LODifier

Generating RDF from Natural Language

Bachelor's Thesis

of

Isabelle Augenstein

Department of Computational Linguistics
Heidelberg University, Germany

Supervisors:

Prof. Dr. Sebastian Padó
Department of Computational Linguistics
Heidelberg University, Germany

Dr. Sebastian Rudolph
Institute of Applied Informatics and Formal Description Methods
Karlsruhe Institute of Technology, Germany

Submission Date: 21 July 2011

Author's declaration

I declare that this thesis has been completed by myself independently without outside help and that only the defined sources and study aids were used. Any thoughts or quotations which were inferred from the work of others are made known through the definition of sources. This thesis has not been submitted, either in part or whole, for a degree at this or any other university or institution.

Heidelberg, 21 July 2011

(Isabelle Augenstein)

Abstract

The automatic extraction of information from text and formal description of this information is an important goal of both the Semantic Web and computational linguistics. Extracted information can be used for a variety of tasks such as ontology generation, question answering or information retrieval.

LODifier is an approach that combines deep semantic analysis with named entity recognition, word-sense disambiguation and controlled Semantic Web vocabularies in order to extract named entities and relations between them from text and to convert them to RDF. The approach is evaluated as an information retrieval task and the evaluation is performed with a manually created corpus. Very promising results could be achieved: a coverage of 72.2% for correctly recognized relations.

Zusammenfassung

Das automatische Extrahieren von Informationen aus Text und die formale Beschreibung dieser Informationen ist sowohl im Semantic Web als auch in Computerlinguistik ein wichtiges Ziel. Die extrahierten Informationen können für viele verschiedene Aufgaben verwendet werden, z.B. für Ontology-Learning, Question Answering oder Information Retrieval.

LODifier ist ein Ansatz, der tiefe semantische Analyse mit Named Entity Recognition, Word Sense Disambiguation und kontrollierten Semantic Web Vokabularen kombiniert um Named Entities und Relationen zwischen ihnen aus Text zu extrahieren und in RDF zu konvertieren. Der Ansatz wird als Information Retrieval Task evaluiert und die Evaluation mit einem manuell erstellten Korpus durchgeführt. Vielversprechende Ergebnisse konnten erzielt werden: eine Coverage von 72,2% für korrekt erkannte Relationen.

Acknowledgements

First, I would like to thank Dr. Sebastian Rudolph who made this Bachelor's thesis possible. He created a stimulating research environment at AIFB that laid the groundwork for this thesis and supported me in moments of doubt. Many thanks go to Prof. Dr. Sebastian Padó for supervising the thesis. He encouraged a cooperation with the AIFB and helped to develop ideas put forward. My thesis would have been impossible without the help of Christoph Mayer and Danny Rehl, who put a lot of effort into annotating the evaluation data.

Contents

Chapter 1.

Introduction

In this Bachelor's thesis, I will describe an approach to convert natural language to RDF triples. The approach combines named *entity recognition* (NER) methods with parsing, semantic analysis and *word sense disambiguation* (WSD) techniques. Several existing tools and resources are used for this task: *Wikifier* for NER, *C&C* and *Boxer* for parsing and semantic analysis, ukb for WSD and *DBPedia* and *WordNet* to provide a controlled vocabulary. The goal of this thesis is to develop a method for automatically converting a text to a semantic structure that is better suited for knowledge extraction than plain, unstructured text. The resulting structure is encoded in RDF and is interlinked with the large knowledge bases DBPedia and WordNet.

The thesis is organized as follows: in chapter 2, background information on the Semantic Web, RDF, ontologies, DBPedia and WordNet is provided. Chapter 3 presents an overview over relevant related work and concludes with a motivation for this thesis. A detailed description of the methodology behind LODifier is described in section 4 and the system's evaluation and results are presented in section 5. In chapter 6 an outlook on possible future work is given. Attached to this document is a CD, which contains the LODifier source code and the evaluation data. The format of the content on CD is described in the appendix.

Chapter 2.

Background

In this section, the basic concept of the *Semantic Web* and Semantic Web technologies (see also Hitzler et al. (2009) for a more detailed introduction to the topic) is explained. One Semantic Web standard, RDF, is explained in detail, since it is crucial for understanding how the output of LODifier is created. The ontology concept and the idea behind the *Linked Open Data* (LOD) initiative are explained. Furthermore, the knowledge bases WordNet and DBPedia are introduced.

2.1. Semantic Web

The Semantic Web can be understood as an extension of the *World Wide Web* (WWW). Its concept was introduced by Tim Berners-Lee (Berners-Lee et al., 2001), who is also the inventor of the WWW. The Semantic Web organizes data in a semantic network.
The WWW consists of interlinked web pages, which were originally intended for human readers. Therefore, they usually do not contain a semantic structure to aid machine processing. The goal of the Semantic Web is to make data widely accessible by establishing a network of links between data and to introduce standards to describe the data so that it can be processed automatically.

The Semantic Web standards are developed by the *World Wide Web Consortium* (W3C). There are standards for describing data, e.g., RDF or OWL and standards for querying data, e.g., SPARQL.

2.2. RDF

The *Resource Description Framework* (RDF) is a Semantic Web standard for modeling data. It was the first Semantic Web language developed by the W3C.

The data model of RDF is a directed graph consisting of nodes that are linked by edges. In RDF syntax, the graph is separated into pairs of nodes that are linked by edges. The resulting *RDF triples* consist of the three parts *subject*, *predicate* and *object*. These names are, however, not to be mistaken for grammatical categories, even though they sometimes overlap.

URIs

For describing RDF data, uniform identifiers are needed. The purpose of uniform identifiers is that on one hand, if two documents contain information about the same resource, this resource also has the same identifier. On the other hand, identical labels could refer to different resources.

To solve this problem, *Uniform Resource Identifiers* (URIs) are introduced to discriminate resources. URIs can either be *Uniform Resources Locators* (URLs), *Uniform Resources Names* (URNs) or both. A URL is used, if the URI has a web address and can be accessed online. In some cases, a resource has no online location, but nevertheless has a name and can be identified. A URI has a clearly defined syntax, regardless of whether it is a URL or a URI. It consists of four parts as illustrated below:

```
scheme:[//authority]path[?query][#fragment]
```

A URI starts with a scheme name. This could be e.g. a protocol like *HTTP* or *FTP*. The scheme is followed by a path. The path optionally starts with an authority, which is usually a hostname. The query is optional and starts with a question mark. The fragment is also optional and is preceeded by a hashmark. It is used if a subpart of the resource shall be addressed.

