain-text collections.” In *Proceedings of the fifth ACM conference on Digital libraries, San Antonio, Texas, USA*. 34

Akbik, A. and J. Broß (2009), “Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns.” In *Proceedings of the Workshop on Semantic Search in Conjunction with the 18th Int. World Wide Web Conference (WWW), Madrid, Spain*. 34

Almuhareb, A. and M. Poesio (2004), “Attribute-Based and Value-Based Clustering: An Evaluation.” In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain*. 41

Angeli, G., M. J. Premkumar, and C. D. Manning (2015), “Leveraging linguistic structure for open domain information extraction.” *Linguistics*. 33

Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives (2007), “Joint proceedings of the 6th international semantic web conference and 2nd asian semantic web conference (iswc 2007 + aswc 2007) busan, korea.” 4, 16

Berners-Lee, T. (2006), “Linked Data.” URL http://www.w3.org/DesignIssues/LinkedData.html. 15

Berners-Lee, T., J. Hendler, O. Lassila, et al. (2001), “The semantic web.” *Scientific american*, 284, 28–37. 14

Blohm, S. and P. Cimiano (2007), “Using the Web to Reduce Data Sparseness in Pattern-Based Information Extraction.” In *Knowledge Discovery in Databases: PKDD 2007*, volume 4702 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg. 33

Boguraev, B. and J. Pustejovsky (1996), *Corpus Processing for Lexical Acquisition*. MIT Press, Cambridge, MA, USA. 40

Boleda Torrent, G. and L. Alonso i Alemany (2003), “Clustering Adjectives for Class Acquisition.” In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics (EACL ’03), Budapest, Hungary*. 40

Boleda Torrent, G., T. Badia, and E. Batlle (2004), “Acquisition of Semantic Classes for Adjectives from Distributional Evidence.” In *Proceedings of the 20th International Conference on Computational Linguistics (COLING’04), Geneva, Switzerland.* 41

Bourigault, D. and C. Jacquemin (1999), “Term Extraction + Term Clustering: An Integrated Platform for Computer-aided Terminology.” In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics (EACL ’99), Bergen, Norway.* 38

Buitelaar, P. and T. Eigner (2009), “Expertise Mining from Scientific Literature.” In *Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP ’09), Redondo Beach, California, USA.* 38

Buitinck, L., G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux (2013), “API design for machine learning software: experiences from the scikit-learn project.” In *Proceedings of ECML PKDD Workshop: Languages for Data Mining and Machine Learning, Prague, Czech Republic.* 137

Cabrio, E., P. Cimiano, V. Lopez, A.-C. N. Ngomo, C. Unger, and S. Walter (2013), “QALD-3: Multilingual Question Answering over Linked Data.” In *CLEF (Working Notes).* 38

Cabrio, E., J. Cojan, A. P. Aprosio, B. Magnini, A. Lavelli, and F. Gandon (2012), “QAKiS: an open domain QA system based on relational patterns.” In *Proceedings of the 11th International Semantic Web Conference (ISWC 2012), Boston, USA.* 2, 40

Chen, K.-h. and H.-H. Chen (1994), “Corpus-Based Analyses of Adjectives: Automatic Clustering.” In *Proceedings of the International Conference on Quantitative Linguistics (IQLA), Moscow, Russia.* 40

Cimiano, P., P. Buitelaar, J. McCrae, and M. Sintek (2011), “LexInfo: A declarative model for the lexicon-ontology interface.” *Web Semantics: Science, Services and Agents on the World Wide Web*, 9, 29–51. 26

Cimiano, P., V. Lopez, C. Unger, E. Cabrio, A.-C. Ngonga Ngomo, and S. Walter (2013), “Multilingual Question Answering over Linked Data (QALD-3): Lab Overview.” In *Proceedings of 4th International Conference of the CLEF Initiative (CLEF 2013) Valencia, Spain.* 39

Corro, L. D. and R. Gemulla (2013), “ClausIE: Clause-Based Open Information Extraction.” In *Proceedings of the 22th International Conference on World Wide Web (WWW), Rio de Janeiro, Brazil.* 34

