

STATISTICAL EXTRACTION OF
MULTILINGUAL NATURAL LANGUAGE
PATTERNS FOR RDF PREDICATES:
ALGORITHMS AND APPLICATIONS

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

im Fachgebiet

Informatik

Vorgelegt

von **M.Sc Daniel Gerber**

geboren am 16.12.1985 in Bad Schlema, Deutschland

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Ing. habil. Klaus-Peter Fähnrich, Universität Leipzig
2. Prof. Dr. Axel Polleres, Wirtschaftsuniversität Wien

Die Verleihung des akademischen Grades erfolgt mit Bestehen der
Verteidigung am 06.07.2016 mit dem Gesamtprädikat *magna cum laude*

BIBLIOGRAPHIC DATA

TITLE:

Statistical Extraction of Multilingual Natural Language Patterns for
RDF Predicates: Algorithms and Applications

AUTHOR:

Daniel Gerber

STATISTICAL INFORMATION:

143 pages, 29 Figures, 29 tables, 10 listings, 1 appendix, 157 literature
references

SUPERVISORS:

Prof. Dr. Ing. habil. Klaus-Peter Fähnrich
Dr. Axel-Cyrille Ngonga Ngomo

INSTITUTION:

Universität Leipzig, Fakultät für Mathematik und Informatik

TIME FRAME:

January 2011 - Oktober 2015

ABSTRACT

The Data Web has undergone a tremendous growth period. It currently consists of more than 3300 publicly available knowledge bases describing millions of resources from various domains, such as life sciences, government or geography, with over 89 billion facts. In the same way, the Document Web grew to the state where approximately 4.55 billion websites exist, 300 million photos are uploaded on Facebook as well as 3.5 billion Google searches are performed on average every day. However, there is a gap between the Document Web and the Data Web, since for example knowledge bases available on the Data Web are most commonly extracted from structured or semi-structured sources, but the majority of information available on the Web is contained in unstructured sources such as news articles, blog post, photos, forum discussions, etc. As a result, data on the Data Web not only misses a significant fragment of information but also suffers from a lack of actuality since typical extraction methods are time-consuming and can only be carried out periodically. Furthermore, provenance information is rarely taken into consideration and therefore gets lost in the transformation process. In addition, users are accustomed to entering keyword queries to satisfy their information needs. With the availability of machine-readable knowledge bases, lay users could be empowered to issue more specific questions and get more precise answers.

In this thesis, we address the problem of Relation Extraction, one of the key challenges pertaining to closing the gap between the Document Web and the Data Web by four means. First, we present a distant supervision approach that allows finding multilingual natural language representations of formal relations already contained in the Data Web. We use these natural language representations to find sentences on the Document Web that contain unseen instances of this relation between two entities. Second, we address the problem of data actuality by presenting a real-time data stream RDF extraction framework and utilize this framework to extract RDF from RSS news feeds. Third, we present a novel fact validation algorithm, based on natural language representations, able to not only verify or falsify a given triple, but also to find trustworthy sources for it on the Web and estimating a time scope in which the triple holds true. The features used by this algorithm to determine if a website is indeed trustworthy are used as provenance information and therewith help to create metadata for facts in the Data Web. Finally, we present a question answering system that uses the natural language representations to map natural language question to formal SPARQL queries, allowing

lay users to make use of the large amounts of data available on the Data Web to satisfy their information need.

ZUSAMMENFASSUNG

Das Data Web hat eine enorme Wachstumsphase erlebt. Es besteht aktuell aus mehr als 3300 öffentlich zugänglichen Wissensbasen, die Millionen Ressourcen von unterschiedlichen Domänen, wie etwa Biowissenschaften, Verwaltung und Geografie, mit über 89 Milliarden Fakten beschreiben. In gleicher Weise wuchs das Document Web zu dem Zustand in dem ungefähr 4,55 Milliarden Webseiten existieren und im Tagesdurchschnitt 300 Millionen Fotos auf Facebook hochgeladen und 3,5 Milliarden Google Suchanfragen durchgeführt werden. Trotzdem existiert eine Diskrepanz zwischen dem Document Web und dem Data Web, weil zum Beispiel im Data Web verfügbare Wissensbasen im Regelfall nur von strukturierten beziehungsweise teilweise strukturierten Datenquellen extrahiert worden sind. Allerdings befindet sich der Großteil der Daten im Web in unstrukturierten Datenquellen, wie etwa in Nachrichtenartikeln, Blogs, Fotos, Forendiskussionen, etc. Als ein Resultat dieser Diskrepanz fehlt den Daten im Data Web nicht nur der Großteil der verfügbaren Informationen, sondern lassen Aktualität vermissen, da typische Extraktionsmethoden zeitaufwendig sind und deshalb nur periodisch ausgeführt werden können. Des Weiteren werden Provenienzinformationen nur selten berücksichtigt und gehen damit im Transformationsprozess verloren. Außerdem sind Nutzer an Schlüsselwort-Anfragen gewöhnt, um ihr Informationsbedürfnis zu befriedigen. Mit der Verfügbarkeit von maschinenlesbaren Wissensbasen werden auch unerfahrene Nutzer in die Lage versetzt, spezifischere Fragen zu stellen und genauere Antworten zu erhalten.

In dieser Arbeit beschäftigen wir uns mit dem Problem der Relationsextraktion, eine der wichtigsten Herausforderungen, um die Lücke zwischen Document Web und Data Web zu schließen. Dazu stellen wir vier Methoden vor. Erstens zeigen wir einen Distant Supervision Ansatz, der es erlaubt multilinguale natürlichsprachliche Repräsentationen von formalen Relationen zu ermitteln, die bereits im Data Web enthalten sind. Wir nutzen diese natürlichsprachlichen Repräsentationen, um Sätze im Document Web zu finden, die unbekannte Instanzen dieser Relation zwischen zwei Entitäten enthalten. Zweitens beschäftigen wir uns mit dem Problem der Datenaktualität, indem wir ein Echtzeit-RDF-Extraktionsframework für Datenströme vorstellen und dieses Framework anwenden, um RDF aus RSS Nachrichten-Feeds zu extrahieren. Drittens präsentieren wir ein neuartiges Fact Validation Verfahren, basierend auf natürlichsprachlichen Repräsentationen formaler Relationen, das nicht nur in der Lage ist, ein gegebenes Tripel zu verifizieren beziehungsweise zu widerlegen, sondern auch vertrauenswürdige Quellen dafür im Web

findet und zusätzlich ein Zeitintervall bestimmt, in dem das Triple wahr ist. Die Merkmale, die von diesem Algorithmus genutzt werden, um zu bestimmen, ob eine Webseite vertrauenswürdig ist, werden als Provenienzinformationen genutzt und helfen somit Metadaten für Fakten im Data Web zu generieren. Zum Abschluss präsentieren wir ein Question Answering System, das natürlichsprachliche Repräsentationen nutzt, um natürlichsprachliche Fragen auf formale SPARQL-Anfragen abzubilden und es damit unerfahrenen Nutzern ermöglicht, die riesigen Datenvolumen im Data Web nutzbar zu machen um deren Informationsbedürfnis zu befriedigen.

PUBLICATIONS

This thesis is based on the following publications and proceedings. References to the appropriate publications are included at the respective chapters and sections.

AWARDS AND NOTABLE MENTIONS

- **Best Research Paper Award** at ISWC 2014 for *AGDISTIS - Graph-Based Disambiguation of Named Entities using Linked Data*.
- **Best Student Paper Award** at ESWC 2014 for *Hybrid Acquisition of Temporal Scopes for RDF Data*.
- **Spotlight Paper** at ISWC 2012 for *DeFacto - Deep Fact Validation*.

JOURNALS, PEER-REVIEWED

- Journal of Web Semantics: “*DeFacto - Temporal and Multilingual Deep Fact Validation*” [Gerber et al., 2015]

CONFERENCES, PEER-REVIEWED

- 18th International Conference on Knowledge Engineering and Knowledge Management: “*Extracting Multilingual Natural Language Patterns for RDF Predicates*” [Gerber and Ngonga Ngomo, 2012]
- 11th International Semantic Web Conference, 2012: “*DeFacto - Deep Fact Validation*” [Lehmann et al., 2012b]
- 11th International Semantic Web Conference, 2012: “*DEQA: Deep Web Extraction for Question Answering*” [Lehmann et al., 2012a]
- 20th World Wide Web Conference, 2012: “*Template-based question answering over RDF data*” [Unger et al., 2012]
- 12th International Semantic Web Conference, 2013: “*Real-time RDF extraction from unstructured data streams*” [Gerber et al., 2013]
- 4th Conference on Knowledge Engineering and Semantic Web, 2013: “*TBSL Question Answering System Demo*” [Höffner et al., 2013]

- 22nd World Wide Web Conference, 2013: *“Sorry, I don’t speak SPARQL — Translating SPARQL Queries into Natural Language”* [Ngonga Ngomo et al., 2013a,b]
- 8th International Conference on Language Resources and Evaluation: *“N₃ - A Collection of Datasets for Named Entity Recognition and Disambiguation in the NLP Interchange Format”* [Röder et al., 2014]
- 11th Extended Semantic Web Conference, 2014: *“Hybrid Acquisition of Temporal Scopes for RDF Data”* [Rula et al., 2014]
- 13th International Semantic Web Conference, 2014: *“AGDISTIS - Agnostic Disambiguation of Named Entities Using Linked Open Data”* [Usbeck et al., 2014]

BOOK CHAPTERS, PEER-REVIEWED

- Chapter *“From RDF to Natural Language and Back”* in *Towards the Multilingual Semantic Web*, 2014 [Gerber and Ngonga Ngomo, 2013]

WORKSHOPS, PEER-REVIEWED

- 1st Workshop on Web Scale Knowledge Extraction at International Semantic Web Conference, 2011: *“Bootstrapping the Linked Data Web”* [Gerber and Ngonga Ngomo, 2011]

UNPUBLISHED PAPERS

- *“Mapping text to ontology with DBpedia Lemon and BOA”* [Lukovnikov et al., 2014]

ACKNOWLEDGMENTS

I would like to thank all of my colleagues with whom I jointly wrote the papers and articles that led to this work: Axel Ngonga, Sebastian Hellmann, Lorenz Bühmann, Tomasso Soru, Jens Lehmann, Mohamed Morsey, Christina Unger, Philipp Cimiano, Konrad Höffner, Michael Röder, Ricardo Usbeck, Anisa Rula, Matteo Palmonari and Denis Lukovnikov. Special thanks goes to my direct supervisor Dr. Axel-Cyrille Ngonga Ngomo. He continuously supported me through my Ph. D. work, gave advice and recommendations for further research steps and improvements. I would like to thank Prof. Dr. Ing. habil. Klaus-Peter Fähnrich for his scientific experience with the efficient organization of the process of a Ph. D. thesis and Prof. Dr. Sören Auer who helped me to get a scholarship, which made this thesis possible. This thesis was funded by the Medienstiftung der Sparkasse Leipzig. I would like to thank Stephan Seeger for providing me with that opportunity. Also, I would like to thank Michael Martin who supervised my master thesis and guided my way to this dissertation. I would like to thank my close friend Robert Remus who challenged my ideas constantly and always provided valuable feedback. Finally I would like to thank my family, Swen, Sylvana, David, Irene and Rolf who have supported me in all respects over the period of this dissertation and beyond.

Thank You.

CONTENTS

i	ALGORITHMS FOR THE STATISTICAL EXTRACTION OF MULTILINGUAL NATURAL LANGUAGE PATTERNS	1
1	INTRODUCTION	2
1.1	From the World Wide Web to the Semantic Web	2
1.2	Current Drawbacks of the Semantic Web	4
1.3	Chapter Overview	7
1.4	Author Contributions	8
2	PRELIMINARIES	9
2.1	Semantic Web	9
2.1.1	RDF, RDFS, OWL	10
2.1.2	SPARQL	13
2.1.3	DBpedia, YAGO(2) and Freebase	14
3	BOOTSTRAPPING LINKED DATA	18
3.1	Introduction	19
3.2	Related Work	22
3.3	The BOA Framework	28
3.3.1	Corpus Extraction	28
3.3.2	Knowledge Acquisition	29
3.3.3	Pattern Search	29
3.3.4	Pattern Scoring	31
3.3.5	RDF Generation	34
3.3.6	Evaluation	34
3.4	Multilingual Extension of BOA	42
3.4.1	Overview	42
3.4.2	Pattern Extraction	42
3.4.3	Feature Extraction	45
3.4.4	Scoring Approach	47
3.4.5	RDF Generation	48
3.4.6	Evaluation	49
3.5	Conclusion	52
4	REAL-TIME RDF EXTRACTION FROM UNSTRUCTURED DATA STREAMS	54
4.1	Introduction	55
4.2	Overview	56
4.2.1	Data Acquisition	56
4.2.2	Deduplication	57
4.2.3	Pattern Search and Filtering	57
4.2.4	Pattern Refinement	58
4.2.5	Pattern Similarity and Clustering	60
4.2.6	Cluster Labeling and Merging	60

4.2.7	Mapping to RDF and Publication on the Data Web	61
4.2.8	Linking	63
4.3	Evaluation	63
4.3.1	URI Disambiguation	64
4.3.2	Pattern Clustering	65
4.3.3	RDF Extraction and Linking	66
4.3.4	Scalability	67
4.4	Related Work	68
4.5	Conclusion	71
ii	APPLICATIONS OF MULTILINGUAL NATURAL LANGUAGE PATTERNS	72
5	DEFACTO - DEEP FACT VALIDATION	73
5.1	Introduction	74
5.2	Related Work	77
5.3	Approach	80
5.4	DeFacto – Deep Fact Validation	85
5.4.1	BOA	85
5.4.2	Trustworthiness Analysis of Webpages	88
5.4.3	Features for Deep Fact Validation	90
5.4.4	Evaluation	91
5.5	DeFacto – Multilingual and Temporal Extension	96
5.5.1	Training BOA for DeFacto	96
5.5.2	Automatic Generation of Search Queries	97
5.5.3	BOA and NLP Techniques for Fact Confirmation	97
5.5.4	Trustworthiness Analysis of Webpages	99
5.5.5	Features for Deep Fact Validation	100
5.5.6	Temporal Extension of DeFacto	100
5.5.7	FactBench - A Fact Validation Benchmark	103
5.5.8	Evaluation	107
5.6	Conclusion	117
6	TEMPLATE-BASED QUESTION ANSWERING OVER RDF DATA	118
6.1	Introduction	119
6.2	Overview	121
6.3	Evaluation and Discussion	122
6.4	Prototype	124
6.5	Related Work	124
6.6	Conclusion	125
7	MAPPING TEXT TO ONTOLOGY WITH DBPEDIA LEMON AND BOA	127
7.1	Lexical Pattern Library and Seed Lexicon	127
7.2	Combining lemon seeds and BOA	129
7.2.1	Finding mapping extraction patterns	129
7.2.2	Extracting new lexical mappings	130

7.3	Evaluation	131
7.4	Conclusion and future work	131
8	CONCLUSION AND FUTURE WORK	133
8.1	Summary	133
8.2	Future Work	136
iii	APPENDIX	138
A	APPENDIX	139
A.1	Template Based Question Answering	139
	BIBLIOGRAPHY	141

LIST OF FIGURES

Figure 1	The Semantic Web Stack after Tim Berners-Lee	10
Figure 2	Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/	11
Figure 3	Screenshot of the DBPedia graphical user interface.	15
Figure 4	Screenshot of the YAGO(2) ontology browser.	16
Figure 5	Screenshot of the Freebase user interface depicting Tim Berners-Lee’s educational stages in an <i>n-ary</i> relational format.	16
Figure 6	Web-view of the Google Knowledge Graph based on Freebase data.	17
Figure 7	Usage of content languages for webpages. (W3Techs.com, 21 November 2013)	20
Figure 8	Overview of the BOA approach.	28
Figure 9	Distribution of parameters used to compute the <i>support</i> of patterns in log-log scale. The y-axis shows the number of patterns	33
Figure 10	Screenshot of the BOA frontend	35
Figure 11	Overview of the knowledge extraction on DBpedia	36
Figure 12	Overview of the BOA approach. Parts that changed from the previous version are marked red.	43
Figure 13	Distribution of patterns per pattern mapping in logarithmic scale. Each vertical line represents one pattern mapping.	44
Figure 14	Screenshot of the BOA pattern library Web interface.	49
Figure 15	Learning curves of BOA’s neural networks. The x-axis shows the number of input neurons and the y-axis shows the achieved accuracy.	50
Figure 16	Overview of the generic time slice based stream processing.	56
Figure 17	Runtimes for different components and corpora (1% left, 10% right, 100% bottom) per iteration.	69
Figure 18	Number of patterns (log scale) and patterns with $ S'_\theta > 1$ (Patterns ₊) for iterations and test corpus.	70

Figure 19	Number of clusters (log scale) and clusters with $ \mathcal{C} > 1$ (Cluster ₊) for iterations and test corpus. 70
Figure 20	Overview of the DeFacto architecture. 81
Figure 21	Overview of the provenance schema which is used to export the validation result of DeFacto as RDF, given the input fact Albert Einstein, award, Nobel Price in Physics. 83
Figure 22	Screenshot of the extended DeFacto Web interface. 84
Figure 23	Distribution of year numbers in World Wide Web. Shows approximate number of Google search results. Outliers from left to right, 1931, 1972 and 2000. As comparison 'EU' has about 2.280.000.000 and 'Obama' 478.000.000 hits. 106
Figure 24	Overview of time points and time periods in the FactBench train set. 108
Figure 25	Accuracy results for learned J48 <i>mix</i> classifier on correct subset of the test set. The abbreviation ml indicates that multilingual (English, French, German) search results and surface forms were used, en is limited to English only. 111
Figure 26	A plot showing the proportion of correctly classified facts (y-axis) for the FactBench <i>mix-correct-test-set</i> using the J48 classifier. The time intervals (x-axis) are buckets of ten years, e.g. 1910 stands for all years from 1910 to 1919. Results for the multilingual and English-only setting of DeFacto are shown. 115
Figure 27	Overview of the template based SPARQL query generator. 121
Figure 28	Screenshot of prototype available at http://autosparql-tbsl.dl-learner.org . 124
Figure 29	Overview of DBpedia lemon lexicon for word "wife". 128

LIST OF TABLES

Table 1	Comparison of the WWW and the Semantic Web. 4
Table 2	Overview of relevant OWL concepts 13
Table 3	Comparison of DBpedia, YAGO2 and Freebase 17

Table 4	Comparison of algorithms related to BOA approach. 27
Table 5	Example sentences for pattern search. 31
Table 6	Corpora statistics. (All figures in millions.) 37
Table 7	Evaluation results for top-1 pattern. 37
Table 8	Evaluation results for top-2 pattern. 38
Table 9	Overview of extraction statistics of first iteration for <i>en-wiki</i> . 39
Table 10	Overview of extraction statistics of first iteration for <i>en-news</i> . 39
Table 11	Top-2 natural language representations for six most used relations in evaluation. “—” means that no natural language representation was found, patterns in brackets are next in line but were not used for the evaluation because they did not fulfill the threshold requirements. 40
Table 12	Triples extracted from evaluation data set not present in DBpedia. 41
Table 13	Statistical overview of German and English surface forms. 43
Table 14	Example sentences for pattern search. 45
Table 15	Results of the first iteration of the BOA framework. 51
Table 16	Accuracy of RDF Extraction for subjects (S), predicates (P) and objects (O) on 1% dataset with varying cluster sizes E_i . 67
Table 17	Number of non-duplicate sentences in 1% of the data extracted from 1457 RSS feeds within a window of 10 time slices (2h each). The second column shows the original number of sentences without duplicate removal. 67
Table 18	Example for linking between RdfLiveNews and DBpedia. 68
Table 19	Performance measures for several classifiers on the fact confirmation task (AUC = area under the ROC curve, RMSE = root mean squared error). 88
Table 20	Classification results for the different evaluation sets. 95
Table 21	Proofs with language distribution used to train fact classifier. 98
Table 22	Example list of patterns for relations <i>publication</i> (top) and <i>marriage</i> (bottom). 99
Table 23	Example list of temporal patterns extracted from English, German and French Wikipedia. 99

Table 24	Overview of all correct facts of the training and testing set (train/test). 104
Table 25	Classification results for FactBench test sets (C = correctness, P = precision, R = recall, F ₁ = F ₁ Score, AUC = area under the curve, RMSE = root mean squared error). 110
Table 26	Classification results for FactBench mix test set on English language only. 112
Table 27	Overview of the time-period detection task for the FactBench training set with respect to the different normalization methods. ml (multilingual) indicates the use of all three languages (en,de,fr). 114
Table 28	Overview of the <i>domain</i> -normalization on the FactBench test set. ml (multilingual) indicates the use of all three languages (en,de,fr). C _(S E) shows the number of correct start and end years, P ₇₅ is the number of time-periods possible to detect correctly and A is the accuracy on P ₇₅ . 116
Table 29	This table shows precision and recall values for each processed question (i.e. all questions that do not require the YAGO or FOAF namespace). For questions with no precision and recall specified, no query was constructed. Questions printed in cells with red background were not parsed, questions in white cells succeeded and for questions in lightgray cells queries with quality equal or close to the Gold query were built, while questions in yellow cells fail due to a query selection problem and questions in orange cells fail due to some entity identification problem. 140

LISTINGS

Listing 1	URI scheme after Berners-Lee et al. [2005] 10
Listing 2	Example SPARQL query. 14
Listing 3	RDF snippet used for pattern search 30
Listing 4	RDF snippet generated by BOA 34
Listing 5	SPARQL query template used for knowledge acquisition. 36
Listing 6	RDF snippet used for pattern search 44

Listing 7	RDF extracted by BOA. If not stated otherwise, all instances and properties use the DBpedia namespace. 51
Listing 8	Example RDF extraction of RdfLiveNews 62
Listing 9	Input data for Defacto. 85
Listing 10	Example of a fact in FactBench. 106

Part I

ALGORITHMS FOR THE STATISTICAL EXTRACTION OF MULTILINGUAL NATURAL LANGUAGE PATTERNS

The first part of this thesis gives an overview of the current state of the Semantic Web. It highlights the drawbacks the current Semantic Web has with respect to the World Wide Web. These drawbacks simultaneously constitute the motivation of this thesis. This part is focused on algorithms to extract natural language patterns from free text to eliminate current problems such as actuality and completeness. We present algorithms that use the data already available on the Data Web to extract patterns for the English as well as the German language. These multilingual patterns are used to extract new knowledge and therewith expand the Data Web.

INTRODUCTION

1.1 FROM THE WORLD WIDE WEB TO THE SEMANTIC WEB

Since its invention by Tim Berners-Lee and Robert Cailliau in 1991 and its publication to the general public later that year, the *World Wide Web* (WWW) has changed the daily lives of billions of people. Its introduction had an impact comparable to that of other inventions like the steam engine, the car or transistors. It did not only change our lives but also our mindset. The ways in which we communicate and collaborate ever since, not only with other people but also with machines, have changed dramatically. In the course of its history, the Web has transformed from a text- and line-based interface for a handful of websites for high energy particle physicists into an ubiquitous technology accompanying us everywhere and anytime in the form of laptops, phones, watches, cars and even refrigerators. After its first ten years of infancy and its survival of the dot-com bubble burst in the late years of the last and early years of the new millennium, key players such as Google, Amazon, Ebay and Yahoo! crystallized. According to Tim O'Reilly¹, "The central principle behind the success of the giants born in the Web 1.0 era who have survived to lead the Web 2.0 era appears to be this, that they have embraced the power of the web to harness collective intelligence . . .". This collective intelligence can generally be seen as the aggregation of all contributions made to the WWW from all its users. This intelligence, e.g. harnessed by Google through applying PageRank to the user-generated hyper-link structure of the Web or Amazon's focus on customer satisfaction for example via recommendation systems based on past purchases, is also what made the Web 2.0 so successful and growing exponentially. With the introduction of RSS feeds as well as blogs the power was moved away from the server and admin side to user's desktops. This movement drastically increased online collaboration and therewith enabled the rise of user-generated content websites such as Wikipedia, Flickr and Youtube. Unfortunately, with the growing number of web startups and content on the web, mostly generated by lay users, new problems arose. How could data be reused across application, enterprise and community boundaries? How could users find what they were searching for, or an even harder question to answer, what they meant? At the heart of these issues lays the fact, that the Web was, and still is, generated by humans for humans. Documents, and that is all what the Web at that time contained, are not

¹ <http://oreilly.com/pub/a/web2/archive/what-is-web-20.html>

automatically processable by machines. In this *Web of Documents* it is not possible for software algorithms to distinguish between sites focusing on Anne Hathaway the American actress or Anne Hathaway the spouse of 18-year-old William Shakespeare without context. Since data is kept in silos (of the respective application) it is not possible to reuse them. For example the geo- and face-tagged photos of the last family holiday lose their metadata after uploading them to Flickr, because two different applications do not speak the same language. To solve these fundamental problems, Berners-Lee et al. [2001] published their vision of a semantic layer on top of the already existing link infrastructure of the WWW, dubbed the *Semantic Web*. In this vision, communication, e.g. the negotiation of a doctor's appointment, will not need to be handled by humans talking to humans, but by software agents talking to software agents. Even hardware items such as stereos, telephones and laptops can have agents publishing and discovering semantic services to, as exemplified by the vision, send a *turn down speaker volume* request to all nearby devices while receiving an incoming call. Sir Tim Berners-Lee formulated (the second part of) his dream in Berners-Lee and Fischetti [1999] as follows:

“Machines become capable of analyzing all the data on the Web - the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy, and our daily lives will be handled by machines talking to machines, leaving humans to provide the inspiration and intuition. The intelligent ‘agents’ people have touted for ages will finally materialize. This machine-understandable Web will come about through the implementation of a series of technical advancements and social agreements that are now beginning.”

One of the key technical advancements to fulfill this ultimate future is the Resource Description Framework (RDF, see Chapter 2), a W3C standard which defines the RDF graph data model (as well as an abstract syntax and semantics) that can be used to express arbitrary information in a machine-readable format to reuse data across application, enterprise, and community boundaries. If this machine-readable data is exposed to machines and combined with access to a set of inference rules – i.e. to infer automatically that, if Berlin is a city in Germany and Germany is located in Europe, then Berlin is also located in Europe – would transform the Web of Documents into the *Data Web*. The Semantic Web community was faced with a lot of criticism, be it the question if the creation of web-scale ontologies is possible or if databases (triple stores) will ever scale to billions of triples, even Shadbolt et al. stated in 2006: “This simple idea, however, remains largely unrealized.” Fortunately, this early criticism is no longer valid.

	WWW	Semantic Web
Actuality	✓	✗
Editability	✓	✗
Usability	✓	✗
Queryability	✗	✓
Reasoning	✗	✓
Scalability	✓	✗
Design	For humans	Humans and machines
Integrability	✗	✓

Table 1: Comparison of the WWW and the Semantic Web.

Triple stores have successfully loaded and queried over 1 trillion RDF triples², the Linked Open Data Cloud³ has evolved from containing 12 datasets in 2007 to containing more than 3300 datasets and over 89 billion triples in 2015,⁴ governments all over the world started to provide their data to the general public, Facebook’s Open Graph Protocol⁵ allows any webpage to be part of their social graph and Google who introduced *schema.org*, a lightweight vocabulary to annotate websites, reported that over 15% of all sites/pages now have embedded *schema.org* vocabulary terms, which totals to 5 million sites and over 25 billion entity references⁶ etc.

1.2 CURRENT DRAWBACKS OF THE SEMANTIC WEB

These achievements can be seen as a great success for the Semantic Web, but there are still a number of problems residing within its current state, which will be the motivation of this thesis:

COMPLETENESS While the Web 2.0, or Web of Documents, was and still is targeted towards humans, the Data Web aims at providing knowledge both in human- and machine-readable form. The methods developed to populate the Data Web, e.g. DBpedia [Lehmann et al., 2013] or LinkedGeoData [Auer et al., 2009; Stadler et al., 2012], have relied on the extraction and transformation of data from structured or semi-structured data sources. The vast majority of data or in other words collective intelligence, accumulated in blogs, forums, Wikipedia articles, news sites and so on is not integrated into the

² http://franz.com/about/press_room/trillion-triples.lhtml

³ <http://lod-cloud.net/>

⁴ Data provided by <http://stats.lod2.eu/> retrieved 28.07.2015.

⁵ <http://ogp.me>

⁶ Data presented at ISWC 2013, http://videlectures.net/site/normal_dl/tag=817824/iswc2013_guha_tunnel_01.pdf page 23

Data Web. For example, the number of scanned books by Google is estimated to be over 20 million⁷ and the number of medical articles indexed by PubMed in 2013 is over 1.1 million⁸. Researchers, especially in the Artificial Intelligence and Natural Language Processing community have been building machine-readable knowledge bases, such as Cyc [Lenat, 1995] and WordNet [Fellbaum, 1998] for decades. With this incredibly huge amount of available data and the steady increase in computational power and storage, it becomes possible to build web-scale knowledge base construction algorithms extracting knowledge from unstructured data. The data produced by these algorithms can be used in manifold ways. We could build smarter applications than “Eugene Goostman”, the first chat bot able to pass a version of the Turing Test⁹ or answer complex queries from life scientists who want to query an integrated database of genom, protein and drug data [Tsatsaronis et al., 2015]. Finally we could build better question answering systems like Watson [Ferrucci et al., 2010] who support us in everyday life and might be one day the software agents dreamt of in the Semantic Web vision.