As already explained URIs serve the purpose of having an identifier that is the same for all occurrences of the same resource. In practice, it is still possible to have different URIs for the same resource. A solution for this is to use certain *vocabularies*. An example for a well-defined vocabulary is DBPedia (see chapter 2.5). If a vocabulary is not available, it is possible to introduce new URIs. To introduce a new URI, it must first be made sure, that the URI is not already used in a different context.

However, if one has the choice between a self-defined and an existing URI, one should always use the existing one. When existing URIs are URLs, they have an existing webpage that contains more information about the resource, e.g., a documentation or references to other related URIs, which is an advantage over self-defined URIs.

Notations

There are different options for presenting RDF content, also called *notations*. The most commonly used notations include *RDF/XML* and *N3*. LODifier uses the N3 notation, because of the simplicity of the syntax. However, N3 can be automatically converted into RDF/XML and vice versa.

N3

N3 is a realization of RDF that is closely oriented on the data model of graphs. It is basically a collection of every triple of the graph. The N3 Notation comprises the less complicated derivations *N-Triples* and *turtle*. A very basic sample of the usage of N3 is given in the example below.

```
<http://dbpedia.org/page/Google>
   <http://example.org/buy> <http://dbpedia.org/page/YouTube> .
```

N3 also provides an opportunity to abbreviate URIs. This is done using *namespaces*. The above sample can therefore be rearranged as follows:

```
@prefix dbpedia: <http://dbpedia.org/page/> .
@prefix ex: <http://example.org/> .
dbpedia:Google ex:buy dbpedia:YouTube .
```

RDF/XML

Although N3 is very easy to read for humans and also machine-processable, it is not the most commonly used RDF syntax. The most commonly used RDF syntax is RDF/XML, an XML realization of RDF. This may be due to the large distribution of and its compatibility with other applications. The main difference between N3 and RDF/XML is that in RDF/XML, triples are organized as hierarchical structures that are grouped by subjects. A very basic example of the usage of RDF/XML is again given in the sample below.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:ex="http://example.org/"
  xmlns:dbpedia="http://dbpedia.org/page/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <rdf:Description rdf:about="http://dbpedia.org/page/Google">
             <ex:buy rdf:resource="http://dbpedia.org/page/YouTube"/>
        </rdf:Description>
</rdf:RDF>
```

Literals

Not every RDF content has to be modeled using URIs. One opportunity for modeling concrete values are *literals*. Numbers, dates or times can be modeled this way. A literal in RDF is represented as a string, i.e., a sequence of characters. The literal *32* would e.g., refer to the number 32. It should be noted that this kind of literals are called *untyped literals*. That means the literal *32* is not by default recognized as a number. In order to interpret the literal *32* as a number, data types have to be used.

Data types

RDF allows certain data types for literals, which are represented by URIs. Data type information is added to the RDF triples using XML Schema.

In LODifier, three kinds of XML Schema data types are used: *decimal*, *date* and *time*. Below is an example in N3 notation how to integrate these data types in RDF.

```
@prefix ex: <http://example.org/> .
@prefix xmls: <http://www.w3.org/2001/XMLSchema#> .
ex:decimal xmls:decimal "32" .
ex:date xmls:date "2011-01-07" .
ex:time xmls:time "00:08:17" .
```

Blank Nodes

RDF does not only allow modeling of simple binary relations, but also of more complex data structures. For describing complex data structures, anonymous resources called *blank nodes* may be required. Since blank nodes do not contain any information themselves, they are not modeled with URIs. Instead, node IDs are used. The notation of blank nodes in RDF is as follows:

```
@prefix xmls: <http://www.w3.org/2001/XMLSchema#> .
_:x0 xmls:decimal "32" .
```

The notation of a blank node starts with an underscore and a colon and is followed by an arbitrary ID that may consist of characters and digits.

Resource Identification

As already mentioned RDF triples consist of a subject, a predicate and an object. To describe these parts of the triple, several structures are possible: URIs, literals and blank nodes. An RDF subject can be either a URI or a blank node, an RDF predicate must be a URI and an RDF object can be a URI, a blank node or a literal.

RDF Vocabulary

RDF has a defined vocabulary to model knowledge. The concepts used by LODifier are described here.

rdf:type

New class names are introduced with *rdf:type*. This concept declares a resource as an instance of a class. Here is an example of how to use rdf:type :

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/> .
@prefix class: <http://example.org/class> .
_:var0x0 rdf:type class:buy .
```