Cucerzan, S. and E. Agichtein (2005), “Factoid Question Answering over Unstructured and Structured Web Content.” In *Proceedings of the 4th Text REtrieval Conference (TREC), Gaithersburg, Maryland, USA.* 40

de Marneffe, M.-C., B. MacCartney, and C. D. Manning (2006), “Generating typed dependency parses from phrase structure parses.” In *Proceedings International Conference on Language Resources and Evaluation (LREC), Genoa, Itay.* 28

Derczynski, L., D. Maynard, G. Rizzo, M. van Erp, G. Gorrell, R. Troncy, J. Petrak, and K. Bontcheva (2015), "Analysis of named entity recognition and linking for tweets." *Information Processing & Management*, 51. 36

Durrett, G., D. Hall, and D. Klein (2013), "Decentralized Entity-Level Modeling for Coreference Resolution." In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL), Sofia, Bulgaria.* 37

Durrett, G. and D. Klein (2013), "Easy Victories and Uphill Battles in Coreference Resolution." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Seattle, Washington, USA.* 37

Fader, A., S. Soderland, and O. Etzioni (2011), "Identifying relations for open information extraction." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Edinburgh, United Kingdom.* 34

Ferrucci, D., E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. (2010), "Building Watson: An overview of the DeepQA project." *AI magazine*, 31. 2

Finkel, J. R., T. Grenager, and C. Manning (2005), "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling." In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05), Michigan, USA.* 35

Finlayson, M. A. (2014), "Java libraries for accessing the princeton wordnet: Comparison and evaluation." In *Proceedings of the 7th Global Wordnet Conference, Tartu, Estonia.* 137

Gabrilovich, E. and S. Markovitch (2007), "Computing semantic relatedness using Wikipedia-based explicit semantic analysis." In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), Hyderabad, India.* 40

Gerber, D. and A.-C. N. Ngomo (2011), "Bootstrapping the linked data web." In *Proceedings of the 1st Workshop on Web Scale Knowledge Extraction, workshop co-located with the 10th International Semantic Web Conference (ISWC 2011), Bonn, Germany.* 33, 60, 86, 95

Gerber, D. and A.-C. N. Ngomo (2012), "Extracting multilingual natural-language patterns for rdf predicates." In *Knowledge Engineering and Knowledge Management*, Springer. 33, 60

Gilles, S. (2015), "DBnary: Wiktionary as a Lemon-based multilingual lexical resource in RDF." *Semantic Web*, 6. 97

Girju, R. (2003), "Automatic Detection of Causal relations for Question Answering." In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering, Sapporo, Japan.* 34, 40

Graves, M., A. Constabaris, and D. Brickley (2007), "Foaf: Connecting people on the semantic web." *Cataloging & classification quarterly*, 43. 16

Gruber, T. R. (1995), "Toward principles for the design of ontologies used for knowledge sharing?" *International journal of human-computer studies*, 43. 20

Hakimov, S., H. Tunc, M. Akimaliev, and E. Dogdu (2013), "Semantic Question Answering System over Linked Data Using Relational Patterns." In *Proceedings of the Joint EDBT/ICDT 2013 Workshops (EDBT '13), Genoa, Italy.* 40

Hakimov, S., C. Unger, S. Walter, and P. Cimiano (2015), "Applying Semantic Parsing to Question Answering Over Linked Data: Addressing the Lexical Gap." In *Proceedings of the 20th International Conference on Applications of Natural Language to Information Systems (NLDB 2015) Passau, Germany.* 40

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten (2009), "The WEKA Data Mining Software: An Update." *SIGKDD Explor. Newsl.*, 11. 105, 137

Hartung, M. and A. Frank (2010a), "A Semi-supervised Type-based Classification of Adjectives: Distinguishing Properties and Relations." In *Proceedings of the 7th international conference on Language Resources and Evaluation (LREC), Malta.* 41

Hartung, M. and A. Frank (2010b), "A Structured Vector Space Model for Hidden Attribute Meaning in Adjective-noun Phrases." In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10), Beijing, China.* 41