ACTUALITY As previously mentioned, the methods developed to populate the Semantic Web rely on input from semi-structured and structured knowledge sources. These sources usually do not provide an API to their live data, but for the most time provide backups in the form of database dumps. These dumps are generated in a periodical manner, but, for that reason, fail to provide up-to-date data. Also, these methods are not fully automated but have to be carried out by academic personal on top of their actual research work. For example the last four releases of DBpedia had a release cycle of about 10 months and the last LinkedGeoData release is already 10 months old. For applications with strong timeliness requirements like decision support, where software agents help humans to decide on critical issues such as whether to buy stock or not or even how to plan their drive through urban centers, out-of-date data of this magnitude is unacceptable. For the vision of the Semantic Web to succeed, future applications need to have access to up-to-date data as regular Web 2.0 applications have now.

PROVENANCE The future agents of the Semantic Web consume data from many different sources. The agent from the example above has to find doctors with a rating of excellent or very good. If the service providing the ratings cannot be trusted then the results are unusable. An important tool to measure trustworthiness and data quality in the Data Web is provenance [Hartig, 2009]. A multitude

⁷ <http://searchengineland.com/google-books-fair-use-177093>

⁸ <http://dan.corlan.net/cgi-bin/medline-trend?Q=>

⁹ <http://www.reading.ac.uk/news-and-events/releases/PR583836.aspx>

of standards have been proposed to express provenance information in RDF. For example the Dublin Core Metadata Initiative (DCMI) released the DCMI Metadata Terms [DCMI Usage Board, 2006] and the World Wide Web Consortium (W3C) released the provenance ontology PROV-O [Belhajjame et al., 2012]. Still, only few knowledge bases actually provide provenance information. For instance, less than 3% of the more than 607.7 million RDF documents indexed by Sindice¹⁰ in June 2012 contained metadata such as creator, created, source, modified, contributor or provenance.¹¹ This coverage increased to around 10% for the 708.26 million documents indexed by Sindice in December 2013, but is still far from optimal. The WikiData project, a collaboratively edited, machine-readable knowledge base operated by the Wikimedia Foundation, also has only about 12% of their statements pointing to sources outside of Wikipedia¹². Despite the fact that these numbers are growing, it is still far from generating (data, user or service) trust levels reached by some Web 2.0 platforms, such as Ebay, Reddit or Google (PageRank).

QUALITY The size of the Semantic Web, i.e. the amount of triples and different data sets available, has exceeded by any standards what anybody could have imagined in the early days. Additionally, the logical foundations of RDF and especially the description logic based OWL (Web Ontology Language) make knowledge bases very susceptible to errors. The principle of explosion [Carnielli and Marcos, 2001], saying that every statement can be proven from a contradiction, makes reasoning over inconsistent knowledge bases a demanding task. Also, question answering systems would be of no use, if they were correctly mapping natural language to a formal query language and the underlying database contained errors. Recent research has shown that for example a lot of RDF data not only contains syntactical, but also a large number of semantical errors. From 2122 datasets, analyzed by Demter et al. [2012], 1185 datasets (56.1%) contained errors. [Hogan et al., 2010] showed that in a random sample of 55000 RDF documents (approximately 12.5 million triples) 14.3% of all triples use undeclared properties, 8.1% use undeclared classes and 1,329 members of disjoint classes can be found. Zaveri et al. [2013] evaluated the quality of DBpedia and observed a 11.93% error rate through manual evaluation. A semi-automatic analysis revealed 200,000 violations of property characteristics. To clean knowledge bases already created from semi-structured sources and validate the output of algorithms for the extraction of knowledge from unstructured sources, fact validation algorithms, on a semantic level, have to be researched in more detail.

¹⁰ <http://www.sindice.com>

¹¹ Data retrieved on June 6, 2012.

¹² Data retrieved from <https://tools.wmflabs.org/wikidata-todo/stats.php> on 26th June, 2014.

After only 25 first years of its existence, the Web has become the biggest achievement mankind has ever made. As [Hendler and Golbeck \[2008\]](#) have shown, the power of the Web is enhanced through the network effect produced as resources link to each other with the value determined by Metcalfe's law [[Shapiro and Varian, 1998](#)]. The law states that the value of a network is proportional to the square of the number of connected resources of the system. Therefore, the goal of this thesis is to increase the number of correct and decrease the number of incorrect triples in the Data Web, by harvesting the collective knowledge accumulated by millions of users to further increase the network effect, making the Semantic Web as successful as the Web 2.0.

1.3 CHAPTER OVERVIEW

This thesis is divided into two parts. *Part I* introduces the Semantic Web and its core technologies as well as underlying knowledge bases utilized in this thesis. It secondly focuses on the first two aforementioned problems, *Completeness* and *Actuality*, by extracting multilingual natural language patterns for formal relations and using them to further extract new knowledge with the help of statistical and machine learning methods. We apply these methods to two different text domains: firstly to general, encyclopedic knowledge and secondly to newspapers covering a wide variety of news domains such as politics, sports, business and travel etc. Additionally, we start to investigate the effects of our methods in a *Closed Relation Extraction* setting in Chapter 3, where we have a set of predefined relations for which we want to extract new instances. Our approach can be used to bootstrap the Data Web, since it uses its data to generate even more knowledge, which is directly insertable into it. As a result of our method we provide a multilingual knowledge base of natural language representations for predicates found on the Data Web, a set of features that can be used to distinguish high-quality from poor-quality patterns and show that we can extract knowledge from text with high accuracy. Then, we explore an *Open Relation Extraction* setting in Chapter 4, where no predefined relations are given and have to be found in continuous text, grouped and linked to existing knowledge bases.

Part II of this thesis focuses on the application of the learned natural language to formal relations mapping in three different use cases. Chapter 5 tackles the problems *Provenance* and *Quality* by using the multilingual mappings to validate facts. Chapter 6 and 7 focus on the ontology linking problem, i.e. finding what vocabulary element from an OWL ontology is the target of referral, in a question answering and dictionary population problem space.

This thesis is, for consistency and redundancy reasons, generally structured in a way that the main chapters (Chapters 3-5) are self contained. This means that e.g. the related work as well as the evaluation sections for the corresponding algorithms are included within the chapters. Chapter 3 and Chapter 5 are build iteratively. This results in a two-tier structure where first, the findings for a non multilingual (English) version of the respective algorithms are presented. Based on these findings and the gained insights, we extend the previously presented methodology to support multiple languages in the second tier.

This thesis concludes with Chapter 8 which contains a summary of the presented algorithms for multilingual pattern extraction. Furthermore, it provides pointers to future work.

1.4 AUTHOR CONTRIBUTIONS

Parts of this thesis have been joint work and have been previously published in the papers named in the publication list above. The contributions of this thesis' author are clearly marked at the beginning of every main chapter (Chapter 3-7). Wherever contributions of others are involved, every effort is made to indicate this clearly.

PRELIMINARIES

In this chapter we first give a brief introduction to the idea, history and vision of the Semantic Web. We then describe its core building blocks such as URIs, RDF(S) and OWL. Furthermore, we will present SPARQL, a formal language to query RDF datasets and its importance for this thesis. In the final section we give an overview of current large-scale knowledge bases like DBpedia, YAGO and Freebase and show their importance as invaluable resources, thereby providing background knowledge for the machine learning algorithms presented in Chapters 3, 4 and 5.

2.1 SEMANTIC WEB

With the introduction of the World Wide Web in 1989, with its original goal to foster the exchange of scientific documents and its subsequent rapid growth, arose a multitudinous amount of interlinked documents within only a few years. Those documents, however, contained a vast amount of information, which in general, could only be understood by humans. The idea of the Semantic Web is not the replacement, but the extension of the available network to empower machines to not only present but also to process these information. Tim Berners-Lee outlined this concept in [Berners-Lee et al. \[2001\]](#) as follows:

“The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.”

This definition has been extended by the World Wide Web Consortium (W3C), which plays a leading role in standardizing the applied technologies:

“The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners.”

The technologies which emerged after the first presentation of the Semantic Web, e.g. *URIs*, *RDF*, *SPARQL* and *OWL*, have been hierarchically organized in the so called *Semantic Web Stack*, shown in Figure 1. These still evolving technologies are of major importance for fulfilling

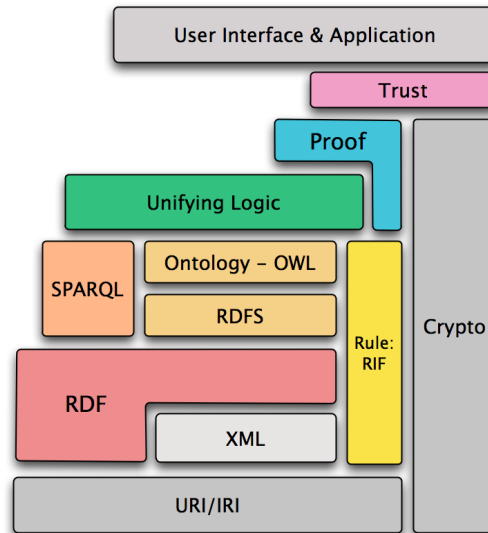


Figure 1: The Semantic Web Stack after Tim Berners-Lee

the vision imagined in [Berners-Lee et al., 2001] and therewith act as a foundation for this thesis.

2.1.1 *RDF, RDFS, OWL*

UNIFORM RESOURCE IDENTIFIER Uniform Resource Identifiers (URIs) build the foundation of the Semantic Web technology stack and therewith form the link between RDF and the Web. With their help, it is possible to unambiguously define and reference abstract as well as concrete concepts on a global level. In contrast, this is not possible with relational or XML based systems. In the context of identifying documents in the WWW with URIs, it is also called Uniform Resource Locator (URL). URIs are constructed by the scheme defined in Berners-Lee et al. [2005] and shown in Listing 1.

```

1 URI      = scheme ":" hierarchy-part [ "?" query ] [ "#" fragment ]
2
3 hierarchy-part = "://" authority path-abempty
4               / path-absolute
5               / path-rootless
6               / path-empty

```

Listing 1: URI scheme after Berners-Lee et al. [2005]

The parts *scheme* and *hierarchy-part* always have to occur, whereas the *query* and *fragment* parts are optional. Furthermore, the *scheme* part determines the interpretation of the substring after the colon. Typical in the WWW occurring schemes are *http* for the description of web sites, *mailto* for email addresses or *doi* for description of digital objects. In the context of the Semantic Web, URIs are above all used to define concepts like *places*, *organizations* or *persons* but also to

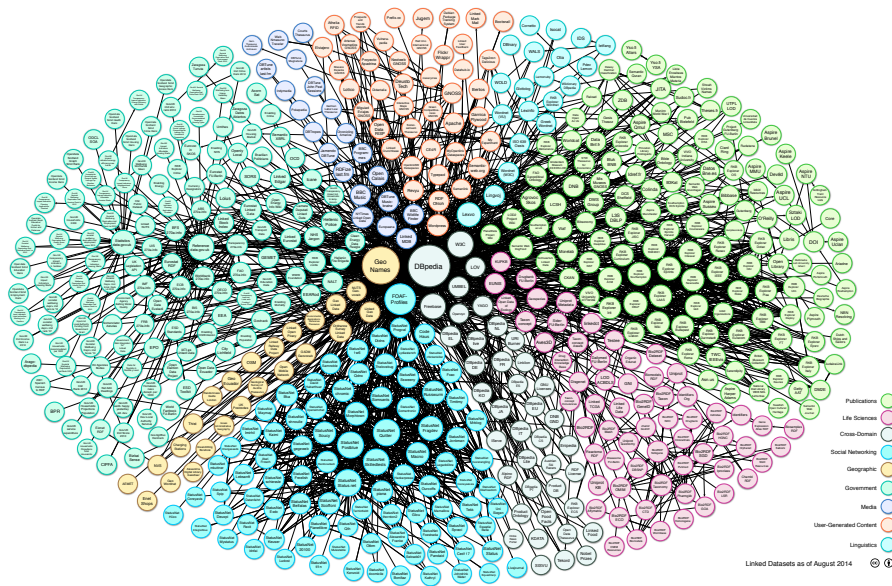


Figure 2: Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. <http://lod-cloud.net/>

define relationships between those concepts, e.g. *person-was-born-in-city*, globally and unambiguously. To ensure that those URIs remain unchanged, Berners-Lee [1998] and Sauermann and Cyganiak [2008] introduced general rules for stable URI generation.

RESOURCE DESCRIPTION FRAMEWORK The Resource Description Framework (RDF) is a collection of standards [Klyne and Carroll, 2004; Hayes and McBride, 2004; Manola and Miller, 2004; Brickley and Guha, 2004; Beckett, 2004; Grant and Beckett, 2004] which were initially released as W3C recommendation in version 1.0 in 2004 and have recently been updated to version 1.1 [Lanthaler et al., 2014]. RDF is thereby a language to express information or assertions about resources (which get identified by URIs) respectively. These assertions, typically called *triple* in the RDF data model, consist of the following three components:

SUBJECT

The subject is the resource over which an assertion is made. Only URIs and anonymous nodes¹ are allowed at this position.

PREDICATE

A predicate is typically understood as an attribute of the resource to describe (a datatype property in OWL), or a binary relation, which links this resource to another one (an object prop-

¹ A anonymous node, also call *blank node*, is only unambiguous in a local context. It is typically used to group assertions.

erty in OWL). At the predicate column of a triple only URIs are valid.

OBJECT

An object stands either for an attribute value or the second argument, another resource, of the binary relation. Valid objects are URIs and blank nodes, but also strings. These strings, also called literals, can be typed or untyped. A typed literal gets assigned a datatype which defines how it is interpreted. Untyped literals do not have this datatype but may be annotated with an optional language definition, which defines the literal's language.

If a set of triples are combined a directed graph emerges in which the edges between subject and object correspond to the triple's predicate. These graphs can be serialized in different formats. Data transfer has typically been carried out in the RDF/XML format [Beckett, 2004, 2014] but since this format is hard to read for human users and may contain a large amount of boilerplate syntax other formats such as *Turtle* [Beckett and Berners-Lee, 2008] and *N-Triples* [Carothers and Seaborne, 2014] were introduced. In this thesis, as long as not stated otherwise, the Turtle syntax is used.

RESOURCE DESCRIPTION FRAMEWORK SCHEMA The Resource Description Framework Schema, short RDFS [Guha and Brickley, 2004], was developed as a semantic extension to RDF and also published as a *Recommendation* in 2004 by the W3C. It has recently been updated to version 1.1 [Guha and Brickley, 2014]. Although it is possible in RDF to link resources to other resources with the help of predicates, it is not possible to make further assertions over those predicates or to relate them to other predicates. In RDFS it is now possible to define the domain and range of a given predicate. Furthermore, predicates can be ordered in hierarchies. Likewise it is possible to structure a class – hence a set of resources with equal properties – in hierarchies. Since RDFS is written in RDF, all valid RDFS documents are also valid RDF documents.

WEB ONTOLOGY LANGUAGE Since RDFS is categorized as a lightweight ontology language [Hitzler et al., 2008] and therewith makes it impossible to model certain assertions, the Web Ontology Language, OWL [Bechhofer et al., 2004], was introduced. OWL builds upon RDFS and was as well publicized by W3C as a *Recommendation* in 2004 and is available in its second version since 2009 [Motik et al., 2008]. With OWL it is now possible among other things to model complex concept definitions and cardinality restrictions. Though these constructs may lead to a decreased runtime or even undecidability. For those reasons, OWL was separated into the following three sub-

Concept	Description
<i>owl:Object-Property</i>	Is a class of predicates which links instances to instances.
<i>owl:Datatype-Property</i>	Is a class of predicates which links instances to typed as well as untyped literals.
<i>owl:sameAs</i>	Is a predicate which links instances to instances and therewith defines that both URIs reference the same <i>thing</i> , i.e. both instances have the same identity.

Table 2: Overview of relevant OWL concepts

languages with different levels of expressiveness and complexity respectively. Table 2 presents OWL concepts which were used in this thesis.

OWL LITE is a proper sublanguage of OWL DL. It is decidable and only uses a part of the OWL vocabulary.

OWL DL is a proper sublanguage OWL Full. It was designed to retain computation completeness and decidability and it includes the complete OWL vocabulary. The decidability in this connection is preserved by the introduction of restrictions.

OWL FULL contains the complete OWL vocabulary and is therewith the most expressive sublanguage, which is the reason why OWL Full is undecidable.

2.1.2 SPARQL

SPARQL, the SPARQL Protocol and Query Language is a collection of W3C Recommendations [Prud'hommeaux and Seaborne, 2008; Clark et al., 2008; Beckett and Broekstra, 2008]. The goal of these recommendations is to provide a consistent query language for local and remote RDF databases, hereafter referred to as *triple stores*. The collection consists of the SPARQL query language itself, the SPARQL protocol to query (remote) triple stores and the SPARQL query results XML format for binding the results of a query. The query language builds upon SQL syntax and is based on *basic graph patterns* which can be combined to complex *group graph patterns*. A basic graph pattern consists thereby of a triple pattern, where subject, predicate and/or object can be replaced by a variable and an optional filter condition. An example SPARQL query is given in Listing 2. This query returns all pairs of organizations (?s, ?o) which are connected through predicates ?property and have at least one English label. SPARQL is heavily used in this thesis to extract training data from DBpedia and other triples stores.

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX dbpo: <http://dbpedia.org/ontology/>
4 SELECT ?s ?sLabel ?prop ?o ?oLabel ?domain ?range
5 FROM <http://dbpedia.org>
6 WHERE {
7   ?s rdf:type dbpo:Organization .
8   ?o rdf:type dbpo:Organization .
9   ?s rdfs:label ?sLabel .
10  ?o rdfs:label ?oLabel .
11  ?s ?property ?o .
12  FILTER (langMatches(lang(?sLabel),'en') && langMatches(lang(?oLabel),'en')) .
13 }

```

Listing 2: Example SPARQL query.

Since the original collection of SPARQL Recommendations did not include any specifications on how to alter an existing RDF knowledge base SPARQL 1.1 [Seaborne and Harris, 2013] was introduced. With SPARQL 1.1 it is furthermore possible to execute a single SPARQL query in a federated way, i.e. one query is sent to multiple (remote) SPARQL endpoints and the results will be merged.

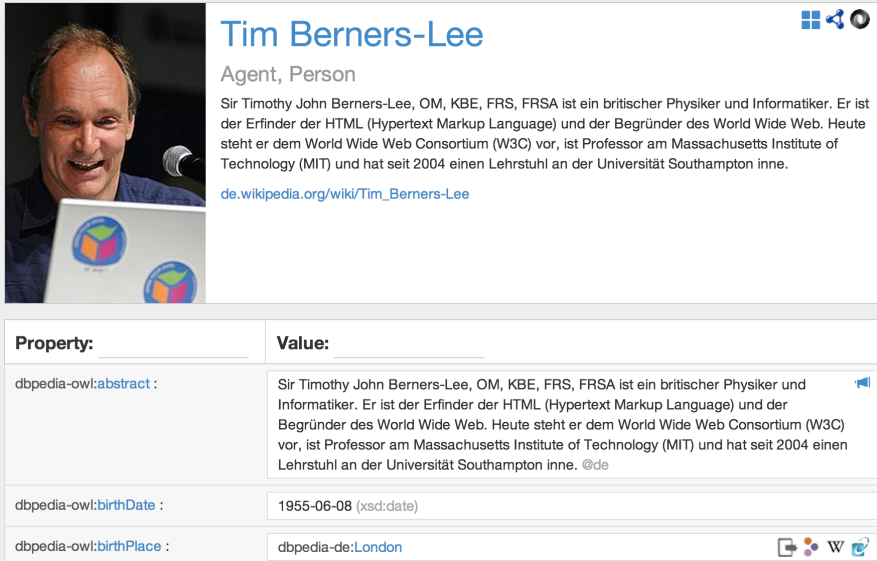
2.1.3 DBpedia, YAGO(2) and Freebase

In recent years, a multitude of large-scale, machine-readable knowledge bases have emerged. Due to its great success and wide coverage in different domains Wikipedia has become the cornerstone to many of these knowledge bases, or to their underlying relation extraction algorithms respectively. These kinds of knowledge bases started with Cyc [Lenat, 1995] and WordNet [Fellbaum, 1998] and have now reached a new level with DBpedia, YAGO and even commercial applications such as Wolfram Alpha or Freebase. The following three paragraphs, depicted in Table 3, present three knowledge bases, which are of key importance to this thesis.

DBPEDIA The goal of the DBpedia project is to create and publish a machine readable version of Wikipedia freely available for everyone. It therefore utilizes a crowd-sourced approach to map the semi-structured information found in Wikipedia infoboxes², authored by millions of editors from the Wikipedia community to a formal ontology. DBpedia, depicted in Figure 3, contains 1.86 billion pieces of information about geographie, people, companies, films, music, genes, drugs, books and scientific publications etc. DBpedia has evolved significantly from its initial publication [Lehmann et al., 2009] to its most recent state [Lehmann et al., 2013]. It now uses a manually crafted ontology with 320 classes and 1650 properties, compared to the initial ontology with 170 classes and 720 properties to semantically annotate the 13.7 million total entities from 111 languages compared to 30 languages and 2.6 million entities in the earlier version. It also includes a live synchronization module which is triggered whenever

² Infoboxes are typically found in the top right corner of an article.

a Wikipedia article changes, providing an always up-to-date knowledge base. Additionally, it contains 27 million links to 30 external data sources, making it the central interlinking hub in the Linked Open Data Cloud (see Figure 2).



Tim Berners-Lee
Agent, Person

Sir Timothy John Berners-Lee, OM, KBE, FRS, FRSA ist ein britischer Physiker und Informatiker. Er ist der Erfinder der HTML (Hypertext Markup Language) und der Begründer des World Wide Web. Heute steht er dem World Wide Web Consortium (W3C) vor, ist Professor am Massachusetts Institute of Technology (MIT) und hat seit 2004 einen Lehrstuhl an der Universität Southampton inne.

de.wikipedia.org/wiki/Tim_Berners-Lee

Property:	Value:
dbpedia-owl:abstract :	Sir Timothy John Berners-Lee, OM, KBE, FRS, FRSA ist ein britischer Physiker und Informatiker. Er ist der Erfinder der HTML (Hypertext Markup Language) und der Begründer des World Wide Web. Heute steht er dem World Wide Web Consortium (W3C) vor, ist Professor am Massachusetts Institute of Technology (MIT) und hat seit 2004 einen Lehrstuhl an der Universität Southampton inne. @de
dbpedia-owl:birthDate :	1955-06-08 (xsd:date)
dbpedia-owl:birthPlace :	dbpedia-de:London

Figure 3: Screenshot of the DBpedia graphical user interface.

YAGO(2) YAGO was originally introduced by Suchanek et al. [2007, 2008]. YAGO aims at extracting knowledge from Wikipedia and unify it with WordNet using a combination of rule-based and heuristic methods. The YAGO ontology browser is depicted in Figure 4. The original version contained about 1 million entities and 5 million facts with a manually evaluated accuracy of 95%. In contrast to DBpedia, YAGO does not use a handcrafted ontology but exploits Wikipedia categories to deliver type information for entities, which results in more specific type information, e.g. *yago:HostCitiesOfTheSummerOlympicGames*. YAGO2, presented by Hoffart et al. [2013], adds a spatial and temporal dimension to entities, facts and events. The authors also introduced a new representational model, coined SPOTL tuples (Subject + Predicate + Object + Time + Location) and further extended these to SPOTLX tuples, where X corresponds to a list of keywords extracted from the facts context. The following is an example of the SPOTL(X) query language used to query this model:

```
?p directed ?m after [1970] matches (+cowboys +mexico)
```

YAGO2 is built automatically from the English Wikipedia articles³, GeoNames and WordNet and currently contains about 124 million facts for 2.6 million entities.

³ Only English articles are considered as entities, but multilingual labels are provided.

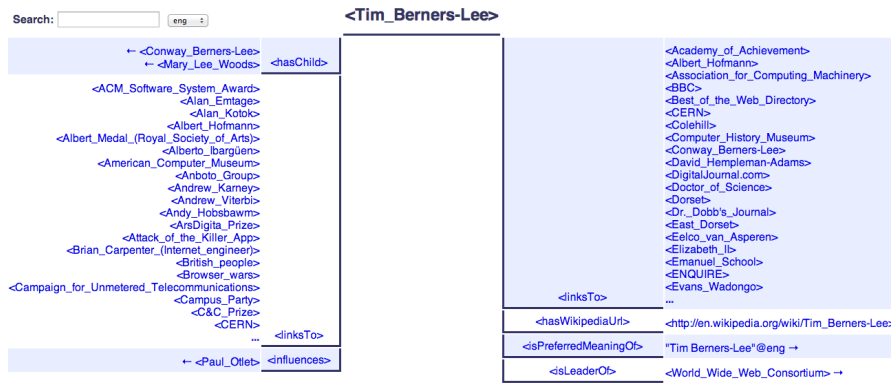


Figure 4: Screenshot of the YAGO(2) ontology browser.

FREEBASE Freebase is a large-scale graph database. It was originally developed by Metaweb Technologies, Inc. and was acquired by Google in 2010⁴. It is “a community-curated database of well-known people, places, and things” (<http://freebase.com>, 2014). It currently contains about 2.64 billion facts for 44.15 million entities, which derive not only from Wikipedia but also from other data sources such as MusicBrainz⁵. In contrast to DBpedia and YAGO(2), where users have to change Wikipedia articles in order to change the respective output, Freebase offers a rich public interface through which users can contribute directly to the underlying knowledge base. Also, Freebase uses n -ary relations, depicted in Figure 5, to store data in their graph database. This allows Freebase, similar to YAGO2 and to a lower extend DBpedia⁶, to scope their facts temporally and spatially. Additionally Freebase provides a JSON based query language called Metaweb Query Language (MQL)⁷. In 2012 Google has incorporated the Freebase dataset⁸, also called *Knowledge Graph*, into their search engine results as shown in Figure 6.

Education /people/person/education						
Institution	Start Date	End Date	Degree	Major/Field Of Study	Specialization	Minor
The Queen's College, Oxford	1973	1976	Bachelor's degree	Physics		
Emanuel School	1969	1973	Secondary education			

Figure 5: Screenshot of the Freebase user interface depicting Tim Berners-Lee’s educational stages in an n -ary relational format.

⁴ <http://googleblog.blogspot.de/2010/07/deeper-understanding-with-metaweb.html>

⁵ <https://musicbrainz.org/>

⁶ RDF reification can be used to model facts about facts, but this entails significantly larger database sizes and query execution times.

⁷ <http://mql.freebaseapps.com/index.html>

⁸ <http://googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html>


[Tim Berners-Lee - Wikipedia, the free encyclopedia](#)
 en.wikipedia.org/wiki/Tim_Berners-Lee
 Sir Timothy John "Tim" Berners-Lee, OM, KBE, FRS, FREng, FRSA, DFBCS (born 8 June 1955), also known as "TimBL," is a British computer scientist, best ...
 Conway Berners-Lee - ENQUIRE - Ferranti Mark 1 - Mary Lee Woods

[Tim Berners-Lee - Wikipedia](#)
 de.wikipedia.org/wiki/Tim_Berners-Lee Translate this page
 Sir Timothy John Berners-Lee, OM, KBE, FRS, FRSA (* 8. Juni 1955 in London) ist ein britischer Physiker und Informatiker. Er ist der Erfinder der HTML ...

[Tim Berners-Lee - World Wide Web Consortium](#)
 www.w3.org/People/Berners-Lee/
 A graduate of Oxford University, Tim Berners-Lee invented the World Wide Web, an internet-based hypermedia initiative for global information sharing while at ...

[Longer Bio for Tim Berners-Lee - World Wide Web Consortium](#)
 www.w3.org/People/Berners-Lee/Longer.html
 Tim Berners-Lee graduated from the Queen's College at Oxford University, England, 1976. Whilst there he built his first computer with a soldering iron, TTL gates ...

[Tim Berners-Lee: The next web | Video on TED.com](#)
 www.ted.com/.../tim_berners_lee_on_the_next_web...
 20 years ago, Tim Berners-Lee invented the World Wide Web. For his next project, he's building a web for ...



Tim Berners-Lee
 Computer Scientist
 Sir Timothy John "Tim" Berners-Lee, OM, KBE, FRS, FREng, FRSA, DFBCS, also known as "TimBL," is a British computer scientist, best known as the inventor of the World Wide Web. Wikipedia
Born: June 8, 1955 (age 58), London, United Kingdom
Nationality: British
Books: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor
Awards: MacArthur Fellowship, Marconi Prize, Charles Stark Draper Prize, Mountbatten Medal, President's Medal
Education: The Queen's College, Oxford (1973–1976), Emanuel School (1969–1973)
Parents: Mary Lee Woods, Conway Berners-Lee

Figure 6: Web-view of the Google Knowledge Graph based on Freebase data.

	DBpedia	YAGO2	Freebase
Classes	320	365,372	25,514
Properties	1,650	104	37,161
Instances	13.7M	2.6M	44.2M
Facts	1.86B	124M	2.6B
Languages	111	en	47
User-editable	No	No	Yes
Releases	13	2	weekly
Storage	RDF	SPOTL(X)	Graph (n-ary)
Query Language	SPARQL	SPOTL(X)	MQL
License	CC BY-SA 3.0	CC BY 3.0	CC BY 2.5

Table 3: Comparison of DBpedia, YAGO2 and Freebase

Most knowledge sources on the Data Web were extracted from structured or semi-structured data. Thus, they encompass solely a small fraction of the information available on the document-oriented Web. In this chapter, we present BOA, an iterative bootstrapping strategy for extracting RDF from unstructured data. The idea behind BOA is to use the Data Web as background knowledge for the extraction of natural language patterns that represent predicates found on the Data Web. These patterns are used to extract instance knowledge from natural language text. This knowledge is finally fed back into the Data Web, therewith closing the loop. The approach followed by BOA is quasi independent of the language in which the corpus is written. To show the language independence, we begin by proposing an approach for English, the most dominant Web language, and extend the BOA approach on the thereby gained knowledge to support multilingual relation extraction. We evaluate our English approach on two data sets using DBpedia as background knowledge. To demonstrate the multilingual approach we apply it to four different corpora and two different languages using DBpedia as background knowledge. Our results show in both evaluation scenarios that we can extract several thousand new facts in one iteration with very high accuracy. Moreover, we provide the first multilingual repository of natural language representations of predicates found on the Data Web.

This chapter is mainly based on Gerber and Ngonga Ngomo [2011, 2012, 2013].

3.1 INTRODUCTION

While the document-oriented Web aimed at providing information targeted towards humans, the Linked Data Web (short: Data Web) aims to provide knowledge in both human- and machine-readable form. Several approaches have been developed to populate the Data Web. Most of these approaches, like DBpedia [Lehmann et al., 2013] or LinkedGeoData [Auer et al., 2009; Stadler et al., 2012], rely on transforming semi-structured and structured data available on the Web into RDF. The results of the utilization of these approaches can be seen in the significant growth of the Linked Data Cloud (see Figure 2) from 12 knowledge bases to 295 knowledge bases and a total of more than 30 billion triples [Auer et al., 2011] in about four years [Heath and Bizer, 2011]. While these approaches provide a viable mean to expose semi-structured and structured data on the Data Web, they suffer of one fatal drawback: They can only be applied to 15-20% [Blumberg and Atre, 2003; Gaag et al., 2009] of the information on the Web, as the rest of the information in the document-oriented Web is only available in unstructured form. Consequently, the data in the Linked Data Web suffers from a lack of coverage and actuality that has been eradicated from the Web by Web 2.0 and crowdsourcing approaches.

In the first part of this chapter, we present an approach that can bootstrap the knowledge available on the Data Web by harvesting triples from unstructured data. Our approach, dubbed BOA¹ (Bootstrapping the Data Web), starts with the triples available on the Data Web. Then, it extracts natural language patterns that express the predicates found in the triples already available on the Data Web. By using a combination of these patterns and Named Entity Recognition, our approach can identify the labels of instances that stand in the relation expressed by any given predicate. The resulting novel instance knowledge can be finally fed back into the Data Web and can be reused for extracting even more patterns and triples as well as correcting existing knowledge. Our approach is completely agnostic of the knowledge base upon which it is deployed. It can thus be used on the whole Data Web. In addition, it can be used to extract natural language representations of predicates from virtually any language following a subject - predicate - object sentence structure if provided with a Named Entity Recognition service.

Our main contributions are:

1. We present an approach for bootstrapping the Data Web. Our approach uses knowledge from the Data Web to extract even more knowledge that can be inserted directly into the Data Web.

¹ <http://boa.aksw.org>

2. We provide a knowledge base of natural language representations of predicates found on the Data Web (especially in DBpedia).
3. We present an evaluation of the quality of the natural language patterns extracted automatically by our approach and show that we can extract knowledge from text with a precision of up to 99%.

In the second part of this chapter, we extend on the above-mentioned BOA framework. As previously mentioned, the BOA approach is quasi language independent. To prove this claim we extended the BOA framework for multilingual support and evaluated its performance. An important argument for this extension is the language distribution of content in the World Wide Web. As can be seen in Figure 7 English is with 55.5% still the most dominant content language on the web. But a historical analysis shows that this value dropped by 2.2% from January 2011 to January 2014. This finding also shows that almost 45% of the Web are written in a non-English language and therewith present an enormous opportunity for multilingual relation extraction.

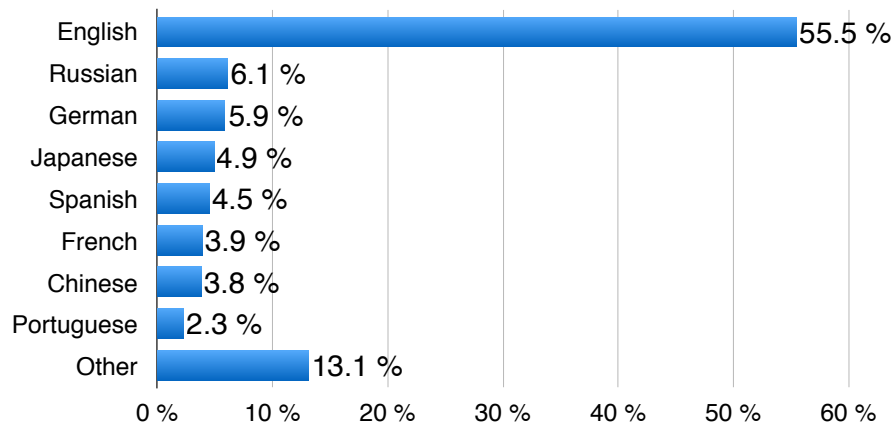


Figure 7: Usage of content languages for webpages. (W3Techs.com, 21 November 2013)

Again, the goal of the BOA framework is to allow extracting structured data as RDF from unstructured data. Unlike many approaches (e.g. Carlson et al. [2010]) which start with their own ontologies and background knowledge as seeds, BOA makes use of the vast amount of multilingual knowledge available on the Data Web to retrieve high-confidence natural language patterns that express the predicates available in the Data Web in multiple languages. The recognition of high-confidence patterns is carried out using supervised machine learning trained on a small set of manually annotated patterns. Based on these patterns, BOA can extract new instance knowledge (i.e. both new entities and relations between these new entities) from the Human Web

with high accuracy. This extension is still completely agnostic of the knowledge base upon which it is deployed and can be used on the whole Data Web. One of the byproducts of the BOA framework is a set of multilingual natural language patterns constituting a valuable resource for tools, that require the conversion of natural language into structured data, (e.g. semantic information retrieval [Shekarpour et al., 2011], question answering frameworks [Unger et al., 2012] and fact validation [Gerber et al., 2015]). A demo of the framework can be found at <http://boa.aksw.org> and the code of the project is hosted at <http://boa.googlecode.com>.

The main contributions to this extension are as follows:

1. We extend the approach implemented by the BOA framework and apply it to corpora written in English and German.
2. We provide a multilingual library of natural language representations of predicates found on the Data Web (especially in DBpedia).
3. We present a set of features that can be used to distinguish high-quality from poor natural language patterns for Data Web predicates.
4. We evaluate our machine-learning approach and the BOA framework on four text datasets against DBpedia and show that we can achieve a high-accuracy extraction in both languages.

The rest of this chapter is structured as follows: In Section 3.2, we give an overview of previous work that is related to our approach. Thereafter, in Section 3.3, we present our bootstrapping framework and several insights that led to the approach currently implemented therein. In Section 3.3.6 we evaluate our approach on two different English data sets and show its robustness and accuracy as well as discuss our results. We present our extended bootstrapping framework and in particular the improvements needed to support multilingual fact extraction in Section 3.4. The evaluation of the multilingual approach on four different datasets in English and German and the discussion of the results is presented in Section 3.4.6. Finally, we discuss our results and conclude.

AUTHOR CONTRIBUTIONS The author of this thesis has been the lead author of the relevant publications [Gerber and Ngonga Ngomo, 2011, 2012, 2013] as well as the main developer of the *English* and *Multilingual* version of BOA. He has developed the corpora extraction module, the background knowledge module, the pattern search, filtering and feature extraction, the RDF generation module and was responsible for the generation and analysis of the evaluation data.

3.2 RELATED WORK

BOA is related to a large number of disciplines due to the different areas of knowledge from which it borrows methods. Like Information Extraction approaches, BOA aims at detecting entities in text. Three main categories of natural language processing (NLP) tools play a central role during the extraction of information from text: Keyphrase Extraction (KE) algorithms aim to detect multi-word units that capture the essence of a document [Matsuo and Ishizuka, 2004; Kim et al., 2010]. Named Entity Recognition (NER) approaches try to discover instances of predefined classes of entities [Ratinov and Roth, 2009; Finkel and Manning, 2010]. The third and most closely related research discipline is Relation Extraction (RE). A detailed overview of all hereafter mentioned RE systems can be found in Table 4. RE approaches are used among other things to discover relations between entities detected using NER [Mintz et al., 2009; Yan et al., 2009] or Part-Of-Speech tagging. One of the first RE systems was introduced by Hearst [1992]. This system focused on extracting un-ary relations, in particular the *hyponym* (*is-a*) relation, from free text. The system produced a set of lexical patterns, such as “such NP as {NP}* {(or | and)} NP”, which can be used to extract facts like “hyponym(‘author’, ‘Herrick’)” or “hyponym(‘author’, ‘Shakespeare’)” from text like “... works by such authors as Herrick, Goldsmith, and Shakespeare”. This work also introduced a procedure for the discovery of new patterns, which greatly influenced future RE systems. DIPRE (Dual Iterative Pattern Relation Extraction), introduced by Brin [1999], utilized this procedure for the extraction of binary *book/author*-relation pairs from HTML documents with the help of lexical patterns. Snowball, presented by Agichtein and Gravano [2000], extended the lexical patterns to also exploit named-entity tags and probability values to extract *company/headquarter*-relation pairs, which resulted in more precise, semantic patterns, e.g.: *ORGANIZATION* (0.42) *s*(0.42) *headquarters*(0.42) *in*(0.12) *LOCATION*. Like DIPRE and the approach presented by Hearst [1992], Snowball also needed manually generated background knowledge for the extraction process. KnowItAll [Etzioni et al., 2004], was the first system trying to use the Web as background knowledge corpora, but focused mainly on un-ary relations and relied on a large number of search engine queries and webpage downloads. DARE (Domain Adaptive Relation Extraction based on Seeds), presented by Xu et al. [2007], is the first approach extracting n-ary relation patterns with the help of dependency trees. While this approach is able to overcome linguistic representations, e.g. subject-verb-object, it is neither suitable for the bootstrapping of the Semantic Web since relations are typically expressed in binary form, nor for web-scale knowledge extraction, because of the high computational costs of dependency parsing. While these approaches are suitable for

the extraction of facts from NL, the use of the Data Web as source for background knowledge for fact extraction is still in its infancy. Suchanek et al. [2009] presented an approach which used YAGO as background knowledge and combined pattern selection, entity disambiguation, and consistency checking in a unified method based on reasoning. The system’s unified approach is elegant and leads to high precision extractions but it needs 16 hours to parse and extract a corpus of 3440 HTML documents, making its web-scale application impossible. Mintz et al. [2009] introduced the term “distant supervision” to describe this paradigm but developed an approach that led to extractors with low precision (approx. 67.6%). The most precise approaches for IE rely on supervised machine learning [Nguyen and Kan, 2007; Zhou and Su, 2002; Curran and Clark, 2003; Finkel et al., 2005]. Thus, they can only extract a small fraction of the knowledge on the Web due to the scarcity of large training datasets. In addition to the work done by the NLP community, several frameworks have been developed with the explicit purpose of bridging the gap between NLP and the Data Web by extracting RDF and RDFa out of NL [Huynh et al., 2005; Adrian et al., 2010]. Services such as Alchemy², OpenCalais³, Extractiv⁴, FOX [Ngonga Ngomo et al., 2011a] and Spotlight [Mendes et al., 2011] allow to extract entities and relations from text. Yet, they do not rely on the Data Web as training data and are thus restricted with respect to the number of relations they can detect. The problem of extracting knowledge from the Web at large scale, which is most closely related to this approach, has been the object of recent research, especially in the projects ReadTheWeb and PROSPERA. The aim of the ReadTheWeb project⁵ [Carlson et al., 2010] is to create the never ending language learner NELL that can read webpages. To achieve this goal, NELL is fed with the ClueWeb09⁶ data set cyclically. The input data for NELL consisted of an initial ontology that contained hundreds of categories and relations, as well as a small number of manually generated instances for each category and relation. In each iteration, NELL uses the available instance knowledge to retrieve new instances of existing categories and relations between known instances by using pattern harvesting. The approach followed by PROSPERA [Nakashole et al., 2011] is similar to that of NELL but relies on the iterative harvesting of n-grams-itemset patterns. These patterns allow to generalize NL patterns found in text without introducing more noise into the patterns during the generalization process. In addition, PROSPERA uses reasoning to discard statements that are logically inconsistent with the available knowledge. In a more recent work, Krause et al.

² <http://www.alchemyapi.com>

³ <http://www.opencalais.org>

⁴ <http://extractiv.com>

⁵ <http://rtw.ml.cmu.edu>

⁶ <http://lemurproject.org/clueweb09>

[2012] presented an approach based on a high-performance dependency parser⁷ to extract n-ary relations in web-scale based on Freebase background knowledge.

BASIC ENGLISH VERSION Our approach goes beyond the state of the art in two key aspects. First, it is the first approach that uses the Data Web as background knowledge for the large-scale extraction of RDF from natural language, therewith making this knowledge effortlessly integrable into the Data Web. ReadTheWeb and PROSPERA rely on their own ontology for this purpose. Thus, their results cannot be linked directly into the Data Web. The approach presented by Mintz et al. [2009] does not generate RDF. In addition, our experiments show that our approach can extract a large number of statements (like PROSPERA and Mintz et al. [2009]) with a high precision (like ReadTheWeb). For example, 98% of the 2657 statements extracted by BOA on organizations in one iteration were not available in the underlying data source. We minimize the effect of semantic drift by adopting a conservative approach with respect to the patterns that are used to generate RDF from text.

MULTILINGUAL VERSION The multilingual version of BOA extends the basic English version in the following ways: First, to the best of our knowledge, it is the first approach to extract multilingual patterns and apply these patterns to extract new knowledge from English and German text. We provide the resulting pattern library to the research community. Since BOA uniquely generates RDF, its output can be used to populate a multilingual knowledge base that can be readily made available for querying via SPARQL, integrating and linking. Second, we show that we can significantly improve recall achieved by BOA through the generation and application of surface forms in the pattern search process. Third, we present a set of features which can be used to distinguish between good and bad patterns. Finally, our experiments show that BOA can extract a large number of statements (like PROSPERA and Mintz et al. [2009]) with a high precision (like ReadTheWeb). For example, 78,944 of the 80,773 statements extracted from the English and 22883 of the 23681 statements extracted from the German Wikipedia were not available in DBpedia.

⁷ <http://mdparser.sb.dfki.de/>

System	Type	Pattern Type	Background Knowledge	Corpora	Reasoning	Relations	Arity	Year	Language
Hearst [Hearst, 1992]	Closed	Lexical	-	American Academic Encyclopedia	✗	hyponym	2	1992	en
DIPRE [Brin, 1999]	Closed	Lexical	Manual	Stanford WebBase	✗	author	2	1998	en
Snowball [Agichtein and Gravano, 2000]	Closed	Lexical, Semantic	Semi-Automatic	North American News Text Corpus	✗	headquarter	2	2000	en
KnowItAll [Etzioni et al., 2004]	Closed	Lexical, Syntactic	-	WWW (search engine query)	✗	hyponym	1	2005	en
DARE [Xu et al., 2007]	Closed	Lexical, Semantic	Manual	MUC 6, Nobel Prize Corpora	✗	award management	n	2007	en

System	Type	Pattern Type	Background Knowledge	Corpora	Reasoning	Relations	Arity	Year	Language
TextRunner [Banko et al., 2007]	Open	Lexical	– (Penn Treebank)	9M Websites	✗	–	2	2007	en
SOFIE [Suchanek et al., 2009]	Closed	Logical Rules	YAGO	Wikipedia	✓	general	2	2009	en
[Mintz et al., 2009]	Closed	Lexical, Syntactic, Semantic	Freebase	Wikipedia	✗	Freebase Top 102	2	2009	en
NELL [Carlson et al., 2010]	Closed	Lexical, Syntactic, Semantic	Manual	ClueWeb09	✓	hyponym (general)	1	2010	en
WOE [Wu and Weld, 2010]	Open	Syntactic, Semantic	Wikipedia	WW, WSI, Wikipedia Sample	✗	–	2	2010	en
ReVerb [Fader et al., 2011]	Open	Lexical, Syntactic	–	ClueWeb09	✗	–	2	2011	en

System	Type	Pattern Type	Background Knowledge	Corpora	Reasoning	Relations	Arity	Year	Language
BOA [Gerber and Ngonga Ngomo, 2011]	Closed	Lexical, Syntactic	DBpedia	Wikipedia, News	✗	general	2	2011	de
PROSPERA [Nakashole et al., 2011]	Closed	Lexical (n-gram itemsets)	NELL	ClueWeb09	✓	sports, academic	2	2011	en
[Krause et al., 2012]	Closed	Semantic	Freebase	WWW (search engine query)	✗	marriage	n	2012	en
RdfLiveNews [Gerber et al., 2013]	Open	Lexical, Syntactic	DBpedia	RSS Feeds	✗	–	2	2013	en

Table 4: Comparison of algorithms related to BOA approach.

3.3 THE BOA FRAMEWORK

In this section we present the BOA framework for extracting natural language representations of relations found on the Data Web. Our workflow is implemented as outlined in Figure 8. We first gather data from the Web by using the corpus extraction module. Alternatively, existing cleaned corpora can be loaded into BOA’s corpus repository. Given a set of predicates whose representations are to be learned, the instance knowledge available on the Data Web is harvested. Then, for each predicate, our pattern extraction module searches for prototypical natural language patterns, that are specific for this predicate. The patterns are subsequently filtered, scored and finally used for extracting RDF statements from natural language text. These statements can then be allocated to the input knowledge base and the process can be restarted. In the following, we present each of the modules of BOA in more detail. In addition, we exemplify their use and their output by using the example of learning patterns from Wikipedia and DBpedia.

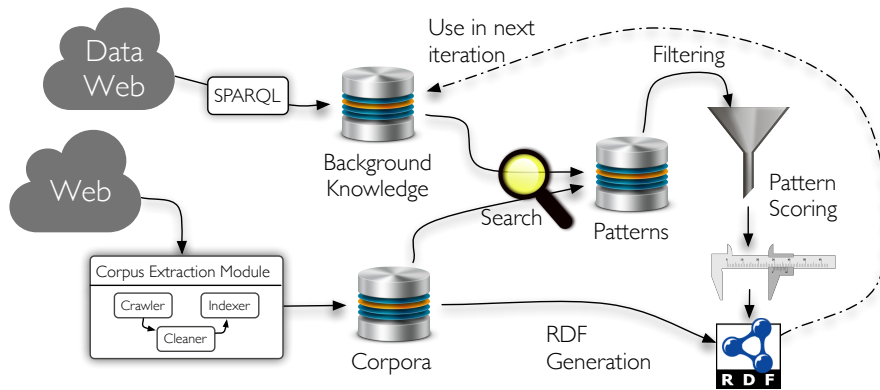


Figure 8: Overview of the BOA approach.

3.3.1 Corpus Extraction

The corpus extraction component of BOA consists of three main modules: A crawler, a cleaner and an indexer. The role of the *crawler module* is to retrieve raw text data from the document-oriented Web. The seed pages for this process are determined by querying the Web with the labels of instances that are linked by the predicates whose natural language representations are to be learned. Once presented with a set of seed pages, our crawler can be configured to follow the links in the seed pages up to a certain depth and until a given corpus size is reached. Note that the crawler gets rid of the markup contained in the webpage text. In addition, it allows to integrate corpus-specific pre-processing components so as to enable it to extract raw text. For example, when extracting a corpus from a Wikipedia dump, we used the

tool WP2TXT presented by Yoichiro [2006] to transform the Wikipedia data from XML to UTF-8 encoded text. The raw text extracted from all pages is merged to a corpus that is sent to the cleaner module.

The *cleaner module* implements the functionality that is necessary to remove noise from the text retrieved by the crawler. It begins by splitting its input into sentences by using the Sentence Boundary Disambiguation provided by the stanford NLP core toolkit⁸. Subsequently, all sentences go through a data cleaning process with 24 UTF-8 compatible filters, introduced by Biemann et al. [2007]. For example, sentences with too many spaces in relation to length, with uncommon symbols like | [] « » and sentences with more than eight capital words in a row are discarded. The cleaned corpus is finally sent to the indexer module.

The *indexer module* allows for the time-efficient search of instance labels and of patterns in our corpus. In BOA, this functionality is implemented by the Lucene indexer⁹, which we configured by using the defaults provided by the engine. The corpus extraction process for Wikipedia led to a corpus of 7.6GB that consisted of 44.7 million sentences.

3.3.2 Knowledge Acquisition

Due to the mere size of the Data Web, extracting natural language representations for all relations found on it would require substantial hardware resources. While our approach is generic enough to be deployed on any knowledge base and on any predicate found on the Data Web, its current implementation demands the input of

- a class C that serves as the `rdfs:domain` or as the `rdfs:range` of the predicates whose representations are to be learned and of
- a knowledge base that serves as background knowledge.

Once C is given, we retrieve all statements that have entities of `rdf:type` C as their subject or objects. By these means, we retrieve all predicates that link such entities to other and ensure that we only retrieve predicates that have been instantiated in the knowledge base of interest. This set of instances is the background knowledge upon which we deploy our pattern extraction approach.

3.3.3 Pattern Search

The pattern search is carried out independently for each predicate. Let $p \in \mathfrak{P}$ be a predicate whose natural language representations are to be detected, where \mathfrak{P} is the set of all predicates. In addition, let

⁸ <http://nlp.stanford.edu/software/tokenizer.shtml>

⁹ <http://lucene.apache.org/java/docs/index.html>

\mathcal{K} be the knowledge base that is used as background knowledge. We use the symbol “ \in ” between triples and knowledge bases to signify that a triple can be found in a knowledge base. The starting point for the pattern search for p is the set of pairs $\mathcal{J}(p) = \{(s, o) : (s \text{ p } o) \in \mathcal{K}\}$ that instantiate p . In the following, we use $\lambda(x)$ to signify the label of any resource x and $\mu(x)$ to signify x 's URI. The pattern search process begins with the even distribution of the set $\mathcal{J}(p)$ across pattern search threads. Each of these threads then retrieves all sentences which contain the pairs of labels $(\lambda(s), \lambda(o))$ assigned to it from the input corpus. An example of such sentences for the DBpedia relation `:subsidiary` is shown in Listing 3. If a thread finds a sentence σ that contains both $\lambda(s)$ and $\lambda(o)$, it deletes all tokens that are not found between $\lambda(s)$ and $\lambda(o)$ in σ . The labels are then replaced with the placeholders D for $\lambda(s)$ and R for $\lambda(o)$. We call the resulting string a *natural language representation* of p and denote it with θ . Each θ extracted is used to create a new instance of a BOA pattern.

Definition 1 (BOA Pattern) A BOA pattern is a pair $\mathcal{P} = (\mu(p), \theta)$, where $\mu(p)$ is p 's URI and θ is a natural language representation of p .

Definition 2 (BOA Pattern Mapping) A BOA pattern mapping is a function \mathcal{M} such that $\mathcal{M}(p) = \mathfrak{S}$, where \mathfrak{S} is the set of natural language representations for p .