Reification

Reification is used to describe complex data structures. A blank node is introduced which serves as a referent for the whole complex statement. Each part of the triple that is an object of the blank node can be described using *rdf:subject*, *rdf:predicate* and *rdf:object*. Below is an example that shows a reified triple.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/> .
ex:Anne ex:propose _:var0x0 .
_:var0x0 rdf:subject ex:Peter .
_:var0x0 rdf:predicate ex:buy .
_:var0x0 rdf:object ex:lettuce .
```

2.3. Linked Open Data

The goal of the LOD initiative is somewhat similar to the goal of Semantic Web in general: Interlinking structured data. All of the data to be interlinked has to fulfill certain requirements. Resources have to be described using HTTP URIs and information has to be provided in a Semantic Web standard. If related URIs are available, they should be interlinked.

Currently, there are 203 published data sets, which consist of more than 25 billion RDF triples. Data sets include e.g., DBPedia, Friend of a Friend, Geonames and WordNet.

2.4. Ontologies

An ontology in computer science is a standardized representation of knowledge. It is used to describe concepts and relation types. The first definition of ontologies was given by Gruber (1993). He described an ontology as a "formal, explicit specification of a shared conceptualisation".

Components

Gruber (1993) states that "ontologies are often equated with taxonomic hierarchies of classes, class definitions, and the subsumption relation, but ontologies need not be limited to these forms. Ontologies are also not limited to conservative definitions that is, definitions in the traditional logic sense that only introduce terminology and do not add any knowledge about the world."

As Gruber (1993) clarifies, ontologies can consist of two components - terminological knowledge (*TBox* statements) about the concepts of a domain and assertion knowledge (*ABox* statements), knowledge about entities of these concepts. An ontology is composed of either ABox or TBox statements or of both. Ontologies are often modeled using Semantic Web technologies.

2.5. DBPedia

DBPedia (Bizer et al., 2009) is a large, freely available domain-independent multilingual ontology that is extracted from Wikipedia. It currently contains information about 3.5 million entities and consists of 672 RDF triples.

The knowledge DBPedia consists of is extracted from Wikipedia articles. That knowledge comprises Wikipedia page names, infobox templates, categorization information, images, geo-coordinates and links to external webpages. Each entity in DBPedia is described by a URI of the form *http://dbpedia.org/resource/Name*, where *Name* is the name of the Wikipedia page. DBPedia contains links to various data sets including FOAF, Geonames and WordNet.

2.6. WordNet

WordNet is a large scale lexical database for the English language. Version 3.0 of WordNet contains more than 155000 words of the word types nouns, verbs, adjectives and adverbs.

Words are grouped into sets of synonyms, which are called *synsets*. Each word is disambiguated resulting in different *word senses*. Therefore, a word can appear in several synsets. The synsets themselves are linked to other synsets by *conceptual relations*. Synsets contain *glosses* (short definitions) and short example sentences.

The most common relation in WordNet is *hyponymy*, also called *ISA* relation. Thus, synsets have links to more general synsets. Other relation types in WordNet include *meronymy* (part-whole relation), *troponymy* (specific manner characterizing an event described by verbs) and *antonymy* (direct opposites).

WordNet is a very valuable resource for NLP due to the large coverage of the English language. Since there are RDF versions of WordNet available, RDF WordNet 2.0 (van Assem et al., 2006) and RDF WordNet 3.0 (van Assem and van Ossenbruggen, 2011), WordNet can also be used as a vocabulary for the Semantic Web. WordNet is part of the Linked Open Data cloud and is interlinked to several data sets including DBPedia.

Chapter 3.

Related Work

This section presents approaches that are in some ways similar to the methods used by LODifier. After the approaches are discussed, the methodology LODifier uses is motivated in section 3.2

3.1. Existing Systems

Ontology Generation

An overview over ontology generation systems is given by Bedini and Nguyen (2007). Although the article does not consider systems developed after 2007, to my knowledge its conclusion would still be similar. The ontology generation process can be divided into five steps. The first step is the *extraction* of information from a structured, semi-structured or unstructured corpus. Techniques used for extraction include NLP techniques, clustering and machine learning. The extraction is followed by an *analysis*, where the retrieved information is clustered and relations are extracted. In the *generation* process, the model is converted into a formal specification like RDF or OWL. After this the ontology is *validated* to remove wrong concepts and relationships, which is usually done manually. The final step is the *evolution* of the ontology, i.e., to adapt the ontology to changes.

Bedini and Nguyen (2007) compare several ontology systems on their performance and on their automation of these five steps. The conclusion of the article is, however, that there is no system that fully automates the process of evolution which means, the ontologies generated are static. Also, none of the systems described fully automates the process of extraction, analysis and generation and operate on unstructured data.

Relationship Extraction

There are various approaches to extracting relationships from text. These approaches usually include the annotation of text with named entities and relations and the extraction of those relations. Two approaches that are very similar to LODifier are of Byrne and Klein (2009) and Ramakrishnan et al. (2006). They both use NER, POS-tagging and parsing to discover named entities and relations between them. The resulting relations are converted to RDF.

The disadvantage of these methods is however that they use labeled data as a base for extracting relations, which is not flexible, as labeled data requires manual annotation. Another interesting approach is to use DBPedia for relationship discovery (Heim et al., 2010). The *RelFinder* maps selected named entities to DBPedia and then searches the data set for relations. The results are presented to the user as graph visualization. The goal of this approach is to provide more detailed information about relations and then it is possible with other sources (Google and Wikipedia are used for comparison in a user study) that are used for relationship discovery. The system performs rather well and outperforms Google and Wikipedia in user satisfaction and efficiency.

The *RelFinder* provides a very interesting user application for relationship extraction. It depends, however, on existing, rather static datasets and could thus not be used, e.g., to extract relations from news that are updated daily.

Extracting Semantic Networks

Harrington and Clark (2007) show a successful approach to automatically creating a semantic network. The *AsKNet* system uses C&C and Boxer to extract semantic relations. To decide, which nodes refer to the same entities, similarity scores are computed based on spreading activation and nodes are then mapped together. An approach building on AsKNet comes from Wojtinnek et al. (2010). They use AsKNet to build a semantic network based on relations between concepts instead of relations between named entities as already present in AsKNet. The resulting graph is then converted to RDF.

AsKNet and also the system building on AsKNet by Wojtinnek et al. (2010) present an approach very similar to LODifier. However, AsKNet and LODifier differ in the way they disambiguate named entities. LODifier uses NER and WSD methods before generating RDF triples and describes the entities and relations using DBPedia and WordNet URIs. AsKNet first generates semantic structures from text and then tries to map nodes and edges together based on similarity.

The approach by Wojtinnek et al. (2010) uses RDF to describe a semantic network based on AsKNet that contains relations between concepts. The resulting RDF triples are not interlinked with other data bases, which is done by LODifier. Unfortunately, there is no RDF version of AsKNet itself, which is a further difference between AsKNet and LODifier.

3.2. Conclusion

In summary, it can be said that there are four main differences between the approaches described and the approach LODifier takes. These are the aspect of automating the extraction process, using unlabeled data for extracting relations, performing a full extraction of all relations in the text and describing those using Semantic Web standards and linking the named entities and relations to well-defined vocabularies.

To achieve these goals, several well-established methods used in these approaches are combined: Subjects and objects are recognized using NER; relations are extracted by parsing the text and performing a semantic analysis with Boxer; finally, the resulting RDF triples are linked to well-defined vocabularies.

Chapter 4.

System

This section describes the resources and algorithms used to build the ontology generation system LODifier. First, the initial approach developed before the Bachelor's thesis is described in section 4.1. Section 4.1.5 gives a summary of the initial approach and explains its shortcomings. The improved system developed during the Bachelor's thesis is then explained in section 4.2.

4.1. Initial System

The goal of the LODifier system is to generate an RDF representation out of English plain text. First, the input is searched for subjects, objects and relations between them. Then, RDF triples are generated and DBPedia URIs are linked to the subjects and objects.

Subjects and objects are recognized using the NER system Wikifier (Milne and Witten, 2011). Relations between these two are then recognized using the statistical parser C&C and Boxer (Curran et al., 2007), a tool for generating discourse representation structures (Kamp and Reyle, 1993).

To get RDF triples, the Boxer DRS output is processed and triples are extracted. The named entities from Wikifier are converted into DBPedia URIs and linked to the RDF triples.

4.1.1. Recognizing subjects and objects

The first step is to identify subjects and objects. They are recognized using the NER system Wikifier (Milne and Witten, 2011) that enriches English plain text with Wikipedia links. Wikifier is used for NER, because named entities are returned as Wikipedia URLs which can be easily converted to DBPedia URIs.

This way, the DBPedia namespace can be used to describe named entities and no new URIs have to be introduced.

To disambiguate the Wikipedia links, Wikifier employs a machine learning approach that uses the links between Wikipedia articles as training data. Since the links between Wikipedia articles are manually created by Wikipedia editors, the training data consists of man-made disambiguation choices and is thus very reliable. Wikifier performs well at a precision of 73.8% and recall of 74.4%.

If Wikifier finds a named entity, it is substituted by the name of the English Wikipedia page. Figure 4.1 shows an example of the Wikifier output for the test sentence *Google buys YouTube*.