Hartung, M. and A. Frank (2011), "Exploring Supervised LDA Models for Assigning Attributes to Adjective-noun Phrases." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'11), Edinburgh, UK.* 41

He, S., K. Liu, Y. Zhang, L. Xu, J. Zhao, et al. (2014), "Question Answering over Linked Data Using First-order Logic." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar.* 40

Hearst, M. A. (1992), "Automatic Acquisition of Hyponyms from Large Text Corpora." In *Proceedings of the 14th Conference on Computational Linguistics (COLING'92), Nantes, France.* 40, 55

Hitzler, P., M. Krötzsch, S. Rudolph, and Y. Sure (2007), *Semantic Web: Grundlagen.* Springer-Verlag. 18

Hoeffner, K., S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A.-C. N. Ngomo (2016), "Survey on Challenges of Question Answering in the Semantic Web." *Semantic Web Journal.* 4, 39

Indurkhya, N. and F. J. Damerau (2010), *Handbook of Natural Language Processing*, 2nd edition. Chapman & Hall/CRC. vii, 39

Ittoo, A. and G. Bouma (2010), "On learning subtypes of the part-whole relation: do not mix your seeds." In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), Uppsala, Sweden.* 33

Kobilarov, G., C. Bizer, S. Auer, and J. Lehmann (2009), “DBpedia-A Linked Data Hub and Data Source for Web and Enterprise Applications.” *Programme Chairs.* 16

Krause, S., L. Hennig, A. Moro, D. Weissenborn, F. Xu, H. Uszkoreit, and R. Navigli (2016), “Sar-graphs: A language resource connecting linguistic knowledge with semantic relations from knowledge graphs.” *Web Semantics: Science, Services and Agents on the World Wide Web.* 35

Kun, A. L., A. Schmidt, A. Dey, and S. Boll (2013), “Automotive user interfaces and interactive applications in the car.” *Personal and Ubiquitous Computing*, 17. 2

Lafferty, J., A. McCallum, and F. Pereira (2001), “Conditional random fields: Probabilistic models for segmenting and labeling sequence data.” In *Proceedings of the eighteenth international conference on machine learning (ICML), Williamstown, Massachusetts, USA.* 35

Lanser, B., C. Unger, and P. Cimiano (2016), “Crowdsourcing Ontology Lexicons.” In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portoro, Slovenia.* 44

Lee, H., Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky (2011), “Stanford’s Multi-pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task.” In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task (CONLL Shared Task ’11), Portland, Oregon.* 37

Lehmann, J., R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, et al. (2015), “DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia.” *Semantic Web*, 6. 4

Lesk, M. (1986), “Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone.” In *Proceedings of the 5th annual international conference on Systems documentation (SIGDOC ’86), Toronto, Canada.* 40, 46

Levenshtein, V. I. (1966), “Binary codes capable of correcting deletions, insertions, and reversals.” In *Soviet physics doklady*, volume 10, 707–710. 104

Lin, D. and P. Pantel (2001), “DIRT @SBT @Discovery of Inference Rules from Text.” In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’01), San Francisco, California, USA.* 34

Lopez, V., M. Fernández, E. Motta, and N. Stieler (2012), “PowerAqua: Supporting users in querying and exploring the semantic web.” *Semantic Web*, 3. 2, 40

Lopez, V., V. Uren, M. Sabou, and E. Motta (2011), “Is question answering fit for the semantic web?: a survey.” *Semantic Web*, 2. 39

Mahendra, R., L. Wanzare, R. Bernardi, A. Lavelli, and B. Magnini (2011), “Acquiring relational patterns from wikipedia: A case study.” In *Proceedings of the 5th Language and Technology Conference (LTC 2011), Poznan, Poland.* 33, 86

Maillard, J. and S. Clark (2015), "Learning Adjective Meanings with a Tensor-Based Skip-Gram Model." *CoNLL 2015.* 41

Marimon, M. and N. Bel (2015), "Dependency structure annotation in the IULA Spanish LSP Treebank." *Language Resources and Evaluation*, 49. 137

Martin, J. H. and D. Jurafsky (2000), "Speech and language processing." *International Edition*, 710. 25