```

1 http://dbpedia.org/resource/Google
2   http://dbpedia.org/ontology/subsidiary
3   http://dbpedia.org/resource/YouTube .
4 http://dbpedia.org/resource/Google rdfs:label "Google"@en .
5 http://dbpedia.org/resource/YouTube rdfs:label "Youtube"@en .

```

Listing 3: RDF snippet used for pattern search

For example, consider the RDF snippet from Listing 3 derived from DBpedia. Querying the index of the Wikipedia corpus for sentences which contain both entity labels returns the sentences depicted in Table 5 amongst others. We can replace “Google” with D , because it is the subject of the `:subsidiary` triple, as well as replace “Youtube” with R because it is the object of the same triple. These substitutions lead to the BOA patterns $(\text{:subsidiary}, \text{“}D\text{'s acquisition of }R\text{”})$ and $(\text{:subsidiary}, \text{“}R, \text{a division of }D\text{”})$. For the sake of brevity and in the case of unambiguity, we also call θ “pattern”.

The search for BOA patterns is completed with a large number of duplicates and requires post-processing. In addition to the storage of the patterns $\mathcal{M}(p)$ for each p , the post-processing includes the computation of the number $f(\mathcal{P}, s, o)$ of occurrences of \mathcal{P} for each element (s, o) of $\mathcal{J}(p)$ and the ID of the sentences in which \mathcal{P} was found. Based on this data, we can also compute

- the total number of occurrences of a BOA pattern \mathcal{P} , dubbed $f(\mathcal{P})$;

Sentence with $\lambda(s)$ before $\lambda(o)$ “ Google’s acquisition of Youtube comes as online video is really starting to hit its stride.”	Sentence with $\lambda(o)$ before $\lambda(s)$ “ Youtube, a division of Google, is exploring a new way to get more high-quality clips on its site: financing amateur video creators.”
---------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 5: Example sentences for pattern search.

- the number of sentences that led to θ and that contained $\lambda(s)$ and $\lambda(o)$ with $(s, o) \in \mathcal{J}(p)$, which we denote $l(s, o, \theta, p)$ and
- $\mathcal{J}(p, \theta)$ is the subset of $\mathcal{J}(p)$ which contains only pairs (s, o) that led to θ .

We denote the set of predicates such that the pattern $\theta \in \mathcal{M}(p)$ by $\mathcal{M}(\theta)$. Note that pattern mappings for different predicates can contain the same pattern.

3.3.4 Pattern Scoring

The pattern scoring process is carried out in parallel and consists of two steps: selection and score computation. The aim of the *selection* step is to retrieve patterns that abide by a set of conditions that make them fit for RDF extraction. We begin by dismissing patterns which are too long or short (we only consider patterns between three and ten tokens) and patterns which only consist of stop words (we used a list of 41 stop words including “(”, “,” etc.). In addition we discard all patterns starting with “and” or “, and” since they denote the conjunction of sentences, which leads to ambiguous references when no co-reference analysis or phrase structure analysis is applied. This can be easily seen in the following example: “Davies’ son John played first-class cricket for **Tasmania** and was thrice Mayor of **Hobart**.” This sentence would lead to “D and was thrice Mayor of R” being a pattern for the :capital predicate, which is clearly wrong. Note that we did not apply any co-reference or phrase structure analysis techniques for performance reasons. The last filter we applied removes all patterns which appear less than three times between labels of the same pair of entities. Note that the statistics used for the pattern scoring step encompass all patterns. The scores are yet only computed for those patterns that abide by the restrictions specified above.

The second part is the actual *score calculation*. The score function integrated in BOA relies on the following set of observations:

1. A good pattern θ for p is used across several elements of $\mathcal{J}(p)$. This characteristic is modeled by computing the *support* of the pattern.

2. A good pattern θ for p allows to map D (resp. R) to entities whose `rdf:type` is the `rdfs:domain` (resp. `rdfs:range`) of p . We call this characteristic *typicity*.
3. A good pattern θ is used exclusively to express p , i.e, it occurs in a small number of pattern mappings. We call this last characteristic *specificity*.

We first express these three characteristics of a good pattern formally. Subsequently, we derive our formula for the score of a pattern.

SUPPORT We calculate the support $s(\theta, p)$ of the pattern θ for the predicate p as follows:

$$s(\theta, p) = \log \left(\max_{(s, o) \in \mathcal{J}(p)} l(s, o, \theta, p) \right) \log(|\mathcal{J}(p, \theta)|). \quad (1)$$

Since both components of the support are Poisson-distributed (see Figure 9), we use the logarithm to reduce the boosting of very popular patterns.

TYPICITY A pattern θ is considered to display a high typicality with respect to a predicate p if it connects only entity labels which match the range and domain restrictions of p . Let d resp. r be functions that map each p to the highest super-class (except *owl:Thing*) of its `rdfs:domain` resp. `rdfs:range` in the provided ontology. Furthermore, let $\delta(\theta, \sigma)$ resp. $\rho(\theta, \sigma)$ be functions which map the class of the named entity used to substitute D resp. R in the pattern θ for the given sentence σ . Finally, let the function $\zeta(x, y)$ be a function that returns 1 if $x = y$ and else 0. We define the typicality of θ as

$$t(\theta, p) = \sum_{\sigma \in S} \left(\frac{\zeta(d(p), \delta(\theta, \sigma)) + \zeta(r(p), \rho(\theta, \sigma))}{2|S|} \right) \cdot \log(|S| + 1), \quad (2)$$

where S is the set of sentences used to evaluate the typicality of θ . Note that the first term of the typicality is simply the precision of the pattern. We multiply this factor with the logarithm of $(|S| + 1)$ to prevent overly promoting patterns which have a low recall, i.e. patterns that return only a small number of sentences. Also note, that the detection of $\delta(\theta, \sigma)$ resp. $\rho(\theta, \sigma)$ is a demanding task, which we solved so far by using a trained NER tagger.

SPECIFICITY A pattern θ is considered to be specific when it occurs in a small number of pattern mappings, i.e, when it expresses exclusively p . We adapted the idea of inverse document frequency (*idf*) as known from Information Retrieval to capture this characteristic. The specificity $i(\theta)$ of θ is thus given by the following expression:

$$i(\theta) = \log \left(\frac{|\mathcal{P}|}{|\mathcal{M}(\theta)|} \right), \quad (3)$$

where \mathfrak{P} is the set of all predicates. All three equations can now be combined to the global score $c(\theta, \rho)$ used by BOA as shown in Equation 4:

$$c(\theta, \rho) = s(\theta, \rho)t(\theta, \rho)i(\theta). \quad (4)$$

We define the normed score $c_n(\theta, \rho)$ as the score divided by the local maximum over all patterns belonging to the same pattern mapping to normalize the scores to the interval $[0, 1]$:

$$c_n(\theta, \rho) = \frac{c(\theta)}{\max_{\theta' \in \mathcal{M}(\rho)} c(\theta')}. \quad (5)$$

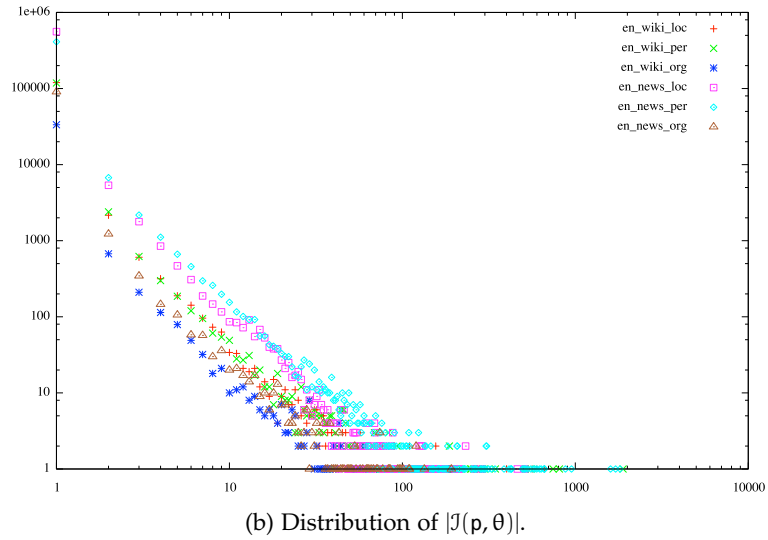
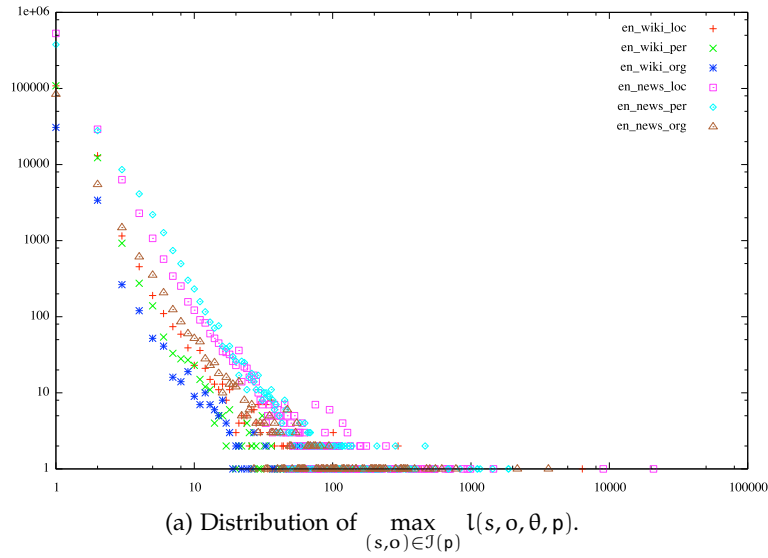


Figure 9: Distribution of parameters used to compute the *support* of patterns in log-log scale. The y-axis shows the number of patterns

3.3.5 RDF Generation

The RDF generation is a very delicate process as each iteration generates the input for the subsequent RDF generation. In previous works, semantic drift has been shown to be one of the key problems of this process [Carlson et al., 2010; Nakashole et al., 2011]. In order to maintain high precision and to avoid semantic drift within the BOA framework, we solely select the top- n patterns θ for each predicate p according to $c_n(\theta, p)$ for generating RDF. In addition, we filter out those patterns θ which display a normed score below a given threshold as well as a $f((\mu(p), \theta))$ below a second threshold. As our evaluation shows, this approach is sufficient to avoid selecting noisy patterns. All patterns which abide by these two conditions are used to retrieve sentences that can be used for RDF generation.

The RDF generation *per se* is carried out as follows: For each pattern θ and each predicate p , we first use the index to retrieve sentences that contain θ stripped from the placeholders “D” and “R”. These sentences are subsequently processed by a NER tool that is able to detect entities that are of the `rdfs:domain` and `rdfs:range` of p . Thereafter, the first named entities on the left and right of θ which abide by the domain and range restrictions of p are selected as labels for the subject and object of p . Each of the extracted labels is then fed into an URI retrieval service that aims to retrieve the entity e in \mathcal{K} whose label is most similar to the label extracted by BOA. If such an e is found, then we use $\lambda(e)$ in \mathcal{K} as URI for the label detected by BOA. Else, we create a new BOA URI.

Once we have computed the URIs, we are finally able to generate RDF triples. The labels retrieved by our URI retrieval approach are attached to the URI by using `rdfs:label`. In addition, note that we are even able to add `rdf:type` statements to our knowledge base by utilizing the domain and range restrictions of p . An excerpt of novel statements (with respect to DBpedia) extracted automatically by BOA using Wikipedia and DBpedia can be found in Listing 4. The results of our extraction can be explored via the dashboard shown in Figure 10.

```

1 http://dbpedia.org/resource/Abdullah_Ahmad_Badawi
2   rdfs:label  "Abdullah Ahmad Badawi"@en ;
3   rdf:type   http://dbpedia.org/ontology/Person .
4
5 http://dbpedia.org/resource/Malaysia
6   rdfs:label  "Malaysia"@en ;
7   rdf:type   http://dbpedia.org/ontology/PopulatedPlace ;
8   http://dbpedia.org/ontology/leaderName http://dbpedia.org/resource/
   Abdullah_Ahmad_Badawi .

```

Listing 4: RDF snippet generated by BOA

3.3.6 Evaluation

Our evaluation was driven by two main questions:

ID	CONF	NATURLANGUAGE REPRESENTATION	OCC
60899	1	?R?, the co-founder of 7D?	47
60387	0.951	?R?, founder of the 7D?	79
59528	0.858	?R?, CEO of 7D?	84
60984	0.822	?R?, the founder of 7D?	118
60071	0.715	?R?, chairman of 7D?	68
61009	0.705	?R?, the founder of the 7D?	34
60869	0.693	?R?, the chairman of the 7D?	32
60670	0.685	?R?, president of the 7D?	59
60878	0.642	?R?, the chief executive of 7D?	20
60664	0.621	?R?, president of 7D?	35
60329	0.619	?R?, founder and chief executive of 7D?	20
60108	0.601	?R?, chief executive of 7D?	72
59766	0.592	?R?, a co-founder of 7D?	53

Figure 10: Screenshot of the BOA frontend

- Q₁: Can we use knowledge found on the Data Web to bootstrap the Data Web, i.e. can we find knowledge not yet available on the Data Web?
- Q₂: Can we retrieve this knowledge with a high precision, i.e. does the score calculation retrieve the right patterns for each predicate?

To answer these questions, we evaluated our approach on two different data sets and used DBpedia as source for background knowledge. Note that we only considered the classes Person, Location and Organisation as seed classes for the extraction due to the restrictions of the NER framework we utilized.

3.3.6.1 Experimental Setup

CORPORA We used two corpora, which differ in topic, size and writing style. The first corpus, dubbed *en-news*, was described by [Biemann et al. \[2007\]](#). It was crawled from news sites in English that were published between the years 2005 and 2010. The corpus contains between 32 million to 50 million unique and cleaned sentences for each year (256.1 million sentences overall). The second corpus, dubbed *en-wiki*, was derived from the English Wikipedia dump of March 2011 without history or discussions entries. The dump was transformed by the WP2TXT tool from XML to UTF-8-encoded text. In contradistinction to *en-news* we did not remove duplicate sentences from *en-wiki* as it did not contain as many duplicates as *en-news*. Overall, the *en-wiki* corpus contains 44.7 million sentences. An overview of the corpora can be found in Table 6.

BACKGROUND KNOWLEDGE We used DBpedia as source for background knowledge. Listing 5 shows an example of the queries used to retrieve properties and instances that are relevant for the classes Organisation, Place and Person from DBpedia (see Figure 11). Overall, the knowledge acquisition process led to 283 different relations

ranging from 1 to 471920 triples, with an average of 4639 triples per relation. Note that the evaluation was carried out independently for each of the six possible combinations of seed classes and corpora.

PARAMETERS We limited the number of sentences returned by Lucene for the pattern search to 25000. We also excluded all patterns $\mathcal{P} = (\mu(p), \theta)$ with $f(\mathcal{P}) < 20$. We used up to 500 sentences to calculate the typicality of the patterns (see Equation 2). For the selection of sentences for the RDF generation, we only used the top-1 and top-2 patterns for each predicate. The evaluation of the accuracy of each pattern was carried out manually by two evaluators on 100 statements that were selected randomly. The inter-annotator agreement was computed by using Cohen’s Kappa. Since the precision of the entity extraction algorithm is out of scope of this evaluation, we considered a triple as correct if the sentence it was generated from, contained the right labels for named entities that matched the `rdf:type` of the domain and range of `p`. Furthermore, sentences which contained references to correct entities (e.g. pronouns) but not the entity labels themselves were considered to be false positives.

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX dbpo: <http://dbpedia.org/ontology/>
4 SELECT ?s ?sLabel ?prop ?o ?oLabel ?domain ?range
5 WHERE {
6   ?s rdf:type dbpo:[Organisation|Person|Place] .
7   ?s rdfs:label ?sLabel .
8   ?o rdfs:label ?oLabel .
9   [?o ?prop ?s|?s ?prop ?o] .
10  FILTER (lang(?sLabel) = en && lang(?oLabel) = en) .
11  ?prop rdfs:range ?range .
12  ?prop rdfs:domain ?domain .
13 }

```

Listing 5: SPARQL query template used for knowledge acquisition.

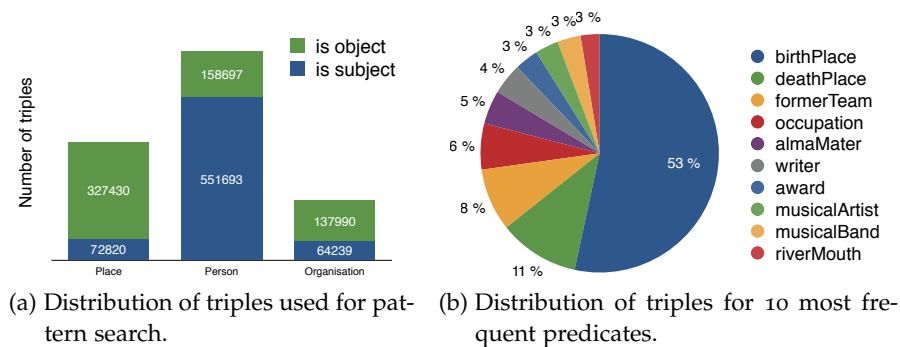


Figure 11: Overview of the knowledge extraction on DBpedia

3.3.6.2 Results and Discussion

The results of our evaluation are shown in Tables 7 and 8. We reached an average inter-annotator agreement of 0.9175. Our approach per-

Name	wiki	news
Language	English	English
Topic	General knowledge	Newspaper articles
Number of lines	44.7	256.1
Number of words	1032.1	5068.7
Number of characters	5689	30289.7
Number of unique words	5.9	26.3

Table 6: Corpora statistics. (All figures in millions.)

formed consistently best on predicates related to organizations and worst on those related to locations. When using the *en-wiki* as corpus, our worst precision was 90.5% on locations when using the top pattern. This value improved to 93% when using the top-2 patterns. Our overall best precision of 99% clearly answers question Q₂ positively. Our good performance on *en-wiki* is due to the encyclopedic character of this dataset. Overall, *en-wiki* contained a (in relation to its size) relatively larger number of sentences that expressed the relations found in DBpedia. Consequently, we could rely on a relatively larger number of good exemplary sentences and compute more accurate scores.

	en-wiki			en-news		
	LOC	PER	ORG	LOC	PER	ORG
Rater 1	88%	97%	99%	61%	73%	91%
Rater 2	93%	97%	99%	62%	74%	91%
Intersection	84%	96%	99%	59%	73%	91%
Average	90.5%	97%	99%	61.5%	73.5%	91%
κ	0.68	0.66	1	0.94	0.97	1

Table 7: Evaluation results for top-1 pattern.

The *en-news* corpus was less reliable in this respect. It contained less information (and consequently more noise) than *en-wiki* and thus led to worse statistics and patterns. This characteristic of the data set was especially noticeable for predicates of the Location class, as locations are very generic and can thus occur in a large number of contexts that do not express the predicate *p* used as input. Consequently, we solely reached precisions between 57% and 61.5% for Location on the *en-news* corpus. On the class Organisation, we still reached precisions of up to 93%.

Note that the patterns extracted from *en-wiki* are by no means bound to *en-wiki* for the extraction of triples. Thus, we can apply the

	en-wiki			en-news		
	LOC	PER	ORG	LOC	PER	ORG
Rater 1	94%	96%	96%	57%	67%	92%
Rater 2	92%	96%	95%	57%	68%	94%
Intersection	94%	96%	95%	57%	67%	92%
Average	93%	96%	95.5%	57%	67.5%	93%
κ	0.9	1	0.88	1	0.98	1

Table 8: Evaluation results for top-2 pattern.

patterns retrieved using *en-wiki* on any corpus. Consequently, the precision scores achieved on *en-wiki* reflect best the overall capabilities of our extraction approach. For example, when applying the top *en-wiki* pattern for predicates from each of the three classes Person, Location and Organisation with the Google index and only considering the top-20 pages returned, we achieved a precision of 95% of the :spouse, 95% of the :capital and 100% on the :subsidiary predicates. Examples of natural language representations extracted by BOA are shown in Table 11.

We measured how much new correct knowledge we were able to generate by counting the number of statements that we generated that could not be found in DBpedia and multiplying it with the precision of our approach. Tables 9 and 10 show that we extracted more than 13,000 new correct facts in one iteration, therewith answering also Q₁ with a clear “yes”. Examples of these new facts are shown in Table 12. As our evaluation shows, our approach is particularly well suited for extracting knowledge on organizations. For example, 98% of the facts considered in our evaluation and extracted by using news data could not be found in DBpedia. When using Wikipedia as text data, 99% of the statements that were evaluated on the organization data were not available in DBpedia. When assuming an even distribution of correct facts and of their inclusion in DBpedia, this implies that 2494 of the 2567 statements on organizations extracted from Wikipedia by using DBpedia are not to be found in DBpedia. Note that one iteration on all predicates linked to organizations on the *en-news* corpus lasts about 15 minutes, therewith showing that our approach is fully suitable for the large-scale extraction of new knowledge from the Web.

A direct comparison of our approach with those presented by Carlson et al. [2010]; Nakashole et al. [2011]; Mintz et al. [2009] cannot be carried out due to the fact that we operate on different background knowledge and on different text corpora. Nevertheless, our evaluation on *en-wiki* shows that although we do not use a reasoner, we extract patterns with a precision equal or superior to those extracted by PROSPERA [Nakashole et al., 2011] with a reasoner. The precision

	LOC	PER	ORG
Extracted Triples in DBpedia	1465	8817	2567
Triples contained in DBpedia	138	183	48
Evaluated triples	100	100	100
Number in DBpedia	8	1	1
Precision (average)	90.5%	97%	99%
New true statements	1200	8375	2494
Pattern mappings	62	72	59
Patterns	1045	612	241

Table 9: Overview of extraction statistics of first iteration for *en-wiki*.

	LOC	PER	ORG
Triples extracted	488	903	916
Triples contained in DBpedia	52	44	7
Evaluated triples	100	100	100
Precision (average)	61.5%	73.5%	91%
New true statements	268	631	827
Pattern mappings	49	70	55
Patterns	3832	7294	1077

Table 10: Overview of extraction statistics of first iteration for *en-news*.

achieved by PROSPERA without a reasoner lies significantly below that of BOA. The same holds for the approach presented by [Mintz et al. \[2009\]](#). In addition, we extract more correct statements in our first iteration than any of the previous approaches even when using a corpus that is more than 650 times smaller than ClueWeb09, there-with hinting towards a higher recall.

en-wiki	en-news
birthPlace (Person/PopulatedPlace)	
<i>D</i> was born in <i>R</i>	<i>D</i> has been named in the <i>R</i>
— (<i>D</i> , the mayor of <i>R</i>)	<i>D</i> , MP for <i>R</i>
foundationPerson (Organisation/Person)	
<i>R</i> , co-founder of <i>D</i>	<i>R</i> , the co-founder of <i>D</i>
<i>R</i> , founder of <i>D</i>	<i>R</i> , founder of the <i>D</i>
subsidiary (Organisation/Organisation)	
<i>R</i> , a subsidiary of <i>D</i>	<i>R</i> , a division of <i>D</i>
— (<i>R</i> , a division of <i>D</i>)	<i>D</i> 's acquisition of <i>R</i>
riverMouth (River/BodyOfWater)	
<i>D</i> , which flows to <i>R</i>	— (<i>D</i> empties into the <i>R</i>)
<i>D</i> , a tributary of the <i>R</i>	— (<i>D</i> , which joins the <i>R</i>)
leaderName (PopulatedPlace/Person)	
<i>D</i> 's Prime Minister <i>R</i>	<i>D</i> 's Prime Minister <i>R</i>
<i>R</i> , the Prime Minister of <i>D</i>	<i>D</i> for talks with President <i>R</i>
capital (PopulatedPlace/City)	
<i>R</i> , the capital of <i>D</i>	<i>R</i> , the capital of <i>D</i>
<i>R</i> , capital of <i>D</i>	<i>R</i> , capital of <i>D</i>

Table 11: Top-2 natural language representations for six most used relations in evaluation. “—” means that no natural language representation was found, patterns in brackets are next in line but were not used for the evaluation because they did not fulfill the threshold requirements.

wiki-loc		
Westlake High School	tenant	Chaparral Stadium
Boston College	tenant	Higgins Hall
Konzerthaus Berlin	architect	Karl Friedrich Schinkel
Villa Foscari	architect	Andrea Palladio

wiki-per		
Elvin Jones	birthPlace	Pontiac, Michigan
Ernest Reyer	birthPlace	Marseilles
Henri Curiel	deathPlace	Paris
Carrero Blanco	deathPlace	Madrid

wiki-org		
Time Warner	subsidiary	DC Comics
Interscope Records	subsidiary	Star Trak Entertainment
Heavy Brigade	notableCommander	James Yorke Scarlett
Federal Department of the West	notableCommander	John C. Fremont

news-loc		
Badakhshan	capital	Faizabad
Quetta	capital	Baluchistan
Bulgaria	leaderName	Boyko Borisov
Japan	leaderName	Taro Aso

news-per		
Leyla Rodriguez Stahl	spouse	Abel Pacheco
Sehba Musharraf	spouse	Pervez Musharraf
Kelly Osbourne	father	Ozzy Osbourne
Svetlana Alliluyeva	father	Josef Stalin

news-org		
College of Cardinals	dean	Cardinal Joseph Ratzinger
Yale University School of Architecture	dean	Robert A.M. Stern
Aldi	foundationPerson	Theo Albrecht
World Wrestling Entertainment	foundationPerson	Vince McMahon

Table 12: Triples extracted from evaluation data set not present in DBpedia.

3.4 MULTILINGUAL EXTENSION OF BOA

In this section, we present the multilingual extension to the BOA framework. We begin by giving an overview of the architecture it implements. Thereafter, we give a deeper presentation of its core components. We begin by explicating the pattern extraction process. Then, we focus especially on the features we extract while searching for adequate patterns. We present our use of neural networks for learning a score function. Finally we show how the scored patterns are used to generate RDF.

3.4.1 Overview

The idea behind the BOA architecture was to provide an architecture that allows extracting structured data from the Human Web iteratively. An overview of the workflow implemented by BOA is given in Figure 12. The input for the BOA framework consists of a set of knowledge bases, a text corpus (mostly extracted from the Web) and (optionally) a Wikipedia dump¹⁰. When provided by a Wikipedia dump, the framework begins by generating surface forms for all entities in the source knowledge base. The surface forms used by BOA are generated by using an extension of the method proposed by Mendes et al. [2011]. For each predicate p found in the input knowledge sources, BOA carries out a sentence-level statistical analysis of the co-occurrence of pairs of labels of resources that are linked via p . Instead of using a hard-coded evaluation function like in previous work, BOA then uses a supervised machine-learning approach to compute the score of patterns for each combination of corpus and knowledge bases. In a final step, our framework uses the best-scoring patterns for each relation to generate RDF data. This data and the already available background knowledge can now be used for a further iteration of the approach. In the following, we describe the core steps of BOA in more detail. Throughout this description, we will use the example of generating new knowledge for the `dbpedia:architect` relation.

3.4.2 Pattern Extraction

Let \mathcal{K} be the knowledge base that is used as background knowledge. The first and optional step of the pattern extraction is the computation of surface forms \mathcal{S}_r for the subject and objects of a relation p for which patterns are to be extracted. To extract surface forms for resources $r \in \mathcal{K}$, we use Wikipedia’s redirect and disambiguation pages as described by Mendes et al. [2011]. The main drawback of this approach is that it is not tailored towards the extraction of datatype properties. Consequently, if the object of a relation is a datatype (i.e.

See Table 13 for a statistical overview of the surface forms for resources found in DBpedia.

¹⁰ <http://wikipedia.c3sl.ufpr.br/>

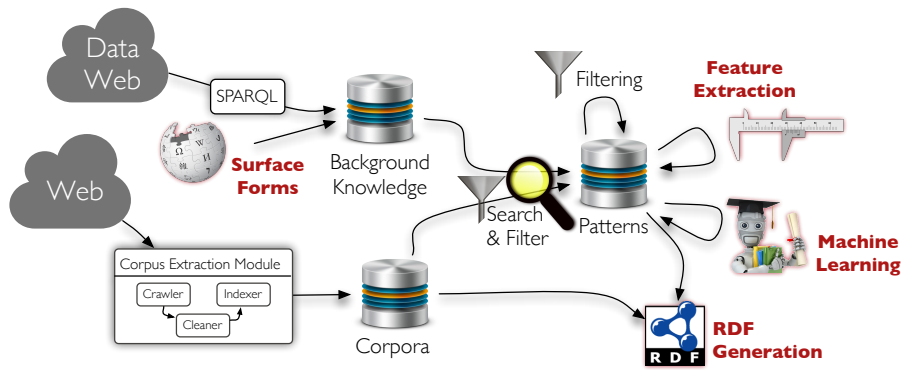


Figure 12: Overview of the BOA approach. Parts that changed from the previous version are marked red.

	German	English
Number of URIs	1,280,859	3,496,082
Number of all surface forms	2,124,084	8,252,275
Maximum of surface forms per resource	132	981
Average of surface forms per resource	1.658	2.360

Table 13: Statistical overview of German and English surface forms.

not a resource), we use data generation modules that allow parsing the datatype at hand and generating possible surface forms for it. For example, when provided with the date “11/12/2012”, BOA generates strings such as “11th Dec 2012”, “11th December 2012” and “Dec 11, 2012”. By using surface forms, we achieve a significantly higher recall than without as shown in Figure 13. It is important to note that the use of Wikipedia dumps does not limit our framework as they exist in more than 90 languages. In addition, BOA can also run without being given surface forms.

The pattern search is carried out independently for each predicate. Let $p \in \mathfrak{P}$ be a predicate whose natural language representations are to be detected, where \mathfrak{P} is the set of all predicates. We use the symbol “ \in ” between triples and knowledge bases to signify that a triple can be found in a knowledge base. The starting point for the pattern search for p is the set of pairs $\mathcal{J}(p) = \{(s, o) : (s \text{ } p \text{ } o) \in \mathcal{K}\}$ that instantiate p . In the following, we use $\lambda(x)$ to signify the set of labels of any resource x and $\mu(x)$ to signify x ’s URI. The pattern search process begins with the even distribution of the set $\mathcal{J}(p)$ across pattern search threads. Each of these threads then retrieves all sentences which contain both labels of all combination of $(\lambda(s), \lambda(o))$ from the input corpus. If a sentence σ containing both labels $l_s \in \lambda(s)$ and $l_o \in \lambda(o)$ is found, it deletes all tokens that are not found between

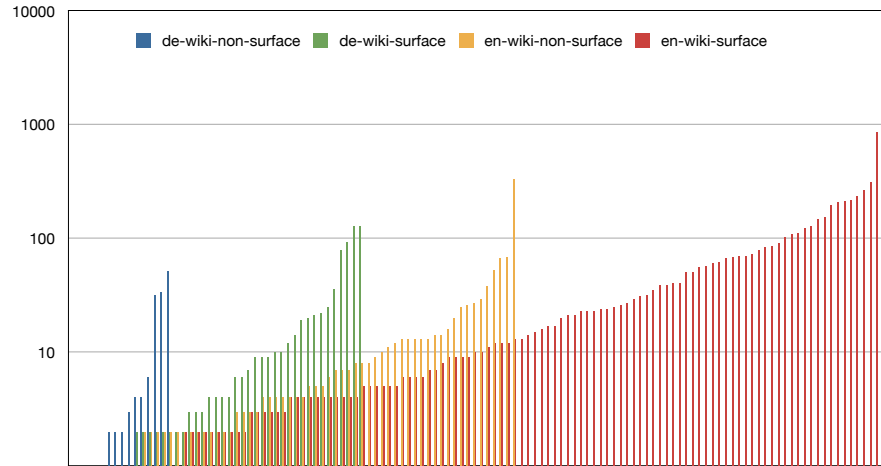


Figure 13: Distribution of patterns per pattern mapping in logarithmic scale. Each vertical line represents one pattern mapping.

l_s and l_o in σ . The labels are then replaced with the placeholders D for l_s and R for l_o . We call the resulting string a *natural language representation* (NLR) of p and denote it with θ . Each θ extracted is used to create a new instance of a BOA pattern.

Definition 3 (BOA Pattern) A BOA pattern is a triple $\mathcal{P} = (\mu(p), \theta, \text{lang})$, where $\mu(p)$ is p 's URI, θ is a natural language representation of p and lang is the language of the NLR.

Definition 4 (BOA Pattern Mapping) A BOA pattern mapping is a function \mathcal{M} such that $\mathcal{M}(p) = \mathfrak{S}$, where \mathfrak{S} is the set of natural language representations for p .

```

1 dbr:Empire_State_Building dbo:architect dbr:Shreve,_Lamb_and_Harmon
2
3 dbr:Empire_State_Building rdfs:label "Empire State Building"@en
4 dbr:Shreve,_Lamb_and_Harmon rdfs:label "Shreve, Lamb and Harmon"@en
5
6 dbr:Empire_State_Building rdfs:label "Empire State Building"@de
7 dbr:Shreve,_Lamb_and_Harmon rdfs:label "Shreve, Lamb und Harmon"@de

```

Listing 6: RDF snippet used for pattern search

For example, consider the RDF snippet from Listing 6 derived from DBpedia. Querying the index of an underlying corpus for sentences which contain both entity labels in the same language returns the sentences depicted in Table 14 amongst others. We can replace “Empire State Building” with D , because it is a label of the subject of the `:architect` triple, as well as replace “Shreve, Lamb and Harmon” and “William F. Lamb” (a surface form of l_r) with R because it is one label of the object of the same triple. The same holds true for the German example. These substitutions lead to the BOA patterns $(:\text{architect}, “D \textit{ was designed by } R”, \text{en})$, $(:\text{architect}, “R \textit{ also designed}$

the D", en), (:architect, "D galt als Hauptarchitekt des R", de) and (:architect, "R , designed und errichtet von D", de). For the sake of brevity and in the case of unambiguity, we also call θ "pattern". Pat-

Sentence with $\lambda(s)$ before $\lambda(o)$	Sentence with $\lambda(o)$ before $\lambda(s)$
"... Shreve, Lamb and Harmon also designed the Empire State Building ."	"The Empire State Building was designed by William F. Lamb ..."
"... William F. Lamb galt als Hauptarchitekt des Empire State Building ."	"... Empire State Building , designed und errichtet von Shreve, Lamb und Harmon ..."