```
[[Google]] buys [[YouTube]].
```

Figure 4.1. Wikifier output for test sentence *Google buys YouTube.*

4.1.2. Recognizing relations

After subjects and objects are identified using Wikifier, the relations between those two are identified. This is done by using *C&C* and the *Boxer* system developed by Curran et al. (2007). The C&C parser uses a maximum entropy tagger that tags words with the grammatical categories from the Penn Treebank. In addition, C&C contains a named entity recognizer that distinguishes between ten different named entity types as displayed in table 4.1.

The parser has precision and recall scores well above 80%. Figure 4.2 shows an example for the C&C output for the test sentence *Google buys YouTube.*.

Boxer is a system that uses the output of the statistical parser *C&C*. Boxer produces DRSs, which consist of discourse referents and conditions. For every new noun phrase or event in a sentence, a discourse referent is introduced and for every new relation, a condition is introduced. The conditions are described as either one- or two-place relations. Since DRSs are recursive, it is also possible that a new DRS is introduced. Boxer then binds anaphoras to previously introduced discourse referents or accommodates them. Since the structure of DRSs is very similar to the structure of RDF triples, they can be used as a preprocessing step for extracting RDF triples from text.

Boxer has several output formats. The default one is PROLOG, where every discourse referent is represented by a Prolog variable. A flat structure (where

Boxer Syntax	Explanation
org	organization
per	person
ttl	title
quo	quotation
loc	location
fst	first name
sur	surname
url	URL
ema	E-Mail
nam	unknown name

Table 4.1. Named Entity Types recognized by C&C

the recursive structure is unfolded), an XML structure or first-order-logic are also possible. Finally, a graphical display of the discourse representation structures, as introduced by Kamp and Reyle (1993), is possible. In figure 4.3, the example sentence *Google buys YouTube.* is displayed in the Prolog and the graphical output format.

DRS conditions in Boxer can be either basic or complex (see table 4.2). There can be one-place predicates, which are introduced by nouns, verbs, adverbs and adjectives. These are e.g., person, event or topic. There are also two-place relations which are introduced by prepositions and verb roles, e.g. agent, patient or theme. A full list of the relations from Curran et al. (2011) can be found in the appendix in tables A.1 and A.2. More details of the Boxer output syntax are also described by Curran et al. (2011).

4.1.3. Generating RDF triples

After the Boxer DRS output is processed, RDF triples are extracted in the next step. The first thing to be considered is what namespace to use for the URIs. Unlike Wikifier, Boxer doesn't provide URLs which can be transformed into URIs, so new URIs must be introduced.

What needs to be considered for defining new URIs is that Boxer has a set of fixed types, predicate and relation types. Tables A.1 and A.2 contain an overview over the different fixed types in Boxer. So to create new URIs there

```
ccg(1,
 rp(s:dcl,
  ba(s:dcl,
   lx(np, n,
    t(n, 'Google', 'Google', 'NNP', 'I-NP', 'I-ORG')),
   fa(s:dcl\np,
    t((s:dcl\np)/np, 'buys', 'buy', 'VBZ', 'I-VP', 'O'),
    lx(np, n,
     t(n, 'YouTube', 'YouTube', 'NNP', 'I-NP', 'I-ORG')))),
  t(period, '.', '.', '.', 'O', 'O'))).
```

Figure 4.2. C&C output for test sentence *Google buys YouTube.*

Basic Condition		Complex Condition	
Boxer Syntax	Explanation	Boxer Syntax	Explanation
pred	one-place predicates	or	disjunction
rel	two-place relations	imp	implication
named	named entities	not	negation
timex	time expressions	nec	necessarily
card	cardinal expressions	pos	possibly
eq	equality	whq	question
		prop	propositional attitude

Table 4.2. Basic and Complex Boxer Conditions

has to be distinguished between fixed predicate and relation types and not fixed predicate and relation types. The namespaces used are explained in table 4.3.

The next step is to extract RDF triples from the different Boxer relation types. Since DRSs are recursive and often contain encapsulated DRSs, it is crucial to unfold the DRSs recursively. This means, that the extraction process begins with the DRS on the highest level and ends with the DRS on the lowest level.

Each DRSs contains a set of conditions, which are either one- or two-place relations. For each relation type, the extraction process is different. Below is a description of the extraction process for every single relation type.

Prefix	Namespace	Usage
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	assigning types
owl	`http://www.w3.org/2002/07/owl#`	Equality
xmls	`http://www.w3.org/2001/XMLSchema#`	Time & cardinal expressions
foaf	`http://xmlns.com/foaf/0.1/`	Boxer NEs
geo	`http://www.w3.org/2003/01/geo/wgs84_pos/`	Boxer NEs
dbpedia	`http://dbpedia.org/resource/`	DBPedia URIs
wn30	`http://purl.org/vocabularies/princeton/wn30/`	WordNet URIs
ex	`http://example.org/`	Boxer complex conditions
ne	`http://example.org/ne`	Boxer NEs
drsclass	`http://example.org/drsclass/`	fixed Boxer predicates
class	`http://example.org/class/`	not fixed predicates
drsrel	`http://example.org/drsrel/`	fixed Boxer relations
rel	`http://example.org/rel/`	not fixed relations

Table 4.3. Namespaces used by LODifier

```
sem(1,
    [word(1001,'Google'),word(1002,buys),word(1003,'YouTube'),
    word(1004,'.')], [pos(1001,'NNP'),pos(1002,'VBZ'),pos(1003,'NNP'),
    pos(1004,'.')], [ne(1001,'I-ORG'),ne(1003,'I-ORG')],
    smerge(drs([[1001]:x0,[1003]:x1],[[1001]:named(x0,google,org,0),
    [1003]:named(x1,youtube,org,0)]),drs([[1002]:x2],
    [[]:pred(x2,event,n,1), [1002]:pred(x2,buy,v,0),
    [1002]:rel(x2,x0,agent,0),[1002]:rel(x2,x1,patient,0)])) ).