Mausam, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni (2012), "Open Language Learning for Information Extraction." In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '12), Jeju Island, Korea.* 35

Maynard, D. and S. Ananiadou (1999), "Term extraction using a similarity-based approach." *Recent advances in computational terminology*, 261–278. 38

Maynard, D. and S. Ananiadou (2000), "Identifying Terms by Their Family and Friends." In *Proceedings of the 18th Conference on Computational Linguistics (COLING '00), Stroudsburg, PA, USA.* 38

Maynard, D. and S. Ananiadou (2001), "TRUCKS: A Model for Automatic Multi-Word Term Recognition." 8. 38

McCrae, J., F. Quattri, C. Unger, and P. Cimiano (2014), "Modelling the Semantics of Adjectives in the Ontology-Lexicon Interface." In *Proceedings of the Cognitive Aspects of the Lexicon (CogAlex), workshop co-located with the 25th International Conference on Computational Linguistics (COLING 2014), Dublin, Ireland.* 94

McCrae, J., D. Spohr, and P. Cimiano (2011), "Linking lexical resources and ontologies on the semantic web with lemon." In *The Semantic Web: Research and Applications*, 245–259, Springer. 26

Mendes, P. N., M. Jakob, A. García-Silva, and C. Bizer (2011), "DBpedia Spotlight: Shedding Light on the Web of Documents." In *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics '11), Graz, Austria.* 36

Miller, G. A. (1995), "WordNet: A Lexical Database for English." *Communications of the ACM*, 38. 5, 97, 137

Monaghan, F., G. Bordea, K. Samp, and P. Buitelaar (2010), "Exploring your research: Sprinkling some saffron on semantic web dog food." In *Proceedings of the Semantic Web Challenge at 9th International Semantic Web Conference (ISWC2010), Shanghai, China.* 38

Moro, A., A. Raganato, and R. Navigli (2014), "Entity linking meets word sense disambiguation: a unified approach." *Transactions of the Association for Computational Linguistics*, 2. 36

Nadeau, D. and S. Sekine (2007), "A survey of named entity recognition and classification." *Lingvisticae Investigationes*, 30. 36

Nakashole, N., G. Weikum, and F. Suchanek (2012), "PATTY: a taxonomy of relational patterns with semantic types." In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Jeju Island, Korea.* 35

Navigli, R. (2009), "Word Sense Disambiguation: A Survey." *ACM Computing Surveys*, 41. 47

Navigli, R. and S. P. Ponzetto (2012), "BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network." *Artificial Intelligence*, 193. 125

Nebhi, K. (2012), "Ontology-Based Information Extraction from Twitter." Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data (COLING 2012), Bombay, India. 32

Nivre, J., J. Hall, and J. Nilsson (2006), "MaltParser: A Data-Driven Parser-Generator for Dependency Parsing." In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC), Genoa, Italy.* 28, 50, 137

Oliver González, A. and M. Vàzquez Garcia (2015), "TBXTools: A free, fast and flexible tool for automatic terminology extraction." 38

Padró, M., N. Bel, and A. Garí (2013), "Verb SCF extraction for Spanish with dependency parsing." *Procesamiento del lenguaje natural*, 51. 137

Page, L., S. Brin, R. Motwani, and T. Winograd (1999), "The PageRank citation ranking: bringing order to the web." 36

Pantel, P. and M. Pennacchiotti (2006), "Espresso: Leveraging generic patterns for automatically harvesting semantic relations." In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL).* 33

Park, Y., R. J. Byrd, and B. K. Boguraev (2002), "Automatic Glossary Extraction: Beyond Terminology Identification." In *Proceedings of the 19th International Conference on Computational Linguistics (COLING '02), Taipei, Taiwan.* 38

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011), "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, 12. 137

Peters, W., E. Montiel-Ponsoda, G. A. de Cea, and A. Gómez-Pérez (2007), "Localizing Ontologies in OWL." In *Actas OntoLex 2007.* 40