Table 14: Example sentences for pattern search.

terns are only considered for storage and further computation if they withstand a first filtering process. For example, they must contain more than one non-stop-word, have a token count between certain thresholds and may not begin with "and" or ", and". As in the previous, English only, version of BOA we compute the number $f(\mathcal{P}, s, o)$ of occurrences of \mathcal{P} for each element (s, o) of $\mathcal{J}(p)$ and the ID of the sentences in which \mathcal{P} was found. Based on this data, we can, as previously shown (see Section 3.3.3) compute the total number of occurrences of a BOA pattern, the number of sentences that led to θ and $\mathcal{J}(p, \theta)$. Again, we apply a second filtering process, where patterns which do not abide a threshold for $|\mathcal{J}(p, \theta)|$, $\max(l(s, o, \theta, p))$ and $f(\mathcal{P})$ are removed. Two patterns $(\mu(x), \theta_x)$ and $(\mu(y), \theta_y)$ are considered to be distinct if $\theta_x \neq \theta_y$. Note that pattern mappings for different predicates can contain the same pattern.

3.4.3 Feature Extraction

The feature extraction is applied on all patterns which overcome both filtering processes. The pattern mappings are evenly distributed to feature extraction threads. Each thread then computes the features for all patterns of the given pattern mappings. Note that although BOA is designed to work independently from the language of the underlying corpus, it can be tailored towards a given language. For example the ReVerb and IICM feature exploit knowledge that is specific for the English language.

SUPPORT The support features $s_1(\theta, p)$ and $s_2(\theta, p)$ of the pattern θ for the predicate p capture how often a pattern occurs between the elements of $\mathcal{J}(p)$. Especially $s_1(\theta, p)$ stands for the number of distinct pairs the pattern has been learned from, while $s_2(\theta, p)$ is the maximum number of occurrences of the same pattern

between the elements of a single element of $\mathcal{J}(p)$. The support features $s_1(\theta, p)$ and $s_2(\theta, p)$ are formally defined as follows:

$$s_1(\theta, p) = \log(|\mathcal{J}(p, \theta)|) \quad (6)$$

$$s_2(\theta, p) = \log \left(\max_{(s, o) \in \mathcal{J}(p)} l(s, o, \theta, p) \right). \quad (7)$$

We use the logarithm to reduce the boosting of very popular patterns.

SPECIFICITY The specificity of a pattern p is equal to the definition of the specificity given in Equation 3.

TYPICITY We define the typicality features of θ similar to the typicality feature from Equation 2. Instead of calculating a combined score for both, through p , connected entities, we split them into 3 individual features:

$$t_1(\theta, p) = \sum_{\sigma \in S} \frac{\delta(d(p), \text{domain}(\theta, \sigma))}{|S|} \quad (8)$$

$$t_2(\theta, p) = \sum_{\sigma \in S} \frac{\delta(r(p), \text{range}(\theta, \sigma))}{|S|} \quad (9)$$

$$t_3(\theta, p) = \log(|S| + 1) \quad (10)$$

The detection of $\text{domain}(\theta, \sigma)$ resp. $\text{range}(\theta, \sigma)$ is, as in the previous version, carried out by a trained NER tagger.

IICM The **Intrinsic Information Content Metric** (IICM) captures the semantic relatedness between a pattern’s NLR and the property it expresses. This similarity measure has been introduced by [Seco et al. \[2004\]](#) and is based in the Jiang-Conrath similarity measure [[Jiang and Conrath, 1997](#)]. The advantage of this measure is that it is purely based on the hierarchical structure of words inside an ontology like WordNet [[Fellbaum, 1998](#); [Miller, 1995](#)], thus avoiding heavyweight corpus statistics. We apply this measure to each BOA pattern mapping independently. First we retrieve all synsets for each token of the pattern mappings associated *rdfs:label* from WordNet. For the pattern mapping $\mathcal{M}(\text{dbpedia-owl:architect})$ this is “architect” and “designer”. If no such synsets are found we use the tokens of the *rdfs:label* of $\mathcal{M}(p)$. We then apply the IICM measure pairwise to these tokens and the tokens derived from one $\mathcal{M}(p)$ assigned pattern’s NLR. The IICM score for one pattern is then the maximum value of all pairs’ similarity scores.

REVERB ReVerb has been introduced by [Fader et al. \[2011\]](#) and distinguishes good from bad relation phrases by two simple constraints. The first constraint is of syntactical nature and is a part-of-speed-based regular expression, where a token sequence is

considered to be a good instance of a relation if it contains a verb. The second constraint is based on the assumption that a valid relation phrase should take many distinct arguments in a large corpus, therewith avoiding the selection of overly specific relation phrases. Since the input of ReVerb is a POS-tagged sentence, but a pattern is only a substring of a sentence, we use all sentences we found the pattern in (see Section 3.4.2) as ReVerbs input. For all of ReVerb’s extracted relations of a particular sentence we check if it matches the pattern in question and use ReVerb’s trained logistic regression classifier to assign a confidence score to this extraction. Note that BOA focuses on the relation between two given resources and discards all other extractions, since those are not mappable to the background knowledge. Finally, we calculate a pattern’s ReVerb feature as the average of all scored extractions.

TF-IDF The Tf-Idf features are an adaption of the tf-idf score used in information retrieval and text mining. The intention behind this feature, as in information retrieval, is to distinguish relevant from irrelevant patterns for a given pattern mapping $\mathcal{M}(p)$. In the BOA case a document is considered to be all tokens of all patterns (without stop-words and the placeholders “D” and “R”) of one pattern mapping. In other words, the total number of documents is equal to the number of pattern mappings with patterns. We then calculate the features $\text{idf}(p)$ and $\text{tf}(p)$ for each token of the patterns NLR as follows:

$$\text{idf}(p) = \sum_{t \in \mathcal{T}(p)} \log\left(\frac{|\mathcal{M}(p)|}{\text{df}(t) + 1}\right) + 1 \quad (11)$$

$$\text{tf}(p) = \sum_{t \in \mathcal{T}(p)} \sqrt{f(t)} \quad (12)$$

Where $\text{df}(t)$ is the document frequency of t , $f(t)$ the term frequency of t and $\mathcal{T}(p)$ the set of tokens for a pattern p .

3.4.4 Scoring Approach

Given the number of features that characterize the input data, devising a simple scoring function transforms into a very demanding task. In this work, we address the problem of computing a score for each BOA pattern by using feedforward neural networks. The input layer of our network consists of as many neurons as features for patterns while the output neuron consists of exactly one neuron whose activation was used as score. We employed the sigmoid function as transfer function. For each data set, we trained the neural network by using manually annotated patterns (200 in our experiments). The patterns were extracted from the set of all patterns generated by BOA by first

randomly sampling the same number of patterns for each predicate (seven in our experiments) and then selecting a subset of these patterns for annotation.

3.4.5 RDF Generation

The generation of RDF out of the knowledge acquired by BOA is the final step of the extraction process. When using BOA iteratively, the output of each RDF generation would provide parts of the input for the subsequent extraction process. In previous work, semantic drift has been shown to be one of the key problems of such iterative approaches [Carlson et al., 2010; Nakashole et al., 2011]. In order to maintain a high precision and to avoid semantic drift within the BOA framework, we solely select the top- n percent of all scored patterns for generating RDF. As our evaluation shows, this approach is sufficient to avoid selecting noisy patterns. All patterns which abide by these two conditions are used to retrieve sentences that can be used for RDF generation.

The RDF generation *per se* is carried out as follows: For each pattern θ and each predicate p , we first use the index to retrieve sentences that contain θ stripped from the placeholders “D” and “R”. These sentences are subsequently processed by a NER tool that is able to detect entities that are of the `rdfs:domain` and `rdfs:range` of p . Thereafter, the first named entities within a limited distance on the left and right of θ which abide by the domain and range restrictions of p are selected as labels for subject and object of p . Each of the extracted labels is then fed into the URI retrieval and disambiguation service implemented by the FOX framework¹¹. If this service returns a URI, then we use it for the label detected by BOA. Else, we create a new BOA URI.

Once we have computed URIs, we are able to generate RDF triples. The labels found in the text corpus are attached to the URI by using `rdfs:label`. In addition, note that we are even able to add `rdf:type` statements to our knowledge base by utilizing the domain and range restrictions of p . Note as well that it is possible, even desirable that one triple is found by multiple patterns. We use this information to calculate a confidence score $s(t)$ for each triple t that we extract:

$$s(t) = \frac{1}{1 + e^{-\left[\sum_{i=1}^n s(p_i(t))\right]^{n+1}}}$$

where $\sum s(p_i(t))$ is the sum of the score given by the neural network of all patterns that found the triple t and n the number of all patterns. This function is derived from the sigmoid function which outputs values ranging from 0 to 1 inclusively. We modified the function to boost

¹¹ A demo of the framework is available at <http://fox.aks.w.org>.

patterns which are learned by more than one pattern. The triple score is the sum of the patterns' scores which found the triple multiplied by the number of triples. With the help of this score, we can avoid semantic drift in further iterations by solely selecting the top n percent of the triples to use as background knowledge in the next iteration. By applying our approach, we were able to extract triples such as `dbr:Washington_Monument dbo:architect dbr:Robert_Mills`, which is not in DBpedia but explicitly stated in Wikipedia. The results of our extraction can be explored via the dashboard shown in Figure 14.

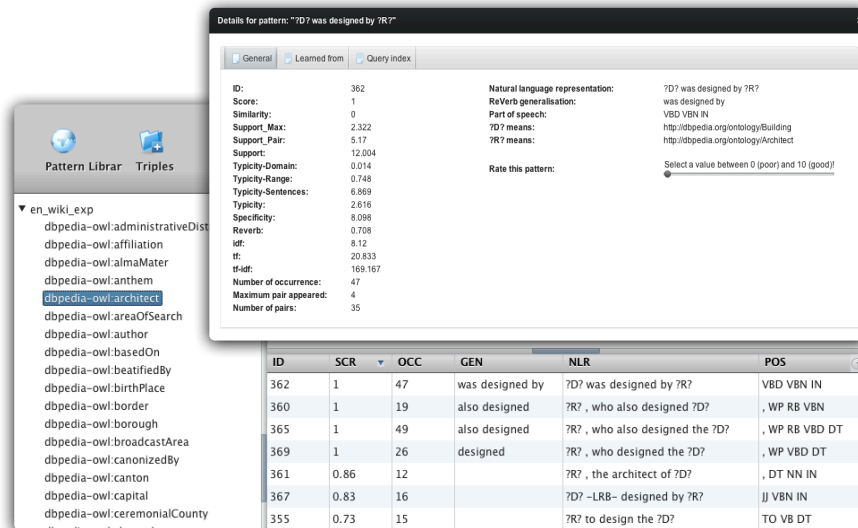


Figure 14: Screenshot of the BOA pattern library Web interface.

3.4.6 Evaluation

The aim of BOA's evaluation was three-fold. First, we aimed at testing whether BOA can be really applied to different languages. To achieve this goal, we applied BOA to German and English. Our second goal was to determine the accuracy of BOA's extraction. For this purpose, we sampled 100 triples from the data extracted by BOA from each corpus and had two annotators measuring the precision of these samples manually. Finally, we wanted to compute the amount of (new) knowledge that can be extracted by BOA. For this purpose, we compute the number of new triples that we were able to extract.

3.4.6.1 Experimental Setup

We excluded temporal properties from the evaluation as BOA does not yet distinguish between different time expressions and conjugations. We evaluated our approach on 4 corpora written in German and English. The first two corpora, en-wiki resp. de-wiki, were ex-

tracted from the English resp. German Wikipedia dumps and contained 58 million resp. 24.6 million sentences. The other two corpora (en-news resp. de-news) were extracted from English resp. German news corpora and were significantly larger, containing, 241.3M resp. 112.8M sentences.

3.4.6.2 Score Function

We began the evaluation by annotating 200 patterns per corpus by hand. Each training data set was annotated independently by the authors, who agreed on the annotations in approximately 90% of the cases. The annotations upon which the authors disagreed were resolved by both authors. High-quality patterns were assigned a score of 1, else they were assigned a 0. We then trained four different neural networks (one for each dataset) to distinguish between the high-precision and poor patterns. In our experiments, we varied the size of the hidden layer between one and three times the size of the input layer. In addition, we varied the error rate to which they were trained. The maximal number of training epochs was set to 10000. The accuracy of the networks was measured by using a 10-fold cross-validation. Patterns scored above 0.5 were considered to be good patterns, while all others were considered to be poor. The best neural network was set to be the smallest network that reaches the maximal accuracy. The resulting learning curves are shown in Figure 15. It is easy to see that networks trained to achieve an error rate of maximally 5% performed best in our experiments.

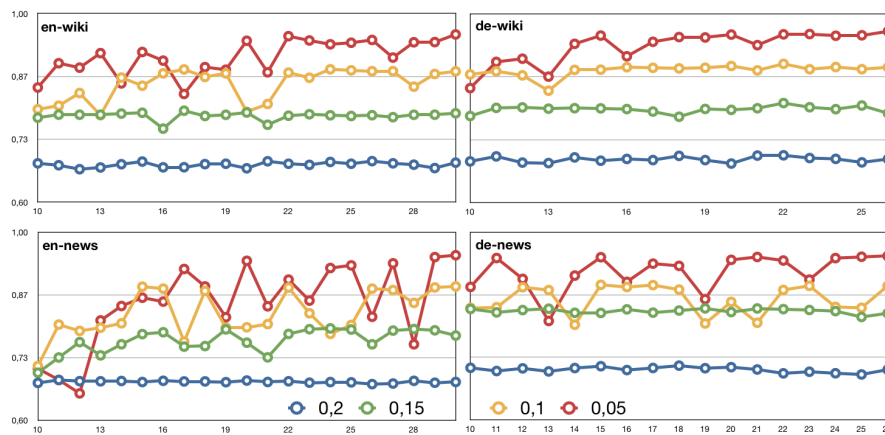


Figure 15: Learning curves of BOA's neural networks. The x-axis shows the number of input neurons and the y-axis shows the achieved accuracy.

3.4.6.3 Multilinguality

Enabling BOA to process languages other than English requires solely the alteration of the NER tools and POS parsers. As the results on German show, languages with a wide range of morpho-syntactical variations demand the analysis of considerably larger corpora to enable the detection of meaningful patterns. For example, while we trained the neural network by using the same number of patterns, we were not able to detect any triples with a score above 0.5 when using the German Wikipedia and DBpedia. Yet, when using a larger German news corpus data set, we were able to detect new patterns with an acceptable precision (see subsequent section).

3.4.6.4 Accuracy

We measured the precision of the extraction carried out by BOA as well as the number of new triples that we were able to extract in one iteration. We achieve a precision superior to 90% overall on the English data sets. This value is comparable to that achieved by the previous versions of BOA [Gerber and Ngonga Ngomo, 2011]. Yet, the addition of surface forms for the extraction yields the advantage of achieving a considerably higher recall both with respect to the number of patterns extracted as well as with respect to the total number of triples extracted. For example, when using the English Wikipedia, we can extract approximately more than twice the amount of triples. The same holds for the number of patterns and pattern mappings as shown in Figure 13.

	en-wiki	de-wiki	en-news	de-news
Nr. of pattern mappings	125	44	66	19
Number of patterns	9551	586	7366	109
Number of new triples	78944	2283	10138	883
Number of known triples	1829	798	655	42
Number of found triples	80773	3081	10793	925
Precision Top-100	92%	70%	91%	74%

Table 15: Results of the first iteration of the BOA framework.

An excerpt of the new knowledge extracted by BOA is shown in Listing 7. Note that the triple Iomega subsidiary ExcelStor_Technology is wrong. Although Iomega planned to buy ExcelStor, the deal was never concluded. Our approach finds the right patterns in the sentences describing the temptative deal and thus extract this triple.

¹ Chinese spokenIn Malaysia .
² Chinese spokenIn China .
³ Italian spokenIn Italy .


```

4 | Greek spokenIn United_States .
5 |
6 | ESV_Blau-Rot_Bonn ground Bonn
7 | TG_Würzburg ground Würzburg
8 |
9 | Weitnau administrativeDistrict boa:Oberallgäu .
10 | Memmingerberg administrativeDistrict boa:Unterallgäu .
11 | Borsdorf administrativeDistrict Leipzig .
12 | Wirsberg administrativeDistrict Kulmbach .
13 |
14 | IBM subsidiary boa:Softek_Storage_Solutions .
15 | Cerberus_Capital_Management subsidiary boa:Chrysler_Holdings_LLC .
16 | Kaspersky_Lab subsidiary boa:Spamtest_Project .
17 | Intel_Corporation subsidiary McAfee .
18 | Iomega subsidiary ExcelStor_Technology
19 |
20 | Pierre-Joseph-Olivier_Chauveau party Conservative_Party_of_Quebec .
21 | Robert_Stanfield party Progressive_Conservative_Party .
22 | Adélarde_Godbout party Quebec_Liberal_Party .
23 |
24 | American_Airlines hubAirport Logan_International_Airport .
25 | Frontier_Airlines hubAirport Akron-Canton_Regional_Airport .
26 | EL_Al hubAirport Los_Angeles_International_Airport
27 |
28 | Neil_Warnock managerClub Sheffield_United_F.C. .
29 | boa:Akira_Ogi managerClub Orix_Buffaloes .
30 | Richard_Money managerClub Walsall .
31 |
32 | Walter_Albrecht_Becker birthPlace Hamburg .
33 | Sophie_Ludovike_Reichenbach birthPlace Ludwigsburg .
34 | Hans_Dreier birthPlace Bremen .

```

Listing 7: RDF extracted by BOA. If not stated otherwise, all instances and properties use the DBpedia namespace.

3.5 CONCLUSION

In this chapter, we presented BOA, an approach for bootstrapping the Data Web. Our approach is based on using instance data for predicates found on the Data Web to retrieve natural language representations for these predicates. Based on these representations, we extract sentences from natural language text that contain both named entities and patterns. These sentences are then used to extract (in particular novel) knowledge from the document-oriented Web and to integrate this knowledge into the Data Web. In Section 3.3 we presented the first implementation of the proposed approach for the English language. Our evaluation shows that when combining knowledge from DBpedia with text from Wikipedia, we achieve precision scores beyond 90% and retrieve more than 12,000 facts that (13,000 with *en-news*) were not previously found in DBpedia within one iteration. In Section 3.4 of this chapter, we presented the multilingual extension of BOA. We gave a detailed description of the improved components of the framework and applied it to English and German corpora. We showed that in all cases, we extract large amounts of RDF triples from the data at hand. Our extraction strategy was to only integrate RDF triples that were generated by at least two patterns, which enabled us to achieve a high precision on all data sets. The precision of German was lower than that on English because of the rich morphology and syntax of the German language as well as the scarcity of German

training data. Overall, the new version of BOA achieves a significantly higher recall by two means. First, we used surface forms to retrieve entities. In addition, we devised an approach to extract triples from datatype properties.

REAL-TIME RDF EXTRACTION FROM UNSTRUCTURED DATA STREAMS

The vision behind the Data Web is to extend the current document-oriented Web with machine-readable facts and structured data, thus creating a representation of general knowledge. However, most of the Data Web is limited to being a large compendium of encyclopedic knowledge describing entities. A huge challenge, the timely and massive extraction of RDF facts from unstructured data, has remained open so far. The availability of such knowledge on the Data Web would provide significant benefits to manifold applications including news retrieval, sentiment analysis and business intelligence. In this chapter, we address the problem of the actuality of the Data Web by presenting an approach that allows extracting RDF triples from unstructured data streams. We employ statistical methods in combination with deduplication, disambiguation and unsupervised as well as supervised machine learning techniques to create a knowledge base that reflects the content of the input streams. We evaluate a sample of the RDF we generate against a large corpus of news streams and show that we achieve a precision of more than 85%.

*This chapter is
mainly based on
Gerber et al. [2013]
and Röder et al.
[2014].*

4.1 INTRODUCTION

Implementing the original vision behind the Semantic Web requires the provision of a Data Web which delivers timely data at all times. The foundational example presented by [Berners-Lee et al. \[2001\]](#)'s seminal paper on the Semantic Web describes a software agent which is tasked to find medical doctors with a rating of excellent or very good within 20 miles of a given location at a given point in time. This requires having timely information on which doctors can be found within 20 miles of a particular location at any given time as well as having explicit data on the rating of said medical doctors. Even stronger timeliness requirements apply in decision support, where software agents help humans to decide on critical issues such as whether to buy stock or not or even how to plan their drive through urban centers. Furthermore, knowledge bases in the Linked Open Data (LOD) cloud would be unable to answer queries such as "Give me all last week's news from the New York Times pertaining to the director of a company". Although the current LOD cloud has tremendously grown over the last years [[Auer et al., 2011](#)], it delivers mostly encyclopedic information (such as albums, places, kings, etc.) and fails to provide up-to-date information that would allow addressing the information needs described in the examples above.

The idea which underlies our work is thus to alleviate this current drawback of the Data Web by developing an approach that allows extracting RDF from unstructured (i.e. textual) data streams in a fashion similar to the live versions of the DBpedia¹ and LinkedGeoData² datasets. The main difference is yet that instead of relying exclusively on structured data like LinkedGeoData or on semi-structured data like DBpedia, we rely mostly on unstructured, textual data to generate RDF. By these means, we are able to unlock some of the potential of the Document Web, of which up to 85% is unstructured [[Gaag et al., 2009](#)]. To achieve this goal, our approach, dubbed RdfLiveNews, assumes that it is given unstructured data streams as input. These are deduplicated and then used as basis to extract patterns for relations between known resources. The patterns are then clustered to labeled relations which are finally used as basis for generating RDF triples. We evaluate our approach against a sample of the RDF triples we extracted from RSS feeds and show that we achieve a very high precision.

The remainder of this chapter is structured as follows: We first give an overview of our approach and give detailed insights in the different steps from unstructured data streams to RDF. Then, we evaluate our approach in several settings. We then contrast our approach with the state of the art and finally conclude.

¹ <http://live.dbpedia.org/sparql>

² <http://live.linkedgeodata.org/sparql>

AUTHOR CONTRIBUTIONS The author of this thesis was the lead author of Gerber et al. [2013] as well as the main developer of the RdfLiveNews extraction framework. He developed the data acquisition module, the pattern search, filtering, refinement and clustering and was responsible for the generation and analysis of the evaluation data. The deduplication, cluster labeling and merging as well as the RDF generation tasks were carried out by the co-authors of Gerber et al. [2013].

4.2 OVERVIEW

We implemented the general architecture of our approach dubbed RdfLiveNews according to the pipeline depicted in Figure 16. First, we gather textual data from data streams by using RSS feeds of news articles. Our approach can yet be employed on any unstructured data published as stream. Since input streams from the Web can be highly redundant (i.e. convey the same information), we then deduplicate the set of streams gathered by our approach. Subsequently, we apply a pattern search to find lexical patterns for relations expressed in the text. After a refinement step with background knowledge, we finally cluster the extracted patterns according to their semantic similarity and transform this information into RDF.

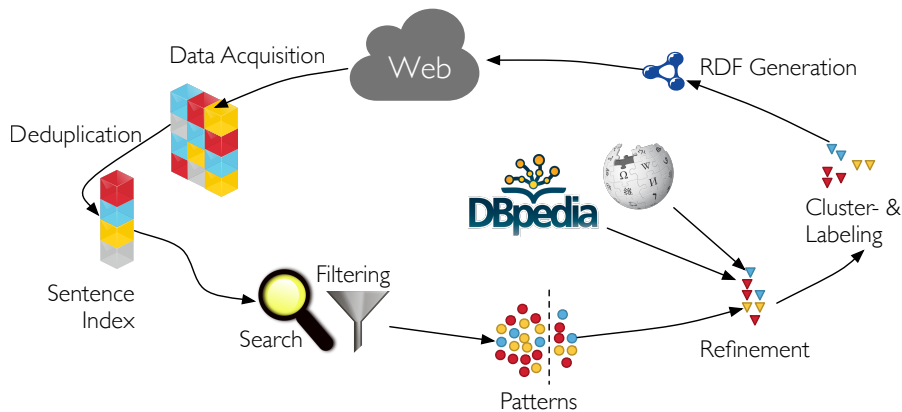


Figure 16: Overview of the generic time slice based stream processing.

4.2.1 Data Acquisition

Formally, our approach aims to process the output of unstructured data sources S^i by continuously gathering the data streams D^i that they generate. Each data stream consists of atomic elements d_j^i (in our case sentences). Let $D_{[t,t+d]}^i$ be the portion of D^i that was emitted by S^i between the times t and $t + d$. The data gathering begins by iteratively gathering the elements of the streams $D_{[t,t+d]}^i$ from all

available sources S^i for a period of time d , which we call the *time slice duration*. For example, this could mean crawling a set of RSS feeds for a duration of two hours. We call $D_{[t,t+d]}^i$ a slice of D^i . We will assume that we begin this process at $t = 0$, thus leading to slices $D_{[k.d,(k+1).d]}^i$ with $k \in \mathbb{N}$. The data gathered from all sources during a time slice duration is called a *time slice*. We apply sentence splitting on all slices to generate their elements.

4.2.2 Deduplication

The aim of the deduplication step is to remove very similar elements from slices before the RDF extraction. This removal accounts for some Web data streams simply repeating the content of one of several other streams. Our deduplication approach is based on measuring the similarity of single elements s_i and s_j found in unstructured streams. Elements of streams are considered to be different iff $qgrams(s_i, s_j) < \theta$, where $\theta \in [0, 1]$ is a similarity threshold and $qgrams(s_i, s_j)$ measures the similarity of two strings by computing the Jaccard similarity of the trigrams they contain. Given that the number of stream items to deduplicate can be very large, we implemented the following two-step approach: For each slice $D_{[k.d,(k+1).d]}^i$, we first deduplicate the elements s_j^i within $D_{[k.d,(k+1).d]}^i$. This results in a duplicate-free data stream $\Delta_{[k.d,(k+1).d]}^i = \{d_j^i : (d_j^i \in D_{[k.d,(k+1).d]}^i) \wedge (\forall s_k^i \in D_{[k.d,(k+1).d]}^i \exists d_j^i \in \Delta_{[k.d,(k+1).d]}^i qgrams(s_k^i, d_j^i) \geq \theta) \wedge (\forall d_j^i, d_k^i \in \Delta_{[k.d,(k+1).d]}^i qgrams(d_k^i, d_j^i) < \theta)\}$. The elements of $\Delta_{[k.d,(k+1).d]}^i$ are then compared to all other elements of the w previous deduplicated streams $\Delta_{[(k-1).d,k.d]}^i$ to $\Delta_{[(k-w).d,(k-w+1).d]}^i$ where w is the size of the deduplication window. Only $\Delta_{[k.d,(k+1).d]}^i$ is used for further processing. To ensure the scalability of the deduplication step, we are using deduplication algorithms implemented in the LIMES framework [Ngonga Ngomo, 2012]. Table 17 gives an overview of the number of unique data stream items in our dataset when using different deduplication thresholds.

4.2.3 Pattern Search and Filtering

In order to find patterns we first apply Named Entity Recognition (NER) and Part of Speech (POS) tagging on the deduplicated sentences. RdfLiveNews can use two different ways to extract patterns from annotated text. The POS tag method uses *NNP* and *NNPS*³ tagged tokens to identify a relation's subject and object, whereas the Named Entity Tag method relies on *Person*, *Location*, *Organization* and *Miscellaneous* tagged tokens. In an intermediate step all consecutive POS and NER tags are merged. An unrefined RdfLiveNews pattern p

³ All POS tags can be found in the Penn Treebank Tagset.

is now defined as a pair $p = (\theta, \mathcal{S}_{\subseteq})$, where θ is the natural language representation (NLR) of p and $\mathcal{S}_{\subseteq} = \{(s_i, o_i) : i \in \mathbb{N}; 1 \leq i \leq n\}$ is the support set of θ , a set of the subject and object pairs. For example the sentence:

David/NNP hired/VBD John/NNP ,/,
former/JJ manager/NN of/IN ABC/NNP ./.

would result in the patterns:

$p_1 = ([\text{hired}], \{(David, John)\})$ and
 $p_2 = ([, \text{ former manager of}], \{(John, ABC)\})$.

After the initial pattern acquisition step, we filter all patterns to improve their quality. We discarded all patterns that did not match these criteria: The pattern should (1) contain at least a verb or a noun, (2) contain at least one salient word (i.e. a word that is not a stop word), (3) not contain more than one non-alpha-numerical character (except ", ' ") and (4) be shorter than 50 characters. Since the resulting list still contains patterns of low quality, we first sort it by the number of elements of the support set \mathcal{S}_{θ} and solely select the top 1% for pattern refinement to ensure high quality.

4.2.4 Pattern Refinement

The goal of this step is to find a suitable *rdfs:range* and *rdfs:domain* as well as to disambiguate the support set of a given pattern. To achieve this goal we first try to find an URI for the subjects and objects in the support set of p by matching the pairs to entries in a knowledge base. With the help of those URIs we can query the knowledge base for the classes (*rdf:type*) of the given resources and compute a common *rdfs:domain* for the subjects of p and *rdfs:range* for the objects respectively. A refined RdfLiveNews pattern p_r is now defined as a quadruple $p_r = (\theta, \mathcal{S}_{\subseteq}', \delta, \rho)$, where θ is the natural language representation, \mathcal{S}_{\subseteq}' the disambiguated support set, δ the *rdfs:domain* and ρ the *rdfs:range* of p_r .