%%%   ______________________   _______________
%%%  | x0 x1                | | x2            |
%%%  |......................| |...............|
%%% (| named(x0,google,org)  |+| event(x2)     |)
%%%  | named(x1,youtube,org) | | buy(x2)       |
%%%  |______________________| | agent(x2,x0)  |
%%%                            | patient(x2,x1) |
%%%                            |_______________|
```

Figure 4.3. Boxer output for test sentence *Google buys YouTube.*

Predicates

```
_:var0 rdf:type (class / drsclass):? .
```

A blank node is introduced and assigned to the URI of the predicate marked as *?*. The predicate can either be a fixed Boxer predicate, in which case the *drsclass* prefix is used or it is not fixed, in which case the *class* prefix is used.

Relations

```
_:var0 (rel / drsrel):? _:var1 .
```

A blank node is introduced and assigned to the URI of the relation marked as *?'*. The relation can either be a fixed Boxer relation, in which case the *drsrel* prefix is used or it is not fixed, in which case the *rel* prefix is used.

Named Entities

```
_:var0 drsclass:named ne:? .
```

```
ne:? rdf:type (foaf:Organization / foaf:Person / foaf:title / foaf:firstName
/ foaf:surname / foaf:name / geo:Point / foaf:homepage / foaf:mbox) .
```

A blank node is introduced and assigned to the URI of the named entity marked as *?*. The named entity is then assigned to a URI of the *foaf* or *geo* namespace.

Time Expressions

```
_:var0 (xmls:date / xmls:time) ? .
```

A blank node is introduced and assigned as either date or time to a time expression marked as *?*.

Cardinal Expressions

```
_:var0 xmls:decimal ? .
```

A blank node is introduced and assigned as a number to a cardinal expression marked as *?*.

Equality Expressions

```
_:var0 xmls:number ? .
```

One blank node is assigned to another blank node.

Disjunction

```
_:var0 rdf:type ex:disjunction .
_:var0 ex:disjunct _:var00 .
_:var0 ex:disjunct _:var01 .
_:var00 rdf:type ex:conjunction .
_:var01 rdf:type ex:conjunction .
```

A blank node is introduced and declared as disjunction. Two new blank nodes are introduced which are conjunct from the first blank node. The two blank nodes are defined as conjunctions.

Implication

```
_:var0 rdf:type ex:implication .
_:var0 ex:antecedent _:var00 .
_:var0 ex:consequent _:var01 .
_:var00 rdf:type ex:conjunction .
_:var01 rdf:type ex:conjunction .
```

A blank node is introduced and declared as implication. Two new blank nodes are introduced; one is the antecedent and the other one the consequent of the first blank node. The two blank nodes are declared as conjunctions.

Negation

```
_:var0 rdf:type ex:negation .
_:var0 rdf:type ex:conjunction .
```

A blank node is introduced and declared as negation. The blank nodes are declared as conjunction.

Necessarily

```
_:var0 rdf:type ex:necessity .
_:var0 rdf:type ex:conjunction .
```

A blank node is introduced and declared as a necessity. The blank nodes are declared as conjunction.

Possibly

```
_:var0 rdf:type ex:possibility .
_:var0 rdf:type ex:conjunction .
```

A blank node is introduced and declared as a possibility. The blank nodes are declared as conjunction.

Wh-question

```
_:var0 rdf:type ex:referent .
_:var0 ex:answer _:var00 .
_:var0 ex:question _:var01 .
_:var00 rdf:type ex:conjunction .
_:var01 rdf:type ex:conjunction .
```

A blank node is introduced and declared as a wh-question. Two new blank nodes are introduced; one is the answer and the other the question of the first blank node. The two blank nodes are declared as conjunctions.

Propositional attitude

```
_:var0 rdf:type ex:proposition .
_:var0 rdf:type ex:conjunction .
```

A blank node is introduced and declared as a proposition. The blank nodes are declared as conjunction.

4.1.4. Linking DBPedia URIs to Boxer classes

After subjects and objects are identified, relations are identified and RDF triples are extracted, the last step is to generate DBPedia URIs out of the Wikifier output and link those DBPedia URIs to previously introduced Boxer classes.

DBPedia is a large ontology which has been created by extracting various parts of Wikipedia pages. Therefore, every Wikipedia page itself has a corresponding DBPedia page. Also, information on the Wikipedia pages such as infobox templates, categorization information, images, geo-coordinates and links to external web pages are extracted (Bizer et al., 2009).

Every Wikipedia page with the URL *http://en.wikipedia.org/wiki/Name* has a corresponding DBPedia page with the URI *http://dbpedia.org/page/Name*. This allows for an easy conversion of Wikipedia URLs to DBPedia URIs. To create DBPedia URIs from the Wikifier output, the English Wikipedia page name is simply appended to the DBPedia prefix.

After this is done, the DBPedia URIs are linked to the corresponding Boxer classes. More precisely, they are linked to the blank nodes that are assigned to Boxer classes.

Figure 4.4 shows an example of the LODifier output for the test sentence *Google buys YouTube.*

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix drsclass: <http://example.org/drsclass/> .
@prefix drsrel: <http://example.org/drsrel/> .
@prefix class: <http://example.org/class/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbpedia: <http://dbpedia.org/page/> .
_:var0x0 drsclass:named dbpedia:Google .
dbpedia:Google rdf:type foaf:Organization .
_:var0x1 drsclass:named dbpedia:YouTube .
dbpedia:YouTube rdf:type foaf:Organization .
_:var0x2 rdf:type drsclass:event .
_:var0x2 rdf:type class:buy .
_:var0x0 drsrel:agent :var0x2 .
_:var0x1 drsrel:patient :var0x2 .
```

Figure 4.4. LODifier output for test sentence “Google buys YouTube”.

4.1.5. Conclusion

In the initial approach to automatically extract RDF triples out of text, the triples were created by converting the output of the C&C Parser and Boxer to RDF format. Named entities were recognized using Wikifier and DBPedia links were linked to the named entities.

On frequent improvised tests, the system produced an accurate output. However, the approach still has two shortcomings: first, it lacks a statistic evaluation of the system's performance; i.e., it is unclear how successful the approach has been. Second, existing URIs are linked to subjects and objects, but for relations, no existing URIs are available new ones are introduced.

My Bachelor's thesis tackles both these issues. The original LODifier system is improved, as existing URIs are searched and linked to relations. Afterwards, an evaluation is performed to measure the coverage of the modified LODifier system.

4.2. Improved System

In order to improve LODifier, existing URIs are searched and when found, linked to relations. An important decision at this point is from what vocabulary URIs are taken. The criteria for choosing a vocabulary are that is contains URIs for relations, that it is possible to determine which URI to assign to which relation and that the vocabulary is published as linked open data (see also section 2.3 on linked open data) and thus interlinked to other vocabularies.