Preiss, J., T. Briscoe, and A. Korhonen (2007), "A System for Large-Scale Acquisition of Verbal, Nominal and Adjectival Subcategorization Frames from Corpora." In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), Prague, Czech Republic.* 40

Prévot, L., C.-R. Huang, N. Calzolari, A. Gangemi, A. Lenci, and A. Oltramari (2010), "Ontology and the lexicon: a multi-disciplinary perspective." In *Ontology and the Lexicon: A Natural Language Processing Perspective*, Cambridge University Press. 5

Pustejovsky, J. (1992), "The Acquisition of Lexical Semantic Knowledge from Large Corpora." In *Proceedings of the Workshop on Speech and Natural Language, workshop co-located with the Association for Computational Linguistics (ACL), Harriman, New York,USA.* 40

Raghunathan, K., H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning (2010), "A Multi-pass Sieve for Coreference Resolution." In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10), Cambridge, Massachusetts, USA.* 37

Ravichandran, D. and E. Hovy (2002), "Learning surface text patterns for a question answering system." In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL),Pennsylvania, USA.* 34, 40

Recasens, M., M. Can, and D. Jurafsky (2013), "Same Referent, Different Words: Unsupervised Mining of Opaque Coreferent Mentions." In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT), Atlanta, Georgia, USA.* 37

Ritter, A., S. Clark, Mausam, and O. Etzioni (2011), "Named Entity Recognition in Tweets: An Experimental Study." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11), Edinburgh, United Kingdom.* 36

Rocktäschel, T., M. Weidlich, and U. Leser (2012), "ChemSpot: a hybrid system for chemical named entity recognition." *Bioinformatics*, 28. 36

Saggion, H., A. Funk, D. Maynard, and K. Bontcheva (2007), "Ontology-Based Information Extraction for Business Intelligence." In *Proceedings of the 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea.* 32

Schalk, T. and L. Gibbons (2013), "Toward Natural User Interfaces in the Car: Combining Speech, Sound, Vision, and Touch." In *Adjunct Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 13), Eindhoven, The Netherlands.* 2

Schmid, H. (1994), "Probabilistic part-of speech tagging using decision trees." In *New methods in language processing.* 137

Schmid, H. (1995), "Improvements in part-of-speech tagging with an application to German." In *Proceedings of the ACL SIGDAT-Workshop.* 137

Sennrich, R., G. Schneider, M. Volk, and M. Warin (2009), "A new hybrid dependency parser for German." 137

Sennrich, R., M. Volk, and G. Schneider (2013), "Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis." In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2013), Hissar, Bulgaria.* 137

Shaalan, K. (2014), "A survey of arabic named entity recognition and classification." *Computational Linguistics*, 40. 36

Speck, R. and A.-C. N. Ngomo (2014), "Named Entity Recognition Using FOX." In *Proceedings of the 13th International Semantic Web Conference, Posters & Demonstrations Track (ISWC-PD'14), Riva del Garda, Italy.* 36

Stoyanov, V., C. Cardie, N. Gilbert, E. Riloff, D. Buttler, and D. Hysom (2010), "Coreference Resolution with Reconcile." In *Proceedings of the ACL 2010 Conference Short Papers (ACLShort '10), Uppsala, Sweden.* 38

Strötgen, J. and M. Gertz (2015), "A Baseline Temporal Tagger for all Languages." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015), Lisbon, Portugal.* 52, 125

Surdeanu, M., M. Ciaramita, and H. Zaragoza (2011), "Learning to Rank Answers to Non-Factoid Questions from Web Collections." *Computational Linguistics*, 37. 39

Sutton, C. and A. McCallum (2006), *An introduction to conditional random fields for relational learning*, volume 2. Introduction to statistical relational learning. MIT Press. 35

Sutton, C. and A. McCallum (2010), "An introduction to conditional random fields." *arXiv preprint arXiv:1011.4088.* 35

Tristram, F., S. Walter, P. Cimiano, and C. Unger (2015), "Weasel: a machine learning based approach to entity linking combining different features." In *Proceedings of 3th International Workshop on NLP and DBpedia, co-located with the 14th International Semantic Web Conference (ISWC 2015), October 11-15, USA.* 36