To find the URIs of each subject-object pair $(s, o) \in \mathcal{S}_{\subseteq}$ we first try to complete the entity name. This step is necessary and beneficial because entities usually get only written once in full per article. For example the newly elected president of the United States of America might be referenced as "President Barack Obama" in the first sentence of a news entry and subsequently be referred to as "Obama". In order to find the subjects' or objects' full name, we first select all named entities $e \in \mathcal{E}_a$ of the article the pair (s, o) was found in. We then use the longest matching substring between s (or o) and all elements of \mathcal{E}_a as the name of s or o respectively. Additionally we can filter the elements of \mathcal{E}_a to contain only certain NER types. Once the

complete names of the entities are found, we can use them to generate a list of URI candidates \mathcal{C}_{uri} . This list is generated with the help of a query for the given entity name on a list of surface forms (e.g. “U.S.” or “USA” for the *United States of America*), which was compiled by analyzing the *redirect* and *disambiguation* links from Wikipedia as presented by Mendes et al. [2011]. Each URI candidate $c \in \mathcal{C}_{\text{uri}}$ is now evaluated on four different features and the combined score of those features is used to rank the candidates and choose the most probable URI for an entity. The first feature is the *Apriori*-score $a(c)$ of the URI candidate c , which is calculated beforehand for all URIs in the knowledge base by analyzing the number of inbound links of c by the following formula: $a(c) = \log(\text{inbound}(c) + 1)$. The second and third features are based on the context information found in the Wikipedia article of c and the news article text (s, o) was found in. For the *global context*-score c_g we apply a co-occurrence analysis of the entities \mathcal{E}_a found in the news article and the entities \mathcal{E}_w found in the Wikipedia article of c . The *global context*-score is now computed as $c_g(\mathcal{E}_a, \mathcal{E}_w) = |\mathcal{E}_a \cap \mathcal{E}_w| / |\mathcal{E}_a \cup \mathcal{E}_w|$. The *local context*-score c_l is the number of mentions of the second element of the pair (s, o) , o in the case of s and vice versa, in \mathcal{E}_w . The last feature to determine a URI for an entity is the maximum string similarity sts between s (or o) and the elements of the list of surface forms of c . We used the qgram distance⁴ as the string similarity metric. We normalize all non-[0, 1] features (c_g, c_l, a) by applying a minimum-maximum normalization of the corresponding scores for \mathcal{C}_{uri} and multiply it with a weight parameter which leads to the overall URI score:

$$c(s, o, \text{uri}) = \frac{\frac{\alpha a}{a_{\max}} + \frac{\beta c_g}{c_{g_{\max}}} + \frac{\gamma c_l}{c_{l_{\max}}} + \delta sts}{4}$$

If the URI’s score is above a certain threshold $\lambda \in [0, 1]$ we use it as the URI for s , otherwise we create a new URI. Once we have computed the URIs for all pairs $(s, o) \in \mathcal{S}_{\subseteq}$ we determine the most likely *domain* and *range* for p_r . This is done by analyzing the *rdf:type* statements returned for each subject or object in \mathcal{S}_{\subseteq} from a background knowledge base. Since the DBpedia ontology is designed in such a way, that classes do only have one super-class, we can easily analyze its hierarchy. We implemented two different determination strategies for analyzing the class hierarchy. The first strategy, dubbed “most general”, selects the highest class in the hierarchy for each subject (or object) and uses the most occurring class as *domain* or *range* of p_r . The second strategy, dubbed “most specific”, works similar to the “most general” strategy with the difference that it uses the most descriptive class to select the *domain* and *range* of p_r .

⁴ <http://sourceforge.net/projects/simmetrics/>

4.2.5 Pattern Similarity and Clustering

In order to cluster patterns according to their meaning, we created a set of similarity measures. A similarity measure takes two patterns p_1 and p_2 as input and outputs the similarity value $s(p_1, p_2) \in [0, 1]$. As a baseline we implemented a qgram measure, which calculates the string similarity between all non stop words of two patterns. Since this baseline measure fails to return a high similarity for semantically related, but not textually similar patterns like “’s attorney ,” and “’s lawyer ,” we also implemented a Wordnet measure. As a first step the Wordnet similarity measure filters out the stop words of p_1 and p_2 and applies the Stanford lemmatizer on the remaining tokens. Subsequently, for all token combinations of p_1 and p_2 , we apply a Wordnet Similarity metric (Path [Pedersen et al., 2004], Lin [Lin, 1998] and Wu & Palmer [Wu and Palmer, 1994]) and select the maximum of all comparisons as the similarity value $s(p_1, p_2)$. As a final similarity measure we created a Wordnet and string similarity measure with the help of a linear combination from the aforementioned metrics. In this step we also utilize the *domain* and *range* of p_r . If this feature is activated, a similarity value between two patterns p_1 and p_2 can only be above 0, iff $\{\delta_{p_1}, \rho_{p_1}\} \setminus \{\delta_{p_2}, \rho_{p_2}\} = \emptyset$.

The result of the similarity computation can be regarded as a similarity graph $G = (V, E, \omega)$, where the vertices are patterns and the weight $\omega(p_1, p_2)$ of the edge between two patterns is the similarity of these patterns. Consequently, unsupervised machine learning and in particular graph clustering is a viable way of finding groups of patterns that convey similar meaning. We opted for using the Border-Flow clustering algorithm [Ngonga Ngomo and Schumacher, 2009] as it is parameter-free and has already been used successfully in diverse applications including clustering protein-protein interaction data and queries for SPARQL benchmark creation [Morsey et al., 2011]. For each node $v \in V$, the algorithm begins with an initial cluster X containing only v . Then, it expands X iteratively by adding nodes from the direct neighborhood of X to X until X is node-maximal with respect to the border flow ratio described in [Morsey et al., 2011]. The same procedure is repeated for all nodes. As different nodes can lead to the same cluster, identical clusters (i.e. clusters containing exactly the same nodes) that resulted from different nodes are subsequently collapsed to one cluster. The set of collapsed clusters and the mapping between each cluster and the nodes that led to it are returned as result.

4.2.6 Cluster Labeling and Merging

Based on the clusters \mathcal{C} obtained through the clustering algorithm, this step selects descriptive labels for each cluster $c_i \in \mathcal{C}$, which can

afterwards be used to merge the clusters. In the current version, we apply a straightforward majority voting algorithm, i.e. for each cluster c_i , we select the most frequent natural language representation θ (stop words removed) occurring in the patterns of c_i . Finally, we use the representative label of the clusters to merge them using a string similarity and WordNet based similarity measure. This merging procedure can be applied repeatedly to further reduce the number of clusters, but taking into account that those similarity measures are not transitive, we are currently only running it once, as we are more focused on accuracy.

4.2.7 Mapping to RDF and Publication on the Data Web

To close the circle of the round-trip pipeline of RdfLiveNews, the following prerequisite steps are required to re-publish the extraction results in a sensible way:

1. The facts and properties contained in the internal data structure of our tool have to be mapped to OWL.
2. Besides the extracted factual information several other aspects and meta data are interesting as well, such as extraction and publication data and provenance links to the text the facts were extracted from.
3. URIs need to be minted to provide the extracted triples as linked data.

MAPPING TO OWL Each cluster $c_i \in \mathcal{C}$ represents an *owl:ObjectProperty* prop_{c_i} . The *rdfs:domain* and *rdfs:range* of prop_{c_i} is determined by a majority voting algorithm with respect to δ and ρ of all $p_r \in \mathcal{C}$. The *skos:prefLabel*⁵ of prop_{c_i} is the label determined by the cluster labeling step and all other NLRs of the patterns in c_i get associated with prop_{c_i} as *skos:altLabels*. For each subject-object pair in \mathcal{S}_{c_i}' we produce a triple by using prop_{c_i} as predicate and by assigning learned entity types from DBpedia or *owl:Thing*.

PROVENANCE TRACKING WITH NIF Besides converting the extracted facts from the text, we are using the current draft of the NLP Interchange Format (NIF) Core ontology⁶ to serialize the following information in RDF: the sentence the triple was extracted from, the extraction date of the triple, the link to the source URL of the data stream item and the publication date of the item on the stream. Furthermore, NIF allows us to link each element of the extracted triple to its origin in the text for further reference and querying.

⁵ <http://www.w3.org/2004/02/skos/>

⁶ <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#>

NIF is an RDF/OWL based format to achieve interoperability between language tools, annotations and resources. NIF offers several URI schemes to create URIs for strings, which can then be used as subjects for annotation. We employ the NIF URI scheme, which is grounded on URI fragment identifiers for text (RFC 5147⁷). NIF was previously used by NERD [Rizzo et al., 2012] to link entities to text. For our use case, we extended NIF in two ways: (1) we added the ability to represent extracted triples via the ITS 2.0 / RDF Ontology⁸. *itsrdf:taPropRef* is an *owl:AnnotationProperty* that links the NIF String URI to the *owl:ObjectProperty* by RdfLiveNews. The three links from the NIF String URIs (str_1 , str_2 , str_3) to the extracted triple (s , p , o) itself make it well traceable and queryable: $str_1 \mapsto s$, $str_2 \mapsto p$, $str_3 \mapsto o$, $s \mapsto p \mapsto o$. An example of NIF RDF serialization is shown in Listing 8. (2) Although Rizzo et al. [2012] already suggested the minting of new URIs, a concrete method for doing so was not yet researched. In RdfLiveNews we use the source URL of the data stream item to re-publish the facts for individual sentences as linked data. We strip the scheme component (`http://`) of the source URL. Then, we percent encode the ultimate part of the path and the query component⁹ and add the md5 encoded sentence to produce the following URI:

```
http://rdflivenews.aksw.org/extraction/ + example.com:8042/over/ +
  urlencode(there?name=ferret) + / + md5('sentence')
```

```

1 @base <http://rdflivenews.aksw.org/extraction/www.necn.com/07/04/12/Scientists-
  discover-new-subatomic-partic/landing.html%3FblockID%3D735470%26feedID%3D4213
  /8ale5928f6815c99b9d2ce613cf24198#>.
2 ## prefixes: please use http://prefix.cc, e.g. http://prefix.cc/rlno
3 ## extracted property + result of linking
4 rlno:directorOf a owl:ObjectProperty ;
5   skos:prefLabel "director of" , skos:altLabel " , director of " ;
6   owl:equivalentProperty dbp:director .
7 ## extracted facts:
8 rlnr:Rolf_Heuer a dbo:Person ;
9   rdfs:label "Rolf Heuer"@en ;
10  rlno:directorOf dbpedia:CERN .
11 dbpedia:CERN a owl:Thing ;
12   rdfs:label "CERN"@en .
13 ## provenance tracking with NIF:
14 <char=0,10> itsrdf:taClassRef dbo:Person ;
15   itsrdf:taIdentRef rlnr:Rolf_Heuer .
16 <char=14,18> itsrdf:taIdentRef dbpedia:CERN .
17 <char=11,24> nif:anchorOf " , director of"^^xsd:string ;
18   itsrdf:taPropRef rlno:directorOf .
19 ## detailed NIF output with context, indices and anchorOf
20 <char=0,> a nif:String, nif:Context, nif:RFC5147String ;
21   nif:isString "Rolf Heuer , director of CERN , said the newly discovered particle
  is a boson , but he stopped just shy of claiming outright that it is the
  Higgs boson itself - an extremely fine distinction." ;
22   nif:sourceUrl <http://www.necn.com/07/04/12/Scientists-discover-new-subatomic-
  partic/landing.html?blockID=735470&feedID=4213>;
23 ## extraction date:
24   dcterms:created "2013-05-09T18:27:08+02:00"^^xsd:dateTime .
25 ## publishing date:
26 <http://www.necn.com/07/04/12/Scientists-discover-new-subatomic-partic/landing.
  html?blockID=735470&feedID=4213>
27   dcterms:created "2012-08-15T14:48:47+02:00"^^xsd:dateTime .
28 <char=0,10> a nif:String, nif:RFC5147String ;
29   nif:referenceContext <char=0,>; nif:anchorOf "Rolf Heuer" ;
30   nif:beginIndex "0"^^xsd:long ; nif:endIndex "10"^^xsd:long ;

```

7 <http://tools.ietf.org/html/rfc5147>

8 <http://www.w3.org/2005/11/its/rdf#>

9 <http://tools.ietf.org/html/rfc3986#section-3>

Listing 8: Example RDF extraction of RdfLiveNews

Republication of RDF. The extracted triples are hosted on: <http://rdflivenews.aksw.org>. The data for individual sentences is crawlable via the file system of the Apache2 web server. We assume that source URLs only occur once in a stream when the document is published and the files will not be overwritten. Furthermore, the extracted properties and entities are available as linked data at

[http://rdflivenews.aksw.org/{ontology|resource}/\\$name](http://rdflivenews.aksw.org/{ontology|resource}/$name)

and they can be queried via SPARQL at <http://rdflivenews.aksw.org/sparql>.

4.2.8 Linking

The approach described above generates a set of properties with several labels. In our effort to integrate this data source into the Linked Open Data Cloud, we use the deduplication approach proposed in Section 4.2.2 to link our set of properties to existing knowledge bases (e.g. DBpedia). To achieve this goal, we consider the set of properties we generated as set of source instances S while the properties of the knowledge base to which we link are considered to be a set of target T . Two properties $s \in S$ and $t \in T$ are linked iff $\text{trigrams}(s, t) \geq \theta_p$, where $\theta_p \in [0, 1]$ is the property similarity threshold.

4.3 EVALUATION

The aim of our evaluation was to answer four questions. First, we aimed at testing how well RdfLiveNews is able to disambiguate found entities. Our second goal was to determine if the proposed similarity measures can be used to cluster patterns with respect to their semantic similarity. Third, we wanted to evaluate the quality of the RDF extraction and linking. Finally, we wanted to measure if all computational heavy tasks can be applied in real-time, meaning the processing of one iteration takes less time than its compilation.

For this evaluation we used a list of 1457 RSS feeds as compiled by Goldhahn et al. [2012]. This list includes all major worldwide newspapers and a wide range of topics, e.g. *World*, *U.S.*, *Business*, *Science* etc. We crawled this list for 76 hours, which resulted in a corpus, dubbed 100% of 38 time slices of 2 hours and 11.7 million sentences. The average number of sentences per feed entry is approximately 26.5 and there are 3445 articles on average per time slice. Additionally, we created two subsets of this corpus by randomly selecting 1% and 10% of the contained sentences. All evaluations were carried out on a MacBook Pro with a quad-core Intel Core i7 (2GHz), a solid state drive and 16 GB of RAM.

4.3.1 URI Disambiguation

To evaluate the URI disambiguation we created a gold standard manually. We took the 1% corpus, applied deduplication with a window size of 40 (contains all time slices) and a threshold of 1 (identical sentences), which resulted in a set of 69884 unique sentences. On those sentences we performed the pattern extraction with part of speech tagging as well as filtering. In total we found 16886 patterns and selected the Top 1%, which have been found by 1729 entity pairs. For 473 of those entity pairs we manually selected a URI for subject and object. This resulted in an almost equally distributed gold standard with 456 DBpedia and 478 RdfLiveNews URIs. We implemented a hill climbing approach with random initialization to optimize the parameters (see Section 4.2.4). The precision of our approach is the ratio between correctly found URIs for subject and object to the number of URIs above the threshold λ as shown in Equation 13. The recall, shown in Equation 14, is determined by the ratio between the number of correct subject and object URIs and the total number of subjects and objects in the gold standard. The F_1 measure is determined as usual by: $F_1 = 2 \cdot \frac{P \cdot R}{P + R}$. We optimized our approach for precision since we can compensate a lower recall and could achieve a precision of **85.01%** where the recall is **40.69%** and the resulting F_1 is **55.03%**. The parameters obtained through the hill-climbing search indicate that the *Apriori*-score is the most influential parameter (1.0), followed by *string-similarity* (0.78), *local-context* (0.6), global context (0.45) and a URI score threshold of 0.61. If we optimize for F_1 , we were able to achieve a F_1 measure of **66.49%** with a precision of **67.03%** and a recall of **65.95%**.

For 487 out of the 934 URI in the gold standard no confident enough URI could be found. The largest number of problems occurred for DBpedia URIs which could not be determined in 305 cases, in comparison to 182 URIs for newly created resources. Additionally, for 30 resources RdfLiveNews created new URIs where DBpedia URIs should be used and in 0 cases a DBpedia URI was used where a new resource should be created. The reasons for those mistakes are tagging errors, erroneous spellings and missing context information. For example Wikipedia has 97 disambiguations for “John Smith” which cannot be disambiguated without prior knowledge.

We used AIDA [Hoffart et al., 2011] to compare our results with a state-of-the-art NED algorithm. We configured AIDA with the Cocktailparty setup, which defines the recommended configuration options of AIDA. AIDA achieved an accuracy of 0.57, i.e. 57% of the identifiable entities were correctly disambiguated. The corpus described above provides a difficult challenge due to the small disambiguation contexts and is limited to graphs evolving from two named entities per text. AIDA tries to build dense sub-graphs in a greedy manner

in order to perform correct disambiguation. This algorithm would profit from a bigger number of entities per text. The drawback is AIDA needs 2 minutes to disambiguate 25 sentences. Overall, AIDA performs well on arbitrary entities.

$$P = \frac{|s_{uri_c}| + |o_{uri_c}|}{|s_{uri}| + |o_{uri}|} \quad (13) \quad R = \frac{|s_{uri_c}| + |o_{uri_c}|}{2 \cdot |GS|} \quad (14)$$

4.3.2 Pattern Clustering

To evaluate the similarity generation as well as the clustering algorithm we relied on the measures Sensitivity, Positive Predictive Value (PPV) and Accuracy. We used the adaptation of those measures as presented by Brohée and van Helden [2006] to measure the match between a set of pattern mappings¹⁰ from the gold standard and a clustering result. The gold standard was created by clustering the patterns as presented in the previous section manually. This resulted in a list of 25 clusters with more than 1 pattern and 54 clusters with 1 pattern. Since clusters with a size of 1 would skew our evaluation into unjustified good results, we excluded them from this evaluation.

Sensitivity. With respect to the clustering gold standard, we define sensitivity as the fraction of patterns of pattern mapping i which are found in cluster j . In $Sn_{i,j} = T_{i,j}/N_i$, N_i is the number of patterns belonging to pattern mapping i . We also calculate a pattern mapping-wise sensitivity Sn_{pm_i} as the maximal fraction of patterns of pattern mapping i assigned to the same cluster. $Sn_{pm_i} = \max_{j=1}^m Sn_{i,j}$ reflects the coverage of pattern mapping i by its best-matching cluster. To characterize the general sensitivity of a clustering result, we compute a clustering-wise sensitivity as the weighted average of Sn_{pm_i} over all pattern mappings:

$$Sn = \frac{\sum_{i=1}^n N_i Sn_{pm_i}}{\sum_{i=1}^n N_i}. \quad (15)$$

Positive Predictive Value. The positive predictive value is the proportion of members of cluster j which belong to pattern mapping i , relative to the total number of members of this cluster assigned to all pattern mappings.

$$PPV_{i,j} = T_{i,j} / \sum_{i=1}^n T_{i,j} = T_{i,j} / T_j \quad (16)$$

T_j is the sum of column j . We also calculate a cluster-wise positive predictive value PPV_{cl_j} , which represents the maximal fraction of patterns of cluster j found in the same annotated pattern mapping. $PPV_{cl_j} = \max_{i=1}^n PPV_{i,j}$ reflects the reliability of cluster j predicting

¹⁰ A pattern mapping maps NLRs to RDF properties.

that a pattern belongs to its best-matching pattern mapping. To characterize the general PPV of a clustering result as a whole, we compute a clustering-wise PPV as the weighted average of PPV_{cl_j} over all clusters:

$$PPV = \frac{\sum_{j=1}^m T_j PPV_{cl_j}}{\sum_{j=1}^m T_j}. \quad (17)$$

Accuracy. The geometric accuracy (Acc) indicates the tradeoff between sensitivity and positive predictive value. It is obtained by computing the geometrical mean of the Sn and the PPV: $Acc = \sqrt{Sn \cdot PPV}$.

We evaluated the three similarity measures with respect to the underlying WordNet similarity metric (see Section 4.2.5). Furthermore we varied the clustering similarity threshold between 0.1 and 1 with a 0.1 step size. In case of the qgram and WordNet similarity metric we performed a grid search on the WordNet and qgram parameter in $[0, 1]$ with a step size of 0.05. We achieved the best configuration with the qgram and WordNet similarity metric with an accuracy of 82.45%, a sensitivity of 71.17% and a positive predictive value of 95.51%. The best WordNet metric is Lin, the clustering threshold 0.3 and the qgram parameter is with 0.45 significantly less influential than the WordNet parameter with 0.75. As reference value, the plain WordNet similarity metric achieved an accuracy of 78.86% and the qgram similarity metric an accuracy of 69.1% in their best configuration.

4.3.3 RDF Extraction and Linking

To assess the quality of the RDF data extracted by RdfLiveNews, we sampled the output of our approach and evaluated it manually. We generated five different evaluation sets. Each set may only contain triples with properties of clusters having at least $i = 1 \dots 5$ patterns. We selected 100 triples (if available) randomly for each test set. As the results in Table 16 show, we achieve high accuracy on subject and object disambiguation. As expected, the precision of our approach grows with the threshold for the minimal size of clusters. This is simply due to smaller clusters having a higher probability of containing outliers and thus noise.

The results of the linking with DBpedia (see Table 18) showed the mismatch between the relations that occur in news and the relations designed to model encyclopedic knowledge. While some relations such as `dbo:director` are used commonly in news streams and in the Linked Data Cloud, relations with a more volatile character such as `rlno:attorney`, which appear frequently in news text, are not mentioned in DBpedia.

E_i	1	2	3	4	5
S_{Acc}	0.81	0.88	0.86	0.857	0.804
P_{Acc}	0.86	0.89	0.90	0.935	1.00
O_{Acc}	0.93	0.91	0.90	0.948	0.941
$Total_{Acc}$	0.86	0.892	0.885	0.911	0.906
$ E_i $	100	100	100	77	51
$ P \in E_i $	28	22	12	6	1

Table 16: Accuracy of RDF Extraction for subjects (S), predicates (P) and objects (O) on 1% dataset with varying cluster sizes E_i .

Time Slice	No deduplication	$\theta = 1.0$	$\theta = 0.95$	$\theta = 0.9$
1	2997	2764	2764	2759
5	3047	2335	2334	2327
10	3113	2033	2040	2022
15	2927	1873	1868	1866
20	3134	1967	1966	1949
25	3065	1936	1932	1924
30	3046	1941	1940	1933

Table 17: Number of non-duplicate sentences in 1% of the data extracted from 1457 RSS feeds within a window of 10 time slices (2h each). The second column shows the original number of sentences without duplicate removal.

4.3.4 Scalability

In order to perform real-time RDF extraction, the processing of the proposed pipeline needs to take up less time than its acquisition requires. This is also required for a growing list of RSS feeds. Therefore, we analyzed the execution time for each module in each iteration and compared these values between the three test corpora. An early approximation of this evaluation implied that the pipeline indeed was not fast enough, which led to the parallelization of the pattern refinement and similarity generation. The results of this evaluation can be seen in Figure 17. With an average time slice processing time of about 20 minutes for the 100% corpus (2.2 minutes for 10% and 30s for 1%), our approach is clearly fit to handle up to 1500 RSS and more. The spike in the first iteration is a result of the fact that RSS feeds contain the last n previous entries, which leads to a disproportional large first time slice. The most time consuming modules are the deduplication, tagging and cluster merging. To tackle these bottlenecks we can for example parallelize sentence tagging and the deduplication.

RdfLiveNews-URI	DBpedia-URI	Sample of cluster
rln:directorOf	dbo:director	[manager], [, director of], [, the director of]
rln:spokesperson	dbo:spokesperson	[, a spokeswoman for], [spokesperson], [, a spokesman for]
rln:attorney	—	[’s attorney ,], [’s lawyer ,], [attorney]

Table 18: Example for linking between RdfLiveNews and DBpedia.

The results of the growth evaluation for patterns until iteration 30 can be seen in Figure 18. The number of patterns grows with the factor of 3 from 1% to 10% and 10% to 100% corpora. Also, the number of patterns found by more than one subject-object pair increases approximately by factor 2. Additionally we observed a linear growth for all patterns (also for patterns with $|S'_0| > 1$) and 100% showing the highest growth rate with a factor 2.5 over 10% and 4.8 over 10%.

The results of the growth evaluation for clusters can be seen in Figure 19. The evaluation shows that the number of clusters increases by a factor of 2.5 from 1% to 10% and 10% to 100%. Moreover, approximately 25% of all clusters have more than 1 pattern and the number of clusters grows linear for 1% and 10% but for the 100% corpus it seems to converge to 800. The same holds true for clusters with more than one pattern, as they stop to grow at around 225 clusters.

4.4 RELATED WORK

While Semantic Web applications rely on formal, machine understandable languages such as RDF and OWL, enabling powerful features such as reasoning and expressive querying, humans use Natural Language (NL) to express semantics. This gap between the two different languages has been filled by Information Extraction (IE) approaches, developed by the Natural Language Processing (NLP) research community [Sarawagi, 2008], whose goal is to find desired pieces of information, such as concepts (hierarchy of terms which are used to point to shared definitions), entities (name, numeric expression, date) and facts in natural language texts and print them in a form that is suitable for automatic querying and processing. Ever since the advent of the Linked Open Data initiative¹¹, IE is also an important key enabler for the Semantic Web. For example, LODifier ([Augenstein et al., 2012; Exner and Nugues, 2012]) combines deep semantic analysis with Named Entity Recognition, word-sense

¹¹ <http://linkeddata.org/>

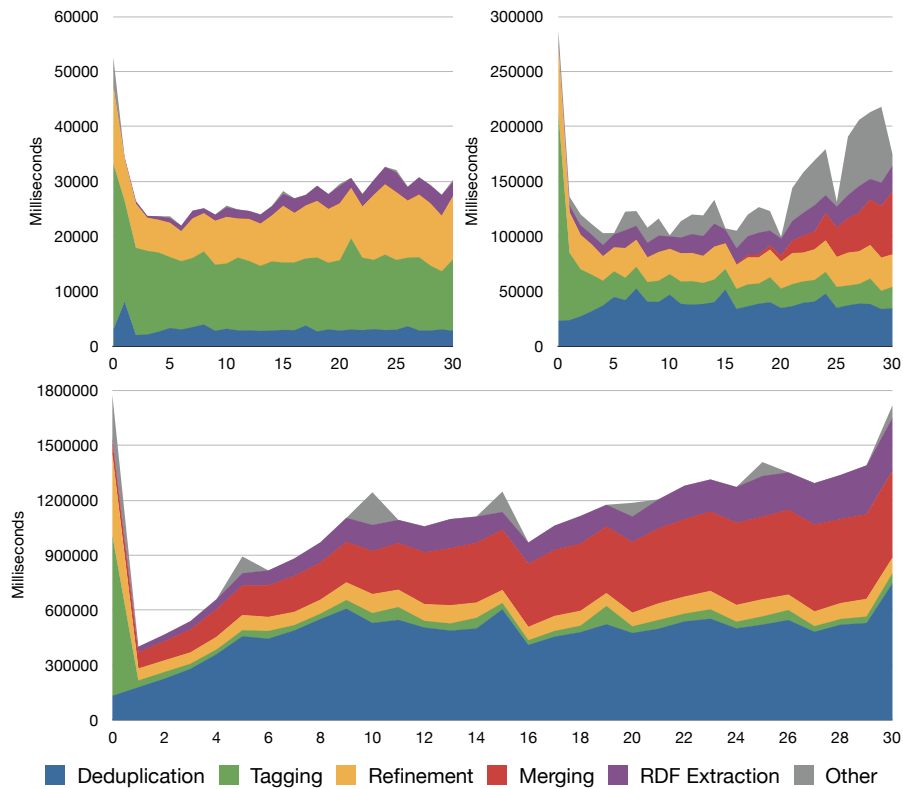


Figure 17: Runtimes for different components and corpora (1% left, 10% right, 100% bottom) per iteration.

disambiguation and controlled Semantic Web vocabularies. FOX [Ngonga Ngomo et al., 2011b] uses ensemble learning to improve the F-score of IE tools. The BOA framework [Gerber and Ngonga Ngomo, 2011] uses structured data as background knowledge for the extraction of natural language patterns, which are subsequently employed to extract additional RDF data from natural language text. Nakashole and Weikum [2012] propose a simple model for fact extraction in real-time, taking into account the vexing challenges that timely fact extraction on frequently updated data entails. A specific application for the news domain is described by Stern and Sagot [2012], wherein a knowledge base of entities for the French news agency AFP is populated.

State-of-the-art open-IE systems such as ReVerb automatically identify and extract relationships from text, relying on (in the case of ReVerb) simple syntactic constraints expressed by verbs [Fader et al., 2011]. Davidov and Rappoport [2008] present a novel pattern clusters method for nominal relationship classification using an unsupervised learning environment, which makes the system domain and language-independent. Ruiz-Casado et al. [2007] show how lexical patterns and semantic relationships can be learned from concepts in Wikipedia.