Two different approaches are pursued: The interlinking of Boxer relations with DBPedia properties (section 4.2.1) and the interlinking of Boxer relations with of RDF WordNet class types (section 4.2.2). Finally, RDF triples are created and the new URIs are interlinked (section 4.2.3).

4.2.1. Assigning DBPedia URIs to Boxer relations

DBPedia contains a data set consisting of about 8000 different property types. The data set is created by extracting properties from infoboxes and templates within Wikipedia articles. The *Raw Infobox Property Definition Set* consists of a URI definition for each property as well as a label.

This approach of assigning DBPedia properties to Boxer relations was discarded. This happened for two reasons: Firstly, the property data set only contains a very limited range of possible relations, namely those which also occur in Wikipedia infoboxes and templates. This means, that only a few of the Boxer relations could be linked to DBPedia properties.

Secondly, the property types are not cleaned and therefore very noisy. The property name is often just an abbreviation and the label doesn't contain more information than the URI itself. In addition, property names are not merged, so there are sometimes different property names for the same property type. Property names describing verbs are most often only available in the passive form. Because these data are so noisy, it is really difficult to decide which URIs to assign to which relation.

4.2.2. Assigning RDF WordNet URIs to Boxer relations

The second approach is to interlink Boxer relations with RDF WordNet class types. WordNet is a lexical database for the English language. WordNet contains entries for nouns, verbs, adjectives and adverbs. Each word has different

word senses. For a more detailed description of WordNet, see chapter 2.6.
For using WordNet to assign URIs to Boxer relations, an RDF version of WordNet is required. There are two RDF versions of WordNet available. One is a mapping from WordNet 2.0 to RDF (van Assem et al., 2006) and the other one is a mapping from WordNet 3.0 to RDF (van Assem and van Ossenbruggen, 2011).
RDF WordNet has several URIs for each word, which represent the different word senses. To determine, which one to use, the words have to be disambiguated first. This is done using ukb (Agirre et al., 2009), a WSD tool.

UKB

UKB is a word sense disambiguation tool that performs graph-based WSD with the full graph of WordNet. For each word in a sentence it returns the most likely WordNet sense. UKB performs at a precision of roughly 58%.

For using ukb, the input sentence has to be preprocessed. UKB expects the representation of each input word to have the following form:

```
lemma#pos#id#d
```

These four variables mean the following: *lemma* is the lemma of the word. *pos* is the *part of speech tag* (POS tag) of the word. The possible POS tags are *n* (noun), *v* (verb), *a* (adjective) and *r* (adverb). The distinct ID of a word is *id*, which can be picked freely, *d* is a Boolean variable which indicates if the word should be disambiguated, in which case the variable is 1 or if it should not be disambiguated, in which case the variable is 0.
An example input for the sentence *Google buys YouTube.* is displayed in figure 4.5

```
ctx_01
google#n#w1#1 buy#v#w2#1 youtube#n#w3#1
```

Figure 4.5. UKB input for test sentence *Google buys YouTube.*

So to convert an input sentence to the ukb input format, the lemma and POS tag have to be determined for each word in the sentence. This is done by using the parser output of C&C, which already contains a lemma and a Penn

Treebank POS tag for each word (see also the sample output of C&C parser in figure 4.2). Then, the Penn Treebank POS tags only have to be mapped to of ukb accepted POS tags (*n*, *v*, *a*, *r*).

The input sentences can then be disambiguated with ukb. The output format of each word in the sentence is the following:

```
context_id word_id (concept_id(/weight)?)+ !! lemma
```

The line consists of a context ID, a word ID, the most likely WordNet sense and a lemma. For the sample sentence "Google buys YouTube", the ukb output is displayed in figure 4.6

```
ctx_01 w1  06578905-n !! google
ctx_01 w2  02207206-v !! buy
```

Figure 4.6. UKB output for test sentence *Google buys YouTube.*

As one can see, a sense is only returned for *google* and *buy*. This is because there is no WordNet entry for *youtube*, in which case ukb does not return a line for the word.

RDF WordNet

After the input sentence is disambiguated with ukb, RDF WordNet is used to get URIs for the corresponding senses. As mentioned before, RDF WordNet exists for WordNet versions 2.0 and 3.0. UKB comes with precompiled knowledge bases for WordNet 1.7 and WordNet 3.0, so the latest RDF WordNet version 3.0 is used.

RDF WordNet contains all the information about words and word senses and relations between them that is also available in WordNet. However, the only information needed for extracting URIs are the word sense definitions. RDF WordNet word sense URIs have the following format:

```
wn30:wordsense-LEMMA-POS-NR
```

LEMMA is the lemma of the word, *POS* is the word class and *NR* is the sense number. RDF WordNet contains the word classes *noun*, *verb*, *adverb*, *adjective* and *adjectivesatellite*.

To get an RDF WordNet URI, the ukb output is now interpreted and a URI is formed. The easiest part is the *LEMMA*, which can be directly extracted from the ukb output. The *NR* which is the sense number can however not be directly extracted, since the ukb output only contains the word sense. To get the sense number, the WordNet 3.0 dictionary which is used by ukb can be utilized. In the WordNet 3.0 dictionary, each word is mapped to the corresponding word senses of all word classes. The entry of the word *buy* looks like this:

```
buy 13253751-n:0 02207206-v:102 02284096-v:2 02646757-v:0 02212103-v:0
    00683670-v:0
```

To get the sense number for a word sense, the word senses have to be numbered consecutively like this: For the respective word class, the word senses have to be counted starting with zero until the desired word sense is reached.

The word class *POS* can be mainly derived from ukb output. The only problem is the distinction made by RDF WordNet between the word classes *adjective* and *adjectivesatellite*. This distinction is not made by ukb.

To get the right word class, the full list of word senses is searched. The information available for the search is the lemma and the corresponding sense number. If both an adjective and an adjective satellite are available for a lemma with the same sense number (which is highly unlikely), the adjective is used since adjectives are generally more frequent.

4.2.3. Generating RDF triples

After the RDF WordNet URIs are generated, RDF triples are generated.
The process of generating RDF triples starts similar to the one described in chapter 4.1.3. This means, the input text is processed with Wikifier, C&C and Boxer and DBPedia URIs are inserted. What is different from this process is that in the improved LODifier system, RDF WordNet URIs are now inserted. But because there are already DBPedia URIs linked to subjects and objects, RDF WordNet URIs are only used for the relations. This means that RDF WordNet URIs for nouns are not used.