Unger, C., L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano (2012), "Template-based Question Answering over RDF Data." In *Proceedings of the 21st International Conference on World Wide Web (WWW '12), Lyon, France.* 2, 40

Unger, C., C. Forascu, V. Lopez, A.-C. Ngonga Ngomo, E. Cabrio, P. Cimiano, and S. Walter (2014a), "Question Answering over Linked Data (QALD-4)." In *Working Notes for CLEF 2014 Conference.* 39, 94

Unger, C., A. Freitas, and P. Cimiano (2014b), "An Introduction to Question Answering over Linked Data." In *Proceedings of Reasoning on the Web in the Big Data Era: 10th International Summer School 2014, Athens, Greece.* 39

Unger, C., J. McCrae, S. Walter, S. Winter, and P. Cimiano (2013), "A lemon lexicon for DBpedia." In *Proceedings of the 1st International Workshop on NLP and DBpedia, co-located with the 12th International Semantic Web Conference (ISWC 2013), October 21-25, Sydney, Australia.* 59

Usbeck, R., A.-C. N. Ngomo, L. Bühmann, and C. Unger (2015), "HAWK – Hybrid Question Answering Using Linked Data." In *Proceedings of12th European Semantic Web Conference (ESWC 2015) , Portoroz, Slovenia.* 40

Usbeck, R., A.-C. Ngonga Ngomo, M. Röder, D. Gerber, S. A. Coelho, S. Auer, and A. Both (2014), *AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data.* 36

Versley, Y., A. Moschitti, M. Poesio, and X. Yang (2008a), "Coreference Systems Based on Kernels Methods." In *Proceedings of the 22Nd International Conference on Computational Linguistics (COLING '08), Manchester, United Kingdom.* 37

Versley, Y., S. P. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang, and A. Moschitti (2008b), "BART: A Modular Toolkit for Coreference Resolution." In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session (HLT-Demonstrations '08), Columbus, Ohio, USA.* 37

Vila, M., H. Rodríguez, and M. A. Martí (2010), "WRPA: A system for relational paraphrase acquisition from Wikipedia." *Procesamiento del lenguaje natural*, 45. 34, 95

Vila, M., H. Rodríguez, and M. A. Martí (2013), "Relational paraphrase acquisition from Wikipedia: The WRPA method and corpus." 21. 34

Walter, S., C. Unger, and P. Cimiano (2014a), "ATOLL - A framework for the automatic induction of ontology lexica." *Data & Knowledge Engineering*, 94. 65

Walter, S., C. Unger, and P. Cimiano (2015), "DBlexipedia: A nucleus for a multilingual lexical Semantic Web." In *Proceedings of 3th International Workshop on NLP and DBpedia, co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pennsylvania, USA.* 113

Walter, S., C. Unger, P. Cimiano, and D. Bär (2012), "Evaluation of a Layered Approach to Question Answering over Linked Data." In *Proceedings of the 11th International Conference on The Semantic Web (ISWC'12), Boston, MA, USA.* 2, 40

Walter, S., C. Unger, P. Cimiano, and B. Lanser (2014b), "Automatic acquisition of adjective lexicalizations of restriction classes." In *Proceedings of the 2nd International Workshop on NLP and DBpedia, co-located with the 13th International Semantic Web Conference (ISWC 2014), October 19-23, Riva del Garda, Italy.* 98, 105

Wimalasuriya, D. C. and D. Dou (2010), "Ontology-based information extraction: An introduction and a survey of current approaches." *Journal of Information Science.* 32

Wong, W., W. Liu, and M. Bennamoun (2007), "Determining Termhood for Learning Domain Ontologies Using Domain Prevalence and Tendency." In *Proceedings of the Sixth Australasian Conference on Data Mining and Analytics (AusDM '07), Gold Coast, Australia.* 38

Wu, F. and D. S. Weld (2010), "Open Information Extraction Using Wikipedia." In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10), Uppsala, Sweden.* 35

Zettlemoyer, L. S. and M. Collins (2005), "Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars." In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI '05), Edinburgh, Scotland.* 28