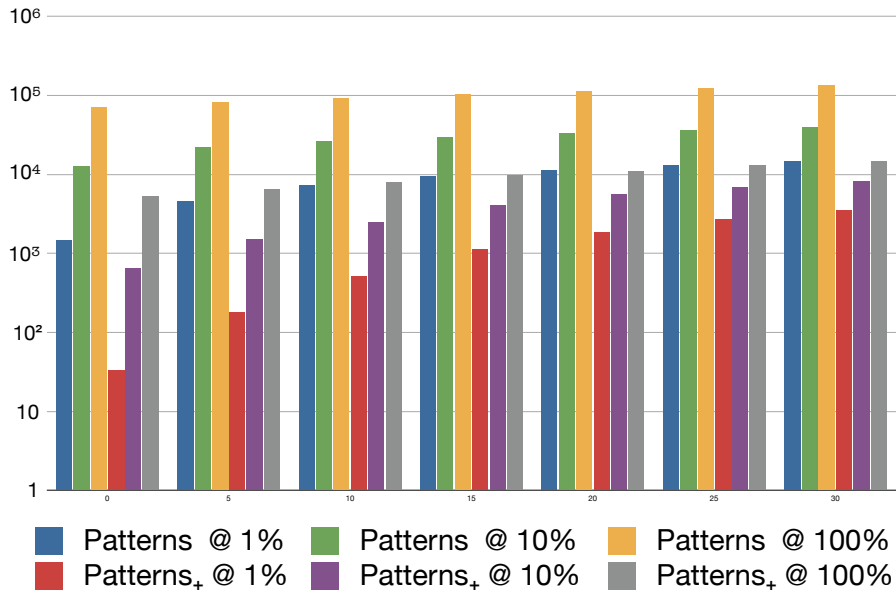


Figure 18: Number of patterns (log scale) and patterns with $|\mathcal{S}'_{\theta}| > 1$ (Patterns₊) for iterations and test corpus.

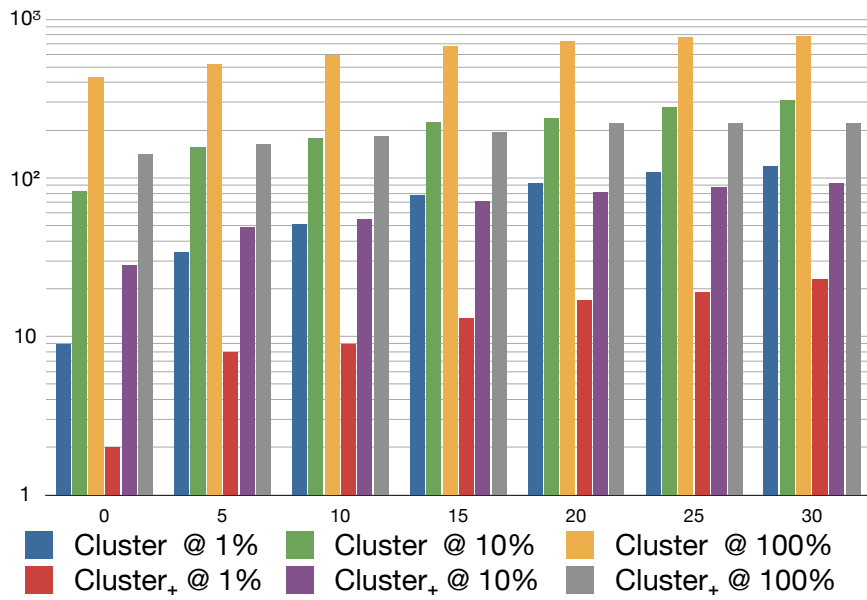


Figure 19: Number of clusters (log scale) and clusters with $|\mathcal{C}| > 1$ (Cluster₊) for iterations and test corpus.

4.5 CONCLUSION

In this chapter, we introduced RdfLiveNews, a framework for the extraction of RDF from unstructured data streams. We presented the components of the RdfLiveNews framework and evaluated its disambiguation, clustering, linking and scalability capabilities as well as its extraction quality. We are able to disambiguate resources with a precision of 85%, cluster patterns with an accuracy of 82.5% and extract RDF with a total accuracy of around 90% and handle two hour time slices with around 300.000 sentences within 20 minutes on a small server.

Part II

APPLICATIONS OF MULTILINGUAL NATURAL LANGUAGE PATTERNS

The second part of this thesis focuses on applications utilizing the multilingual natural language patterns generated by BOA and RdfLiveNews. This part therefore provides solutions for the quality and provenance problems of the current state of the Semantic Web. We first introduce DeFacto, a system which is able to verify or falsify a RDF triple. DeFacto does not only search for textual occurrences of parts of the statement, but also seeks to find webpages which contain the actual statement phrased in natural language. We then prove that our approach can be applied to multiple languages (English, French and German). Furthermore, we introduce a method to temporally scope facts. Finally, we demonstrate that these patterns can be used successfully in an ontology linking problem area in a question answering and a dictionary population task.

One of the main tasks when creating and maintaining knowledge bases is to validate facts and provide sources for them in order to ensure correctness and traceability of the provided knowledge. So far, this task is often addressed by human curators in a three-step process: issuing appropriate keyword queries for the statement to check using standard search engines, retrieving potentially relevant documents and screening those documents for relevant content. And yet, the drawbacks of this process are manifold. Most importantly, it is very time-consuming as the experts have to carry out several search processes and must often read several documents. In this chapter, we present DeFacto (Deep Fact Validation) – an algorithm able to validate facts by finding trustworthy sources for them on the Web. DeFacto aims at providing an effective way of validating facts by supplying the user with relevant excerpts of webpages as well as useful additional information including a confidence score for the correctness of the input fact. The first version of DeFacto, which is presented in Section 5.3, focuses, like the initial version of the BOA Framework, on the most dominant Web language, English. We evaluate our approach on 60 relations found on the Data Web covering a multitude of domains, e.g. biology, music, politics and movies. The extended version of DeFacto achieves the goal of automatically validating input facts by collecting and combining evidence from webpages written in several languages. In addition, it provides support for facts with a temporal scope, i.e. it can estimate in which time frame a fact was valid. Given that the automatic evaluation of facts has not been paid much attention to so far, generic benchmarks for evaluating these frameworks were not available so far. We thus also present a generic evaluation framework for fact checking and make it publicly available.

This chapter is mainly based on Lehmann et al. [2012b] and Gerber et al. [2015].

5.1 INTRODUCTION

The past decades have been marked by a change from an industrial society to an information and knowledge society. This change is particularly due to the uptake of the World Wide Web. Creating and managing knowledge successfully has been a key to success in various communities worldwide. Therefore, the quality of knowledge is of high importance. One aspect of knowledge quality is provenance. In particular, the sources for facts should be well documented, since this provides several benefits such as a better detection of errors, decisions based on the trustworthiness of sources etc. While provenance is an important aspect of data quality [Hartig, 2009], currently only few knowledge bases actually provide provenance information. For instance, less than 3% of the more than 607.7 million RDF documents indexed by Sindice¹ in June 2012 contain metadata such as creator, created, source, modified, contributor, or provenance.² This coverage increased to around 10% for the 708.26 million documents indexed by Sindice in December 2013, but is still far from optimal.

This lack of provenance information makes the validation of the facts in such knowledge bases utterly tedious. In addition, it impedes the adoption of such data in business applications as the data is not trusted [Hartig, 2009]. Our main contribution is the provision of a fact validation approach and tool which can make use of one of the largest sources of information: the Web.

In the first part of this chapter we present the DeFacto system (Deep Fact Validation) which implements algorithms for validating RDF triples by finding confirming sources for it on the web. It takes a statement as input (e.g. that shown in Listing 9) and then seeks evidence for the validity of that statement by searching for textual information on the Web. In contrast to conventional search engines, it does not only search for textual occurrences of parts of the statement, but also tries to find webpages which contain the actual statement phrased in natural language. It provides the user with a confidence score for the input statement as well as a set of excerpts of relevant webpages, which allows the user to manually judge the presented evidence.

DeFacto has two major use cases:

1. Given an existing true statement, it can be used to find provenance information for it. For instance, the WikiData project³ aims at creating a collection of facts, in which sources should be provided for each fact. DeFacto helps to achieve this task.

¹ <http://www.sindice.com>

² Data retrieved on June 6, 2012.

³ <http://meta.wikimedia.org/wiki/Wikidata>

2. It can check whether a statement is likely to be true, provide the user with a confidence score in whether the statement is true, and evidence for the score assigned to the statement.

Our main contributions are thus as follows:

1. an approach that allows checking whether a webpage confirms a fact, i.e. an RDF triple,
2. an adaptation of existing approaches for determining indicators for trustworthiness of a webpage,
3. an automated approach to enhance knowledge bases with RDF provenance data at triple level as well as
4. a running prototype of DeFacto, the first system able to provide useful confidence values for an input RDF triple given the Web as background text corpus.

In the second part of this chapter we present the multilingual and temporal extension of the DeFacto system. Again, our system implements algorithms for validating RDF triples by finding confirming sources for them on the Web.⁴ It takes a fact as input (e.g. the one shown in Listing 10) and then tries to find evidence for the validity of that statement by searching for textual information on the Web. To this end, our approach combines two strategies by searching for textual occurrences of parts of the statements on the one hand as well as trying to find webpages which contain the input statement expressed in natural language on the other hand. DeFacto was conceived to exploit the multilingualism of the Web, as almost half of the content of the Web is written in a language other than English⁵ (see Figure 7).

To this end, our approach abstracts from a specific language and can combine evidence from multiple languages – currently English, German and French. The output of our approach is a confidence score for the input statement as well as a set of excerpts of relevant webpages, which allows the user to manually judge the presented evidence. Apart from the general confidence score, DeFacto also provides support for detecting the temporal scope of facts, i.e. it estimates in which timeframe a fact is or was valid.

The multilingual and temporal extension of DeFacto has three major use cases. The first and second use case are similar to the use cases described for the baseline DeFacto system, namely to generate provenance data, confidence scores and textual evidence. DeFacto's final use case is:

3. Given a fact, DeFacto can determine and present evidence for the time interval within which the said fact is to be considered valid.

⁴ Please note that we use *fact* as a synonym for a *RDF triple*.

⁵ 45% non-English webpages according to http://w3techs.com/technologies/overview/content_language/all.

Our main contributions are thus as follows:

- an extension of the baseline approach, allowing to check whether a webpage, written in a supported language, confirms a fact, i.e. an RDF triple,
- a generalization approach of multilingual and natural language patterns for formal relation in the BOA Framework,
- discuss an adaptation of existing approaches for determining indicators for trustworthiness of a webpage in different languages,
- we present an automated approach to enhance knowledge bases with RDF provenance data at triple level,
- a temporal extension detecting the temporal scope of facts based on text understanding via pattern and frequency analysis.
- a multilingual lexical pattern library for detecting time periods in text,
- an extensive study of the effects of the novel multilingual support in DeFacto, e.g. through the integration of search queries and temporal patterns in several languages.
- a freely available and full-fledged benchmark for fact validation which includes temporal scopes as well as
- an extended prototype of DeFacto.

The rest of this chapter is structured as follows: We start in Section 5.2 by summarizing related work. Section 5.3 describes our baseline approach and the system infrastructure. The next section describes how we extended the BOA framework to enable it to detect facts contained in textual descriptions on webpages. In Section 5.4.2, we describe how we include the trustworthiness of webpages into the DeFacto analysis. Section 5.4.3 combines the results from the previous chapters and describes the mathematical features we use to compute the confidence for a particular input fact. We use those features to train different classifiers in Section 5.4.4 and describe our evaluation results.

In Section 5.5 we start with a description of the overall approach in a nutshell and exemplify the provenance output generated by DeFacto. We display how we extended the BOA framework in order to generalize lexical patterns and build multilingual search queries in Section 5.5.1. In Section 5.5.4, we describe how we calculate and include the trustworthiness of webpages into the multilingual and temporal DeFacto analysis. Section 5.5.5 combines the results from the previous chapters and describes the mathematical features we use to compute the confidence for a particular input fact. Subsequently, we

describe the temporal extension of DeFacto in Section 5.5.6 and provide an overview of the FactBench benchmark in Section 5.5.7. We present a discussion of the evaluation results in Section 5.5.8.

Finally, we conclude in Section 5.6.

AUTHOR CONTRIBUTIONS The author of this thesis was the main developer of the DeFacto fact validation framework. In particular, he was responsible for training the BOA framework for this task, extending the BOA architecture to support multiple languages as well as pattern generalization, the implementation of the machine learning modules for fact and proof scoring, the generation of the FactBench benchmark, the temporal scoping of input facts and the evaluation of the proposed algorithms. The implementation of the first as well as the extended GUI, the RDF provenance NIF output, the Named Entity Disambiguation and the implementation of the statistical triple evidence feature were carried out by the co-authors of [Lehmann et al. \[2012b\]](#); [Gerber et al. \[2015\]](#).

5.2 RELATED WORK

The work presented in this chapter is related to five main areas of research: fact finding as known from NLP, the representation of provenance information in the Data Web, temporal analysis, relation extraction and named entity disambiguation (also called entity linking).

FACT FINDING Fact finding is a relatively new research area which focuses on computing which subset of a given set of statements can be trusted [[Pasternack and Roth, 2013](#)]. Several approaches have been developed to achieve this goal. [Nakamura et al. \[2007\]](#) developed a prototype for enhancing the search results provided by a search engine based on trustworthiness analysis for those results. To this end, they conducted a survey in order to determine the frequency at which the users access search engines and how much they trust the content and ranking of search results. They defined several criteria for trustworthiness calculation of search results returned by the search engine, such as topic majority. We adapted their approach for DeFacto and included it as one of the features for our machine learning techniques. Another fact-finding approach is that presented by [[Yin et al., 2007](#)]. Here, the idea is to create a 3-partite network of webpages, facts and objects and apply a propagation algorithm to compute weights for facts as well as webpages. These weights can then be used to determine the degree to which a fact contained in a set of webpages can be trusted. [Pasternack and Roth \[2011a,b\]](#) present a generalized approach for computing the trustworthiness of webpages. To achieve this goal, the authors rely on a graph-based model similar to hubs and authorities [[Kleinberg, 1999](#)]. This model allows computing the

trustworthiness of facts and webpages by generating a k -partite network of pages and facts and propagating trustworthiness information across it. The approach returns a score for the trustworthiness of each fact. Moreover, the generalized fact-finding model that they present allows expressing other fact-finding algorithms such as TruthFinder [Yin et al., 2007], AccuVote [Dong et al., 2009] and 3-Estimates [Galand et al., 2010] within the same framework. The use of trustworthiness and uncertainty information on RDF data has been the subject of recent research (see e.g. [Hartig, 2008; Meiser et al., 2011]). Our approach differs from previous work on fact finding approaches as it focuses on validating the trustworthiness of RDF triples and not that of facts expressed in natural language. In addition, it can deal with the broad spectrum of relations found on the Data Web.

PROVENANCE The problem of data provenance is an issue of central importance for the uptake of the Data Web. While data extracted by the means of tools such as Hazy⁶ and KnowItAll⁷ can be easily mapped to primary provenance information, most knowledge sources were extracted from non-textual source and are more difficult to link with provenance information. Hartig and Zhao [2010] describe a framework for provenance tracking. This framework provides the vocabulary required for representing and accessing provenance information on the Web. It keeps track of metadata including who created a Web entity (e.g. a webpage) and how the entity was modified. Recently, a W3C working group has been formed and released a set of specifications on sharing and representing provenance information.⁸ Dividino et al. [2011] introduced an approach for managing several provenance dimensions, e.g. source and timestamp. In their approach, they describe an extension to the RDF called RDF⁺ which can work efficiently with provenance data. They also provide a method for enabling SPARQL query processors in a manner such that a specific SPARQL query can request meta knowledge without being modified. Theoharis et al. [2011] argue that the implicit provenance data contained in a SPARQL query result can be used to acquire annotations for several dimensions of data quality. They present the concept of abstract provenance models as known from databases and how it can be extended to suit the Data Web as well. Their model requires the existence of provenance data in the underlying semantic data source. DeFacto uses the W3C provenance group standard for representing provenance information. Yet, unlike previous work, it directly tries to find provenance information by searching for confirming facts in trustworthy webpages.

⁶ <http://hazy.cs.wisc.edu/hazy/>

⁷ <http://www.cs.washington.edu/research/knowitall/>

⁸ <http://www.w3.org/2011/prov/wiki/>

TEMPORAL ANALYSIS Storing and managing the temporal validity of facts is a tedious task that has not yet been studied widely in literature. First works from the Semantic Web community in this direction include Temporal RDF [Gutierrez et al., 2005], which allows representing time intervals within which a relation is valid. Extracting such information from structured data is a tedious endeavor for which only a small number of solutions exist. For example, Talukdar et al. [2012b] present an approach for scoping temporal facts which relies on formal constraints between predicates. In particular, they make use of the alignment, containment, succession and mutual exclusion of predicates. Acquiring the constraints that hold between given predicates is studied by [Talukdar et al., 2012a]. Another approach that aims at extracting temporal information is Timely YAGO [Wang et al., 2010], which focuses on extracting temporally scoped facts from Wikipedia infoboxes. PRAVDA [Wang et al., 2011] relies on constrained label propagation to extract temporal information. Here, an objective function which models inclusion constraints and factual information is optimized to determine an assignment of fact to time slots. To the best of our knowledge, none of the previous approaches has dealt with coupling the validity of a fact with its time scope.

RELATION EXTRACTION The verbalization of formal relations is an essential component of DeFacto as it allows searching for RDF triples in unstructured data sources. This verbalization task is strongly related to the area of relation extraction, which aims at detecting formal relations and entity mentions in unstructured data sources. Some early work on relation extraction based on pattern extraction relied on supervised machine learning (see e.g. [Grishman and Yangarber, 1998]). Yet, such approaches demand large amounts of training data, making them difficult to adapt to new relations. The subsequent generation of approaches to RE aimed at bootstrapping patterns based on a small number of input patterns and instances. For example, Brin [1999] present the Dual Iterative Pattern Relation Expansion (DIPRE) and applies it to the detection of relations between authors and titles of books. This approach relies on a small set of seed patterns to maximize the precision of the patterns for a given relation while simultaneously minimizing their error rate. Snowball [Agichtein and Gravano, 2000] extends DIPRE by a new approach to the generation of seed tuples. Other approaches intend to either collect redundancy information (see e.g. [Yan et al., 2009]) in an unsupervised manner or to use linguistic analysis [Nguyen et al., 2007] to harvest generic patterns for relations. The newest approaches to relation extraction make use of ontologies as seed knowledge. While several approaches (including NELL [Carlson et al., 2010] and PROSPERA [Nakashole et al., 2011]) employ their own ontologies, frameworks such as BOA [Gerber and Ngonga Ngomo, 2012], LODifier [Augenstein et al., 2012] and the sys-

tem presented by Krause et al. [2012] reuse information available on the Linked Data Web as training data to discover natural language patterns that express formal relations and reuse those to extract RDF from unstructured data sources.

NAMED ENTITY DISAMBIGUATION NED is most often an a-priori task to RE. In the last years, approaches began relying on RDF data as underlying knowledge bases. *DBpedia Spotlight* [Mendes et al., 2011] is a Named Entity Recognition and Disambiguation combining approach based on DBpedia [Auer et al., 2008]. This approach is able to work on all classes of Named Entities present in the knowledge base also enabling the user to specify coverage and error tolerance during the annotation task. Based on measures like prominence, topical relevance, contextual ambiguity and disambiguation confidence, *DBpedia Spotlight* achieves a disambiguation accuracy of 81% on their Wikipedia corpus. *AIDA* [Hoffart et al., 2011], considered to be the state-of-the-art algorithm for Named Entity Disambiguation, is based on the YAGO⁹ knowledge base. This approach uses dense sub-graphs to identify coherent mentions. Moreover, *AIDA* makes use of contextual similarity, prominence information and context windows. *AGDISTIS* [Usbeck et al., 2014] is a novel knowledge-base agnostic NED approach which combines an authority-based graph algorithm and different label expansion strategies and string similarity measures. Based on this combination, the approach can efficiently detect the correct URIs for a given set of named entities within an input text. The results indicate that *AGDISTIS* is able to outperform the state-of-the-art approaches by up to 16% F-measure.

5.3 APPROACH

The following section describes the basic system architecture for the DeFacto approach. Since both the baseline DeFacto and the extended DeFacto system architecture are mostly identical, we combine their descriptions and highlight each difference.

INPUT AND OUTPUT The DeFacto system consists of the components depicted in Figure 20. The system supports two types of inputs: RDF triples and textual data. If provided with a fact represented as an RDF triple, DeFacto returns a confidence value for this fact as well as possible evidence for it. In the case of textual data, e.g. from an input form, DeFacto uses *AGDISTIS* to disambiguate the entities and gathers surface forms (see Section 5.5.1) for each resource.¹⁰ The evidence consists of a set of webpages, textual excerpts from those pages and meta-information on the pages. The text excerpts and the associated

The baseline version did only accept RDF triples as input.

⁹ <http://www.mpi-inf.mpg.de/yago-naga/yago/>

¹⁰ Note the disambiguation of the property URI of the fact is out of scope of this section.

meta information enable the user to quickly get an overview of possible credible sources for the input statement. Instead of having to use search engines, browsing several webpages and looking for relevant pieces of information, the user can thus more efficiently review the presented information. Moreover, the system uses techniques which are adapted specifically for fact validation instead of only having to rely on generic information retrieval techniques of search engines.

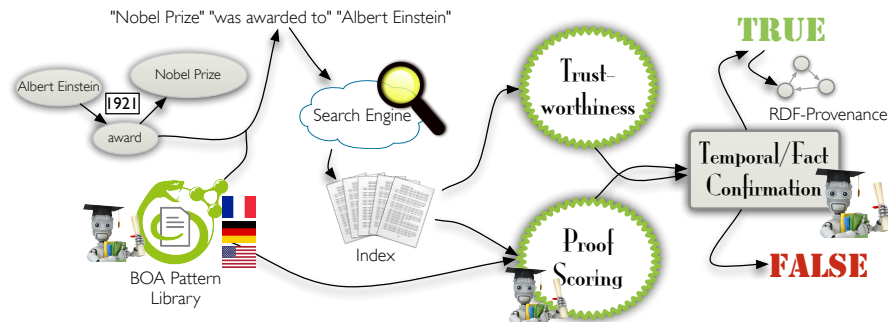


Figure 20: Overview of the DeFacto architecture.

RETRIEVING WEBPAGES The first step of the DeFacto fact validation process is to retrieve webpages which are relevant for the given task. The retrieval is carried out by issuing several queries to a regular search engine. These queries are computed by verbalizing the fact using multilingual natural language patterns extracted by the BOA framework¹¹ [Gerber and Ngonga Ngomo, 2011, 2012]. Section 5.5.2 describes how the search engine queries are constructed. As a next step, the highest ranked webpages for each query are retrieved. Those webpages are candidates for being sources for the input fact. Both the search engine queries as well as the retrieval of webpages are executed in parallel to keep the response time for users within a reasonable limit. Note that usually this does not put a high load on particular web servers as webpages are usually derived from several domains.

EVALUATING WEBPAGES Once all webpages have been retrieved, they undergo several further processing steps. First, plain text is extracted from each webpage by removing most HTML markup. We can then apply our fact confirmation approach on this text, which is described in detail in Section 5.4.1 for the baseline version and in Section 5.5.3 for the multilingual extension. In essence, the algorithm decides whether the webpage contains a natural language formulation of the input fact. This step distinguishes DeFacto from information retrieval methods. If no webpage confirms a fact according to DeFacto, then the system falls back on lightweight NLP techniques

Note that the baseline version did not feature multilingualism nor temporal analysis.

¹¹ <http://boa.aksw.org>

and computes whether the webpage does at least provide useful evidence. In addition to fact confirmation, the system computes different indicators for the trustworthiness of a webpage (see Section 5.4.2 and 5.5.4). These indicators are of central importance because a single trustworthy webpage confirming a fact may be a more useful source than several webpages with low trustworthiness. The fact confirmation and the trustworthiness indicators of the most relevant webpages are presented to the user.

CONFIDENCE MEASUREMENT In addition to finding and displaying useful sources, DeFacto also outputs a general confidence value for the input fact. This confidence value ranges between 0% and 100% and serves as an indicator for the user: Higher values indicate that the found sources appear to confirm the fact and can be trusted. Low values mean that not much evidence for the fact could be found on the Web and that the webpages that do confirm the fact (if such exist) only display low trustworthiness. The confidence measurement is based on machine learning techniques and explained in detail in Sections 5.4.3 and 5.5.5. Naturally, DeFacto is a (semi-)automatic approach: We do assume that users will not blindly trust the system, but additionally analyze the provided evidence.

RDF PROVENANCE OUTPUT

Besides a visual representation of the fact and its most relevant webpages, it is also possible to export this information as RDF, which enables a Linked Data style access or the management in a SPARQL endpoint. We reuse several existing vocabularies for modeling the provenance of the DeFacto output (see 21), especially the PROV Ontology [Belhajjame et al., 2012], which provides a set of classes, properties, and restrictions that can be used to represent and interchange provenance information generated in different systems and under different contexts, and the NLP Interchange Format (NIF) [Hellmann et al., 2013], which is an RDF/OWL-based format that is meant to achieve interoperability between Natural Language Processing (NLP) tools, language resources and annotations. A RDF dump of the generated evidences for the correct facts of FactBench (see Section 5.5.7) can be downloaded from the project home page¹².

The first version of DeFacto did not incorporate the NIF vocabulary.

USING THE LOD CLOUD AS BACKGROUND KNOWLEDGE: As described above, DeFacto relies primarily on natural language from several webpages as input. However, in some cases, confirming facts for an input statement can be found in openly available knowledge bases. Due to the fast growth of the LOD cloud, we expect this source to become increasingly important in the future. In order to use this additional evidence, DeFacto provides a preliminary component which

¹² <http://aksw.org/Projects/DeFacto>

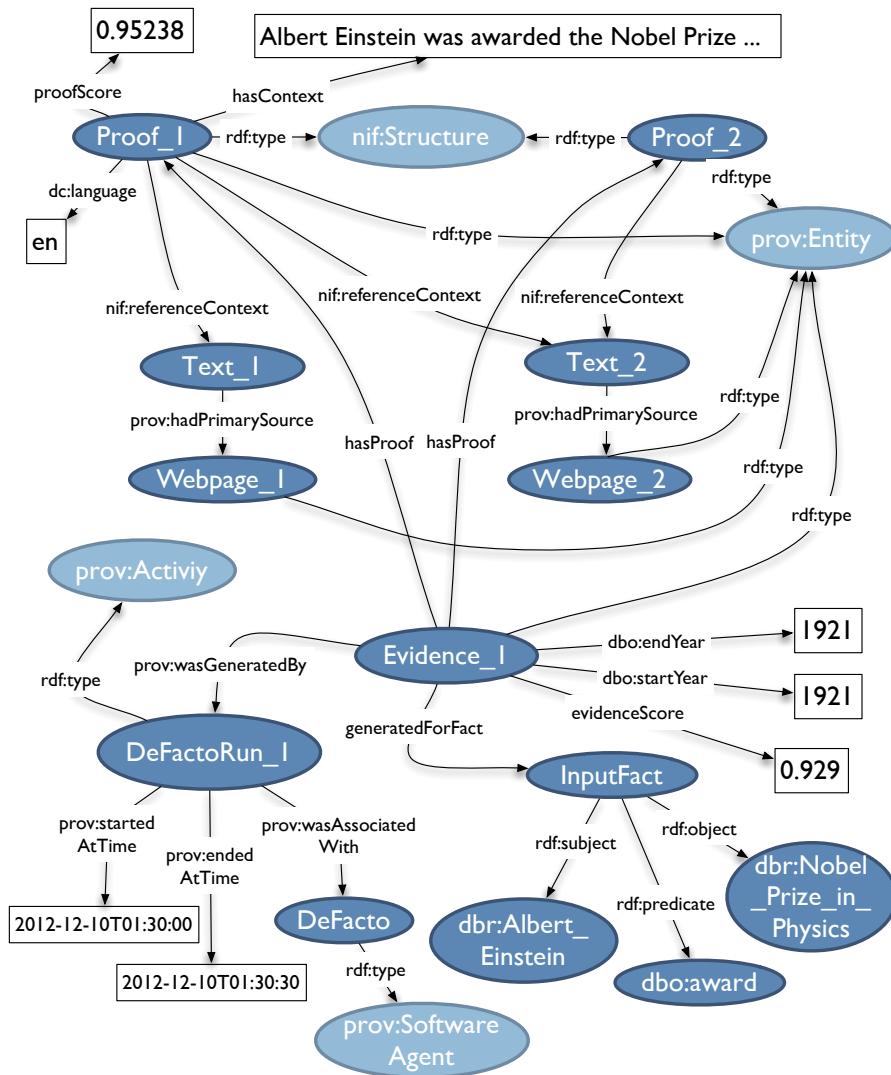


Figure 21: Overview of the provenance schema which is used to export the validation result of DeFacto as RDF, given the input fact Albert Einstein, award, Nobel Prize in Physics.

searches for similar statements in the LOD cloud. To achieve this goal, the system first finds similar resources to the subject and object of the input triple, which is currently done via the <http://sameas.org> service. In a second step, it retrieves all triples which use the detected similar subject and object resources by dereferencing the corresponding Linked Data URIs. Finally, the labels of subject, predicate and object of all triples are retrieved. Those are then compared via string similarity techniques to the input triple. Currently, the average trigram similarity of subject, predicate and object of the triple is used. In this article, we focus on re-using textual evidence and plan to carry out a more detailed evaluation of the LOD as background knowledge in future work.