Figure 4.7 shows an example for the LODifier output of the improved LODifier system for the test sentence *Google buys YouTube.*

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix drsclass: <http://example.org/drsclass/> .
@prefix drsrel: <http://example.org/drsrel/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix wn30: <http://purl.org/vocabularies/princeton/wn30/> .
@prefix dbpedia: <http://dbpedia.org/page/> .
_:var0x0 drsclass:named dbpedia:Google .
dbpedia:Google rdf:type foaf:Organization .
_:var0x1 drsclass:named dbpedia:YouTube .
dbpedia:YouTube rdf:type foaf:Organization .
_:var0x2 rdf:type drsclass:event .
_:var0x2 rdf:type wn30:wordsense-buy-verb-1 .
_:var0x0 drsrel:agent :var0x2 .
_:var0x1 drsrel:patient :var0x2 .
```

Figure 4.7. LODifier output of the improved LODifier system for test sentence "Google buys YouTube".

Chapter 5. Evaluation

In this section, the evaluation of the improved version of LODifier is described. The goal of the evaluation is to determine if the LODifier output could be used as basis for an information retrieval task. The first part of the evaluation process is the manual generation of a corpus (see section 5.1). In section 5.2, the evaluation procedure is described and the results are presented in section 5.3.

5.1. Corpus

Unfortunately, no gold standard is available for this task to evaluate against. So, a corpus is manually created for evaluation. Since the C&C parser and Boxer are both trained on news data, the corpus data are collected in the same domain.
The data was collected from *Google News*, where a news article contains links to other articles with the same topic. In total, the evaluation set consists of 25 articles. This set is divided into five subsets of 5 articles each. These 5 articles share a common topic. For each pair of similar articles, equal relations in these articles were manually identified. The relations usually had different representations. For the relation *has_daughter*, this could be *A has_daughter B* or *B daughter_of A*. Sentences not relevant to the relations were deleted and articles were split in sub articles where possible. Each sub article was then processed with LODifier and the resulting RDF triples were saved in separate files. A graph was created from each file. In total, the evaluation set contains 180 relations.

5.2. Procedure

The evaluation was carried out by three annotators. The third annotator only annotated the data the two other annotators did not agree on. The task was to search the LODifier output files or the graph version of the output files for the relations that occurred in the articles. The annotators received a list of all relations for each pair of sub articles of a topic, the content of the sub articles and the RDF and graph output files. Then, they should write down for each relation if they could find the relation in the RDF representation or not. The annotators were told that it was not important, if Named Entities were properly recognized or if the correct word senses were chosen.

For comparing the results, a string match baseline was computed. The reason for choosing this baseline is to determine whether more information can be retrieved from text when the text is transformed into a more structured format, namely RDF.

Each representation of the relations was transformed into a regular expression. As an example, the relation *has_daughter* between the entities A and B could have the representations *A daughter B*, *B daughter A*, *A B daughter*, *B A daughter*, *daughter A B*. The representation is, however, only counted as valid and transformed into a regular expression, if A and B are nearest neighbors of the relation. This means, that no other entity C may be located between A or B and the relation. The most frequent representation of the relation is then deemed the canonical representation. The baseline for each relation is thus computed as the number of canonical representations divided by 5, which is the total number of equal representations per relation.

5.3. Results

The inter-annotator agreement was 76.1%, which is certainly a good result considering the difficulty of the task. Of all the 180 relations in the corpus, 130 were recognized, which equals to a coverage of 72.2%. Compared to the baseline, which is 36.1%, the coverage has doubled. Separate tables containing evaluation results for each relation can be found in the appendix D.

5.4. Discussion

Although the evaluation results a great improvement over the baseline, there are certain error patterns that occurred. The first very common error is that Boxer fails to dissolve genitives correctly. If the phrase is, e.g., *a friend's baby daughter*, the RDF graph should contain a link between *baby* and *daughter* and between *friend* and *baby daughter*. Unfortunately, the RDF graph only contains a link between *friend's daughter* and *baby*. This means that the *daughter of* relation can't be recognized.

Another related error is that complex named entities are not dissolved correctly. The complex named entity *Black Panther Geronimo Pratt* consists of two named entities: *Black Panther* and *Geronimo Pratt*. In the RDF graph, however, these two named entities are not linked to separate blank nodes, but only to a single one.

But this is not the only error considering named entities: two identical named entities are often not recognized as identical by Boxer, so two separate discourse referents are introduced which leads to two separate nodes in the RDF graph.

Since Boxer doesn't recognize similar named entities as similar, it is no surprise that anaphoras are also rarely resolved properly. If sentences contain long distance dependencies, Boxer sometimes fails to dissolve them correctly. The result is that the arguments of the relation are not connected anymore in the RDF graph.

Chapter 6. Conclusions

In this Bachelor's thesis, I have shown an approach to automatically generate RDF triples out of text combining deep semantic analysis with named entity recognition and word sense disambiguation techniques as well as vocabulary from well-formed knowledge bases. I have provided the theoretical background knowledge, described the resources as well as tools used and defined the method. An evaluation to identify the strengths and weaknesses of the approach was also performed and described.

For future work, the results of the evaluation can be taken into account. A lot of improvement is probably possible concerning the handling of named entities. As already mentioned, Boxer often does not equate similar named entities. The benefit of LODifier is that named entities are not only recognized using the built-in named entity recognizer of Boxer, but that named entities are also recognized using Wikifier and linked to DBPedia URIs. Those DBPedia URIs that are the same for similar named entities could then be used to reduce multiple blank nodes that refer to the same entity to one blank node.

Another improvement that could be made to named entity processing using the DBPedia URIs concerns the issue of complex named entities. Complex named entities are often not linked to separate blank nodes, but only to a single one. If there are now seperate DBPedia URIs for the seperate named entities entailed in the complex named entity, the blank node could be split into these parts.

Other errors LODifier produces are caused by common computational linguistic issues and cannot be eliminated that easily. These are therefore ceded for future research.

References

Agirre, E., de Lacalle, O. L., and Soroa, A. (2009). Knowledge-based WSD and specific domains: performing over supervised WSD. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, IJCAI'2009, Pasadena, USA.

Bedini, I. and Nguyen, B. (2007). Automatic Ontology Generation: State of the Art. Technical report, University of Versailles.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. In *Scientific American*, pages 96–101.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). DBpedia - A crystallization point for the Web of Data. *Web Semant.*, 7:154–165.

Byrne, K. and Klein, E. (2009). Automatic extraction of archaeological events from text. *Technology*.

Clark, S. and Curran, J. R. (2007). Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Curran, J. R., Clark, S., and Bos, J. (2007). Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of the ACL 2007 Demonstrations Session*, ACL'2007 demo, pages 33–36.

Curran, J. R., Clark, S., and Bos, J. (2011). Syntax of Discourse Representation Structures. `http://svn.ask.it.usyd.edu.au/trac/candc/wiki/DRSs`. [Online; accessed July 12, 2011].

Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *International Journal of Human-Computer Studies*, pages 907–928. Kluwer Academic Publishers.

Harrington, B. and Clark, S. (2007). Asknet: Automated semantic knowledge network. In *AAAI*, pages 889–894. AAAI Press.

Heim, P., Lohmann, S., and Stegemann, T. (2010). Interactive Relationship Discovery via the Semantic Web. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *LNCS*, pages 303–317, Berlin/Heidelberg. Springer.

Hitzler, P., Sebastian, R., and Krötzsch, M. (2009). *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, London.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42 of *Studies in Linguistics and Philosophy*. Kluwer, Dordrecht.

Klyne, G. and Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/2006/WD-wordnet-rdf-20060619/`.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38:39–41.

Milne, D. and Witten, I. H. (2008). Learning to link with Wikipedia. In *Proceedings of the ACM Conference on Information and Knowledge Management*, CIKM'2008, Napa Valley, California.

Milne, D. and Witten, I. H. (2011). Wikify service. `http://wdm.cs.waikato.ac.nz:8080/service?task=wikify`. [Online; accessed July 12, 2011].

Park, J., Cho, W., and Rho, S. (2007). Evaluation framework for automatic ontology extraction tools: an experiment. In *Proceedings of the 2007 OTM confederated international conference on On the move to meaningful internet systems - Volume Part I*, OTM'07, pages 511–511, Berlin, Heidelberg. Springer-Verlag.

Ramakrishnan, C., Kochut, K. J., and Sheth, A. P. (2006). A Framework for Schema-Driven Relationship Discovery from Unstructured text. In *Proc. 5th International Semantic Web Conference*, pages 583–596.

van Assem, M., Gangemi, A., and Schreiber, G. (2006). WordNet 2.0 in RDF/OWL. `http://www.w3.org/TR/wordnet-rdf/`. [Online; accessed July 12, 2011].

van Assem, M. and van Ossenbruggen, J. (2011). WordNet 3.0 in RDF. `http://semanticweb.cs.vu.nl/lod/wn30/`. [Online; accessed July 12, 2011].

Wojtinnek, P.-R., Harrington, B., Rudolph, S., and Pulman, S. (2010). Conceptual knowledge acquisition using automatically generated large-scale semantic networks. In Croitoru, M., Ferré, S., and Lukose, D., editors, *ICCS*, volume 6208 of *Lecture Notes in Computer Science*, pages 203–206. Springer.

Appendix A.

Syntax of Boxer Output

Boxer Syntax	Explanation
topic	elliptical noun phrases
thing	used in NP quantifiers: *something*, etc.
person	used in first-person pronouns, *who*-questions
event	introduced by main verbs
group	used for plural descriptions
reason	used in *why*-questions
manner	used in *how*-questions
proposition	arguments of propositional complement verbs
unit of time	used in *when*-questions
location	used in *there* insertion, *where*-questions
quantity	used in *how many*
amount	used in *how much*
degree	
age	
neuter	used in third-person pronouns: it, its
male	used in third-person pronouns: he, his, him
base	
bear	

Table A.1. One-place predicates

Boxer Syntax	Explanation
rel	general, underspecified type of relation
loc_rel	locative relation
role	underspecified role: agent,patient,theme
member	used for plural descriptions
agent	subject
patient	semantic object, subject of passive verbs
theme	indirect object

Table A.2. Two-place predicates

Appendix B.

Source Code

This section describes the format of the LODifier source code, which can be found on the CD.

The LODifier source is located in the folder *lodifier*. The different packages of LODifier are located in the subfolder *src*. In the *pipeline* package, all the different components of LODifier are called. The *ner* package contains the Wikifier module, *preprocessing* contains the tokenizer, C&C and Boxer modules, *wn* contains the WordNet component and *rdf* contains the RDF conversion module. Other packages are *ui*, which contains the User Interface, *data* which stores the output data, *io*, which contains methods for reading and writing files and *conf* which contains the configuration files.

For running LODifier, consult the README-file, which contains information on necessary adjustments. A more detailed description of the source code is available as Javadoc in the subfolder *doc*.

Appendix C.

Evaluation Data

This section describes the format of the evaluation data, which can be found on the CD.

The evaluation data are located in the folder *evaluation*. The subfolder *eval_in* contains all preprocessed input files. Each file has a header which contains relations that appear in all articles. The header is followed by the cropped article text of all five related articles. Each article again has a header that specifies the newspaper and date. The subfolder *eval_out* contains the output RDF and graph files sorted by evaluation set.

Appendix D.

Evaluation Results

Set 1			
Number	Relation	LODifier	Baseline
1	A has B	0.8	0.2
2	A has estranged B	0.8	0.2
3	A's B is guilty of C	0.8	0.2
4	A's B is guilty of C of D	0.8	0.2
5	E has D	0.2	0.4
6	D is one year old	0.4	0.4
7	D fell	0.6	0.2
8	D fell out of F	0.6	0.2
		0.625	0.25

Table D.1. Results for coverage of data set 1

Set 2			
Number	Relation	LODifier	Baseline
1	A is former B	0.2	0.4
2	A died	0.8	0.6
3	A died in C	0.8	0.2
4	A died in a small C	0.8	0.2
5	A died in a small C in D	0.8	0.2
6	A was in E	0.8	0.6
7	A was in E for 27 years	0.8	0.6
8	A was in E for F	0.8	0.6
		0.725	0.425

Table D.2. Results for coverage of data set 2

Set 3			
Number	Relation	LODifier	Baseline
1	A is a B	1	1
2	A has a C	0.4	0
3	A denied D of raping E	0.8	0.8
4	A is from F	1	1
5	A is 52 years old	1	0
6	A was aquitted of G	1	1
		0.867	0.633

Table D.3. Results for coverage of data set 3

Set 4			
Number	Relation	LODifier	Baseline
1	28 A were killed	1	0.6
2	28 A were killed in B	1	0.2
3	B was in C	1	0.4
4	B was between two rival D	1	0.4
5	B was between E	1	0.4
		1	0.4

Table D.4. Results for coverage of data set 4

Set 5			
Number	Relation	LODifier	Baseline
1	A is 29 years old	1	0.4
2	A is from B	1	0.2
3	A is a C	1	0.4
4	A joins D	1	0.4
5	A joins D as a E	1	0.4
		1	0.36

Table D.5. Results for coverage of data set 5