DeFacto

Deep Fact Validation

Enter the fact:

Subject	Predicate	Object	
Albert Einstein	award	Nobel Prize in Physics	Validate

Overall DeFacto Score: 90%
This fact holds for the year 1921 [Export as RDF](#)

72 websites containing the fact:
[30 | janvier | 2010 | Elleenparle's Blog](#)

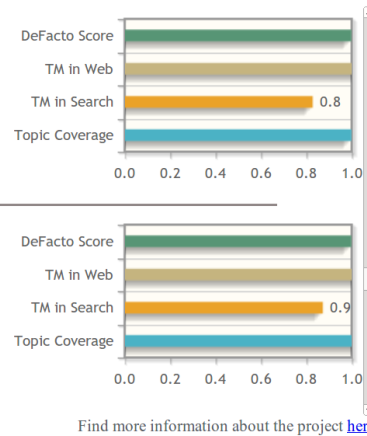
Proof(s):

- itation, qu'il a reçu le prix Nobel de physique en 1921 et mathématiquement connu sous l'équation $E=mc^2$. Alors ? **Albert Einstein** biensûr ! Conclusions. N
- itation, qu'il a reçu le prix Nobel de physique en 1921 et mathématiquement connu sous l'équation $E=mc^2$. Alors ? **Albert Einstein** biensûr ! Concl

[Albert Einstein awarded Nobel Prize in Physics ...](#)

Proof(s):

- Albert Einstein** awarded **Nobel Prize in Physics** Topics: Albert Einstein, Nobel Prize, Nobel Prize In Physics Share In 1922, Einstein was awarded the 1921 Nob
- Albert Einstein** awarded **Nobel Prize in Physics** - WorldHistoryProject.or
- n awarded **Nobel Prize in Physics** Topics: Albert Einstein, Nobel Prize, Nobel Prize In Physics Share In 1922, Einstein was



Find more information about the project [here](#)

Figure 22: Screenshot of the extended DeFacto Web interface.

DEFAC TO WEB DEMO

A prototype implementing the above steps is available at <http://defacto.aksw.org>. A screenshot of the user interface is depicted in Figure 22. It shows relevant webpages, text excerpts and four different rankings per page. In this case many possible sources for the statement were found, which is why DeFacto displays a high confidence score. Note that since the current version of DeFacto implements a temporal fact scoping module, we also display the year, e.g. 1921, in which the fact is or was true. The generated provenance output can also be saved directly as RDF. The source code of both the DeFacto algorithms and user interface are openly available.¹³

It should be noted that we decided not to check for negative evidence of facts in DeFacto, since a) we considered this to be too error-prone and b) negative statements are much less frequent on the Web.

The prototype is an extension of the prototype presented by Lehmann et al. [2012b].

¹³ <https://github.com/AKSW/DeFacto>

5.4 DEFACTO – DEEP FACT VALIDATION

This section describes the DeFacto baseline system as presented by [Lehmann et al. \[2012b\]](#). In particular we present the integration of the BOA framework and its training for the fact validation use case as well as the features used for fact validation and trustworthiness analysis of webpages. These features can be considered as the core of the baseline version and the extended multilingual and temporal extension. Finally we introduce a method to automatically create test sets, containing both true and false RDF triples, which we use to train and evaluate the DeFacto machine learning components.

```
1 dbpedia-res:Jamaica_Inn_%28film%29 dbpedia-owl:director
2 dbpedia-res:Alfred_Hitchcock .
```

Listing 9: Input data for Defacto.

5.4.1 BOA

The idea behind BOA is twofold: first, it is meant to be a framework that allows extracting structured data from the Human Web by using Linked Data as background knowledge. Second, it provides a library of natural language patterns that allows to bridge the gap between structured and unstructured data. The input for the BOA framework consists of a set of knowledge bases, a text corpus (mostly extracted from the Web) and (optionally) a Wikipedia dump¹⁴. When provided with a Wikipedia dump, the framework begins by generating surface forms for all entities in the source knowledge base. The surface forms used by BOA are generated by using an extension of the method proposed in [Mendes et al. \[2011\]](#). For each predicate p found in the input knowledge sources, BOA carries out a sentence-level statistical analysis of the co-occurrence of label pairs for resources that are linked via p . BOA then uses a supervised machine-learning approach to compute a score and rank patterns for each combination of corpus and knowledge bases. These patterns allow generating a natural language representation of the RDF triple that is to be checked.

5.4.1.1 Training BOA for DeFacto

In order to provide a high quality fact confirmation component, we trained BOA specifically for this task. We began by selecting the top 60 most frequently used object properties from the DBpedia [[Morsey et al., 2012](#); [Lehmann et al., 2009](#)] ontology using the DBpedia Live endpoint¹⁵. This query retrieves 7,750,362 triples and covers 78% of the 9,993,333 triples in DBpedia with *owl:ObjectProperty*s from the DB-

¹⁴ <http://dumps.wikimedia.org/>

¹⁵ <http://live.dbpedia.org/sparql>

pedia namespace.¹⁶ Currently, we focus on object properties. The adequate support of datatype properties requires an extension of the presented methods, which is planned in future work. For each of those properties, we selected the top 10 BOA patterns (if available) sorted according to the number of triples a pattern has been learned from. This resulted in a list of 488 patterns which were evaluated by all four authors. During this process, each pattern was labeled by two persons independently. We judged a pattern as positive if it was not generic (e.g. “D ‘s ’ D”) or specific enough (e.g. “D in the Italian region D”) and could be used to express the relation in natural text. The first group achieved a moderate Cohen’s-Kappa value of 0.477 and the second group scored a good value of 0.626. Every conflict was resolved by having the annotators agree on a single annotation. The resulting annotations were used for a 10-fold cross-validation training of BOA’s neural network. We achieved the maximum F-score of 0.732 with an error threshold of 0.01 and a hidden layer size of 51 neurons.

5.4.1.2 Automatic Generation of Search Queries

The found BOA patterns are used for issuing queries to the search engine (see Figure 20). Each search query contains the quoted label of the subject of the input triple, a quoted and cleaned BOA pattern (we remove punctuation characters which are not indexed by the search engine) and the quoted label of the object of the input triple. We use a fixed number of the best-scored BOA patterns whose score was beyond a score threshold and retrieve the first n websites from a web search engine. For our example from Listing 9, an exemplary query sent to the search engine is “Jamaican Inn” AND “written and directed by” AND “Alfred Hitchcock”. We then crawl each website and try to extract *possible proofs* of the input triple, i.e. excerpts of these webpages which may confirm it. For the sake of brevity, we use proof and possible proof interchangeably.

5.4.1.3 BOA and NLP Techniques for Fact Confirmation

To find proofs of a given input triple $t = (s, p, o)$ we make use of the surface forms introduced by Mendes et al. [2011]. We select all surface forms for the subject s and object o of the input triple and search for all occurrences of each combination of those labels in a website w . If we find an occurrence with a token distance $\text{dist}(\text{lab}(s), \text{lab}(o))$ (where $\text{lab}(x)$ is the label of x) smaller than a given threshold we call this occurrence a proof of the input triple. To remove noise from the found proofs we apply a set of normalizations by using regular expression filters which for example remove characters between brackets and non alpha-numeric characters. Note that this normalization

¹⁶ Properties like `wikiPageExternalLink`, `wikiPageRedirects`, `wikiPageDisambiguates` and `thumbnail` have been excluded.

improves the grouping of proofs by their occurrence. After extracting all proofs $pr_i \in \mathcal{P}\nabla(w)$ of a webpage w , we score each proof using a linear regression classifier. We trained a classifier with the following input features for scoring a proof:

- BOA PATTERN:** This is a boolean feature which is 1 if a BOA pattern is contained in the normalized proof phrase.
- BOA SCORE:** If BOA patterns are found in the normalized proof phrase, then the score of the highest score across the set of found patterns is written in this feature. Else, this feature is set to 0.
- TOKEN DISTANCE:** This is the distance $\text{dist}(\text{lab}(s), \text{lab}(o))$ between the two entity labels which found the proof. We limit this distance to a maximum of 20 tokens.
- WORDNET EXPANSION:** We expand both the tokens of the normalized proof phrase as well as all of the tokens of the BOA pattern with synsets from Wordnet. Subsequently we apply the Jaccard-Similarity on the generated expansions. This is basically a fuzzy match between the BOA pattern and the proof phrase.
- TOTAL OCCURRENCE:** This feature contains the total number of occurrences of each normalized proof phrase over the set of all normalized proof phrases.
- PAGE TITLE:** We apply the maximum of the trigram similarity measure between the page title and the subject and object labels. This feature is useful, because the title indicates the topic of the entire webpage. When a title matches, then higher token distances may still indicate a high probability that a fact is confirmed.
- END OF SENTENCE:** A boolean value if the potential proof contains a “.”, “!” or a “?”. When subject and object are in different sentences, their relation is more likely to be weaker.
- PHRASE:** The words between the subject and object, which are encoded as binary values, i.e. a feature is created for each word and its value is set to 1 if the word occurs and 0 otherwise.
- PROPERTY:** The property as a word vector.

To train our classifiers, we randomly sampled 527 proofs and annotated them manually. Those proofs were extracted with DeFacto by applying it on the training set described in Section 5.4.4.1. The results are shown in Table 19. We ran popular classifiers, which are able to work with numeric data and create confidence values. The ability to generate confidence values for proofs is useful as feedback for users

	P	R	F ₁	AUC	RMSE
Logistic Regression	0.769	0.769	0.769	0.811	0.4653
Naïve Bayes	0.655	0.624	0.564	0.763	0.5665
SVM	0.824	0.822	0.822	0.823	0.4223
RBFNetwork	0.735	0.717	0.718	0.718	0.485

Table 19: Performance measures for several classifiers on the fact confirmation task (AUC = area under the ROC curve, RMSE = root mean squared error).

and it also serves as input for the core classifiers described in Section 5.4.4. Based on the obtained results, we selected support vector machines as classifier. We also performed preliminary work on fine-tuning the parameters of the above algorithms, which, however, did not lead to significantly different results. The reported measurements were, therefore, done with default values of the mentioned algorithms in the Weka machine learning toolkit¹⁷ version 3.6.6.

5.4.2 Trustworthiness Analysis of Webpages

To identify the trustworthiness of a webpage we first need to determine its similarity to the input triple. This is determined by how many topics belonging to the query are contained in a search result retrieved by the web search. We extended the approach introduced by Nakamura et al. [2007] by querying Wikipedia with the subject and object label of the triple in question separately to find the topic terms for the triple. Please note that through the availability of multilingual labels for many resources in the LOD cloud, we are able to extract topic terms in multiple languages. A frequency analysis is applied on all returned documents and all terms above a certain threshold that are not contained in a self-compiled stop word list are considered to be topic terms for a triple. Let s and o be the URIs for the subject and object of the triple in question, τ be a potential topic term extracted from a Wikipedia page and let $t = (s, p, o)$ be the input triple. We compare the values of the following two formulas:

$$\text{prob}(\tau|t) = \frac{|\text{topic}(\tau, \text{docs}(t))|}{|\text{docs}(t)|} \quad (18)$$

$$\text{prob}(t|\text{intitle}(\text{docs}(t), s \vee o)) = \frac{|\text{topic}(\tau, \text{intitle}(\text{docs}(t), s) \cup \text{intitle}(\text{docs}(t), o))|}{|\text{intitle}(\text{docs}(t), s) \cup \text{intitle}(\text{docs}(t), o)|} \quad (19)$$

¹⁷ <http://www.cs.waikato.ac.nz/ml/weka/>

where $\text{docs}(t)$ is the set of all web documents retrieved for t (see Section 5.4.1), $\text{intitle}(\text{docs}(t), x)$ the set of web documents which have the label of the URI x in their page title. $\text{topic}(\tau, \text{docs}(t))$ is a function returning the set of documents which contain τ in the page body. We consider τ to be a topic term for the input triple if $\text{prob}(\tau|\tau(\text{docs}(t), s) \vee \tau(\text{docs}(t), o)) > \text{prob}(\tau|t)$. Let $\mathfrak{T}_t = \{\tau_1, \tau_2, \dots, \tau_n\}$ be the set of all topic terms extracted for an input triple. DeFacto then calculates the trustworthiness of a webpage as follows:

TOPIC MAJORITY ON THE WEB This represents the number of webpages that have similar topics to the webpage in question. Let \mathfrak{T}_w be the set of topic terms appearing on the current webpage w . The Topic Majority on the Web for a webpage w is then calculated as:

$$\text{tm}_{\text{web}}(w) = \left| \bigcup_{i=1}^n \text{topic}(\tau_i, d(X)) \right| - 1. \quad (20)$$

where τ_1 is the most occurring topic term in the webpage w . Note that we subtract 1 to prevent counting w .

TOPIC MAJORITY IN SEARCH RESULTS It is used to calculate the similarity of a given webpage for all webpages found for a given triple. Let w_k be the webpage to be evaluated, $v(w_k)$ be the feature vector of webpage w_k where $v(w_k)_i$ is 1 if τ_i is a topic term of webpage w_k and 0 otherwise, $\|v\|$ be the norm of v and θ a similarity threshold. We calculate the Topic Majority for the search results as follows:

$$\text{tm}_{\text{sr}}(w) = \left| \left\{ w_i | w_i \in d(X), \frac{v(w_k) \times v(w_i)}{\|v(w_k)\| \|v(w_i)\|} > \theta \right\} \right|. \quad (21)$$

TOPIC COVERAGE This measures the ratio between all topic terms for t and all topic terms occurring in w :

$$\text{tc}(w) = \frac{|\mathfrak{T}_t \cap \mathfrak{T}_w|}{|\mathfrak{T}_t|}. \quad (22)$$

PAGERANK: The Pagerank¹⁸ of a webpage is a measure for the relative importance of a webpage compared to all others, i.e. higher pageranks mean that a webpage is more popular. There is a positive correlation between popularity of a webpage and its trustworthiness as those pages are more likely to be reviewed by more people or may have undergone stricter quality assurance before their publication. While a high pagerank alone is certainly not a sufficient indicator for trustworthiness, we use it in combination with the above criteria in DeFacto.

¹⁸ <http://en.wikipedia.org/wiki/Pagerank>

5.4.3 Features for Deep Fact Validation

In order to obtain an estimate of the confidence that there is sufficient evidence to consider the input triple to be true, we decided to train a supervised machine learning algorithm. Similar to the above presented classifier for fact confirmation, this classifier also requires computing a set of relevant features for the given task. In the following, we describe those features and why we selected them.

First, we extend the score of single proofs to a score of webpages as follows: When interpreting the score of a proof as the probability that a proof actually confirms the input fact, then we can compute the probability that at least one of the proofs confirms the fact. This leads to the following stochastic formula¹⁹, which allows us to obtain an overall score for proofs scw on a webpage w :

$$scw(w) = 1 - \prod_{pr \in prw(w)} (1 - fc(pr)). \quad (23)$$

In this formula, fc (fact confirmation) is the classifier trained in Section 5.4.1.3, which takes a proof pr as input and returns a value between 0 and 1. prw is a function taking a webpage as input and returning all possible proofs contained in it.

COMBINATION OF TRUSTWORTHINESS AND TEXTUAL EVIDENCE

In general, the trustworthiness of a webpage and the textual evidence we find in it are orthogonal features. Naturally, webpages with high trustworthiness and a high score for its proofs should increase our confidence in the input fact. Therefore, it makes sense to combine trustworthiness and textual evidence as features for the underlying machine learning algorithm. We do this by multiplying both criteria and then using their sum and maximum as two different features:

$$F_{fsum}(t) = \sum_{w \in s(t)} (f(w) \cdot scw(w)) \quad (24)$$

$$F_{fmax}(t) = \max_{w \in s(t)} (f(w) \cdot scw(w)) \quad (25)$$

In this formula f can be instantiated by all four trustworthiness measures: topic majority on the the web (tm_{web}), topic majority in search results (tm_{sr}), topic coverage (tc) and pagerank (pr). s is a function taking a triple t as argument, executing the search queries explained in Section 5.4.1.2 and returning a set of webpages. Using the formula, we obtain 8 different features for our classifier, which combine textual evidence and different trustworthiness measures.

¹⁹ To be exact, it is the complementary even to the case that none of the proofs do actually confirm a fact.

OTHER FEATURES In addition to the above described combinations of trustworthiness and fact confirmation, we also defined other features:

1. The total number of proofs found.
2. The total number of proofs found above a relevance threshold of 0.5. In some cases, a high number of proofs with low scores is generated, so the number of high scoring proofs may be a relevant feature for learning algorithms.
3. The total evidence score: This is the probability that at least one of the proofs is correct, which is defined analogously to scw above:

$$1 - \prod_{pr \in \text{prt}(t)} (1 - \text{fc}(pr)). \quad (26)$$

where $\text{prt}(t)$ is a function returning all proofs found for t from all webpages.

4. The total evidence score above a relevance threshold of 0.5. This is an adaption of the above formula, which considers only proofs with a confidence higher than 0.5.
5. Total hit count: Search engines usually estimate the number of search results for an input query. The total hit count is the sum of the estimated number of search results for each query send by DeFacto for a given input triple.
6. A domain and range verification: If the subject of the input triple is not an instance of the domain of the property of the input triple, this violates the underlying schema, which should result in a lower confidence in the correctness of the triple. This feature is 0 if both domain and range are violated, 0.5 if exactly one of them is violated and 1 if there is no domain or range violation. At the moment, we are only checking whether the instance is asserted to be an instance of a class (or one of its subclasses) and do not use reasoning for performance reasons.

5.4.4 *Evaluation*

Our main objective in the evaluation was to find out whether DeFacto can effectively distinguish between true and false input facts. In the following, we describe how we trained DeFacto using DBpedia, introduce the experiments we used and discuss the results of our experiments.

5.4.4.1 Training DeFacto

As mentioned in Section 5.4.1, we focus our experiments on the top 60 most frequently used properties in DBpedia. The system can easily be extended to cover more properties by extending the training set of BOA to those properties. Note that DeFacto itself is also not limited to DBpedia, i.e. while all of its components are trained on DBpedia, the algorithms can be applied to arbitrary URIs. A performance evaluation on other knowledge bases is subject to future work, but it should be noted that most parts of DeFacto – except the LOD background feature described in Section 5.3 and the schema checking feature in Section 5.4.3 work only with the retrieved labels of URIs and, therefore, do not depend on DBpedia.

For training a supervised machine learning approach, positive and negative examples are required. Those were generated as follows:

POSITIVE EXAMPLES: In general, we use facts contained in DBpedia as positive examples. For each of the properties we consider (see Section 5.4.1), we generated positive examples by randomly selecting triples containing the property. We collected 600 statements in this manner and verified them by checking manually whether it was indeed a true fact. It turned out that some of the obtained triples were modeled incorrectly, e.g. obviously violated domain and range restrictions or could not be confirmed by an intensive search on the web within ten minutes. Overall, 473 out of 600 checked triples could be used as positive examples.

NEGATIVE EXAMPLES: The generation of negative examples is more involved than the generation of positive examples. In order to effectively train DeFacto, we considered it essential that many of the negative examples are similar to true statements. In particular, most statements should be meaningful triples. For this reason, we derived the negative examples from positive examples by modifying them while still following domain and range restrictions. Assume the input triple (s, p, o) in a knowledge base \mathfrak{K} is given and let dom and ran be functions returning the domain and range of a property²⁰. We used the following methods to generate the negative example sets dubbed *subject*, *object*, *subject-object*, *property*, *random*, *20%mix* (in that order):

1. A triple (s', p, o) is generated where s' is an instance of $\text{dom}(p)$, the triple (s', p, o) is not contained in \mathfrak{K} and s' is randomly selected from all resources which satisfy the previous requirements.
2. A triple (s, p, o') is generated analogously by taking $\text{ran}(p)$ into account.

²⁰ Technically, we used the most specific class, which was explicitly stated to be domain and range of a property, respectively.

3. A triple (s', p, o') is generated analogously by taking both $\text{dom}(p)$ and $\text{ran}(p)$ into account.
4. A triple (s, p', o) is generated in which p' is randomly selected from our previously defined list of 60 properties and (s, p', o) is not contained in \mathfrak{R} .
5. A triple (s', p', o') is generated where s' and o' are randomly selected resources, p' is a randomly selected property from our defined list of 60 properties and (s', p', o') is not contained in \mathfrak{R} .
6. 20% of each of the above created negative training sets were randomly selected to create a heterogenous test set.

Note that all parts of the example generation procedure can also take implicit knowledge into account, e.g. by simply extending our approach to use SPARQL 1.1 entailment²¹. In case of DBpedia Live we did not use any entailment for performance reasons and because it would not alter the results in that specific case.

Obviously, it is possible that our procedure for generating negative examples also generates true statements which just happen not to be contained in DBpedia. Similar to the analysis of the positive examples, we checked a sample of the negative examples on whether they are indeed false statements. This was the case for all examples in the sample. Overall, we obtained an automatically created and manually cleaned training set, which we made publicly available²².

5.4.4.2 Experimental Setup

In a first step, we computed all feature vectors, described in Section 5.4.3 for the training set. DeFacto relies heavily on web requests, which are not deterministic (i.e. the same search engine query does not always return the same result). To achieve deterministic behavior and to increase the performance as well as reduce load on the servers, all web requests are cached. The DeFacto runtime for an input triple was on average slightly below 5 seconds per input triple²³ when using caches.

We stored the features in the arff file format and employed the Weka machine learning toolkit²⁴ for training different classifiers. In particular, we were interested in classifiers which can handle numeric values and output confidence values. Naturally, confidence values for facts such as, e.g. 95%, are more useful for end users than just a binary response on whether DeFacto considers the input triple to be true,

²¹ <http://www.w3.org/TR/sparql11-entailment/>

²² <http://aksw.org/projects/DeFacto>

²³ The performance is roughly equal on server machines and notebooks, since the web requests dominate.

²⁴ <http://www.cs.waikato.ac.nz/ml/weka/>

since they allow a more fine-grained assessment. Again, we selected popular machine-learning algorithms satisfying those requirements.

We performed 10-fold cross-validations for our experiments. In each experiment, we used our created positive examples, but varied the negative example sets described above to see how changes influence the overall behavior of DeFacto.

	Subject				
	P	R	F ₁	AUC	RMSE
Logistic Regression	0.799	0.753	0.743	0.83	0.4151
Naïve Bayes	0.739	0.606	0.542	0.64	0.6255
SVM	0.811	0.788	0.784	0.788	0.4609
J48	0.835	0.827	0.826	0.819	0.3719
RBF Network	0.743	0.631	0.583	0.652	0.469
	Object				
	P	R	F ₁	AUC	RMSE
Logistic Regression	0.881	0.86	0.859	0.844	0.3454
Naïve Bayes	0.795	0.662	0.619	0.741	0.5815
SVM	0.884	0.867	0.865	0.866	0.3409
J48	0.869	0.862	0.861	0.908	0.3194
RBF Network	0.784	0.683	0.652	0.75	0.4421
	Subject-Object				
	P	R	F ₁	AUC	RMSE
Logistic Regression	0.871	0.85	0.848	0.86	0.3495
Naïve Bayes	0.813	0.735	0.717	0.785	0.5151
SVM	0.88	0.863	0.861	0.855	0.3434
J48	0.884	0.871	0.87	0.901	0.3197
RBF Network	0.745	0.687	0.667	0.728	0.4401
	Property				
	P	R	F ₁	AUC	RMSE
Logistic Regression	0.822	0.818	0.818	0.838	0.3792
Naïve Bayes	0.697	0.582	0.511	0.76	0.6431
SVM	0.819	0.816	0.816	0.825	0.3813
J48	0.834	0.832	0.832	0.828	0.3753
RBF Network	0.72	0.697	0.688	0.731	0.4545
	Random				
	P	R	F ₁	AUC	RMSE
Logistic Regression	0.855	0.854	0.854	0.908	0.3417

Naïve Bayes	0.735	0.606	0.544	0.853	0.5565
SVM	0.855	0.854	0.854	0.906	0.3462
J48	0.876	0.876	0.876	0.904	0.3226
RBF Network	0.746	0.743	0.742	0.819	0.4156
			Mix		
	P	R	F ₁	AUC	RMSE
Logistic Regression	0.665	0.645	0.634	0.785	0.4516
Naïve Bayes	0.719	0.6	0.538	0.658	0.6267
SVM	0.734	0.729	0.728	0.768	0.4524
J48	0.8	0.79	0.788	0.782	0.405
RBF Network	0.698	0.61	0.561	0.652	0.4788

Table 20: Classification results for the different evaluation sets.

5.4.4.3 Results and Discussion

The results of our experiments are shown in Table 20. Three algorithms – J48, logistic regression and support vector machines – show promising results. Given the challenging tasks, F-measures up to 78.8% for the combined negative example set appear to be very positive indicators that DeFacto can be used to effectively distinguish between true and false statements, which was our primary evaluation objective. In general, DeFacto also appears to be stable against the various negative example sets. In particular, the algorithms with overall positive results also seem less affected by the different variations. When observing single runs of DeFacto manually, it turned out that our method of generating positive examples is particularly challenging for DeFacto: For many of the facts in DBpedia only few sources exist on the Web. While it is widely acknowledged that the amount of unstructured textual information on the Web by far surpasses the available structured data, we found out that a significant amount of statements in DBpedia is difficult to track back to reliable external sources on the Web even with an exhaustive manual search. There are many reasons for this, for instance many facts are particularly relevant for a specific country, such as “Person x studied at University y.”, where x is a son of a local politician and y is a country with only limited internet access compared to first world countries. For this reason, BOA patterns could only be detected directly in 29 of the 527 proofs of positive examples. This number increased to 195 out of 527 when we employed the WordNet expansion described in Section 5.4.1.3. In general, DeFacto performs better when the subject and object of the input triple are popular on the Web, i.e. there are several webpages describing them. In this aspect, we believe our training set is indeed challenging upon manual observation.

5.5 DEFACTO – MULTILINGUAL AND TEMPORAL EXTENSION

One observation of the DeFacto baseline evaluation was that for some facts the recall of found webpages containing the fact was very low. This was due to the fact that most facts are language or country specific respectively. For example, consider the sports domain with particular sports such as american football in the USA and soccer and Germany. Both sports have high publicity values/recall in the national press but soccer fails to achieve high recall in the US and american football in Germany. To validate facts about German soccer players and clubs it is therefore necessary to use German natural language patterns for relations like `playsForClub`. Furthermore, an intelligent fact validation approach can not disregard the temporal scope of facts. For example, the fact `<Franz Beckenbauer> <plays for club> <New York Cosmos>` has been true from 1977 to 1980 but is no longer valid now.

In this section we present the extension of the DeFacto baseline version which is able to handle multilingual texts and temporally scope facts. We begin by illustrating the changes applied to the BOA framework, present new features for fact validation and trustworthiness analysis for webpages as well as a method to temporally scope facts in their time period or time point in which they hold true. Finally we introduce FactBench, a multilingual and temporally scoped benchmark, and use it to evaluate this extension.

5.5.1 *Training BOA for DeFacto*

In order to provide a high quality fact confirmation component, we trained BOA specifically for this task. We began by selecting the relations used in FactBench (see Section 5.5.7) and queried the instance knowledge from DBpedia 3.9 [Lehmann et al., 2013, 2009]. Since first experiments showed that relying on the localized version of DBpedia would result in poor recall, we translated the English background knowledge to German and French respectively. This is carried out by replacing English `rdfs:labels` with localized ones if such exist. If no target language label exists we rely on the English label as backup. We then ran BOA on the July 2013 dumps of the corresponding Wikipedias. Since search engine queries are expensive we ordered the generated patterns by their support set size, the subject and object pairs the patterns were found from, and used the top-n patterns for each relation to formulate search engine queries. We chose not to train BOA's machine learning module, since this would have resulted in high-precision but low-recall patterns. Additionally, we implemented a pattern generalization approach to better cope with similar but low-recall patterns.

LEXICAL PATTERN GENERALIZATION FOR DEFACTO A drawback of the previous version of the BOA framework was that it could not detect similar patterns. For example consider the following two English patterns: “?R ’s Indian subsidiary ?D” and “?R ’s UK subsidiary , ?D”. Both patterns are NLRs for the *dbo:subsidiary* relation but might fail to score high confidence scores because of their individual low number of occurrences. Generalizing these patterns into “?R ’s NE subsidiary ?D” can therefore help boost pattern scores for low recall patterns. We generalize patterns individually for each language based on manually crafted regular expressions and Part-of-speech tags provided by a language-specific tagger. In the current version of DeFacto, we generalize personal pronouns, named entities, date/year occurrences and forms of “be” as well as numerical values.

5.5.2 Automatic Generation of Search Queries

The found BOA patterns are used for issuing queries to the search engine (see Figure 20). Each search query contains the quoted label (forces an exact match from the search engine) of the subject of the input triple, a quoted and cleaned BOA pattern (i.e. without punctuation) and the quoted label of the object of the input triple. Note that we can fully localize the search query in most cases since there are multilingual labels for many resources available on the LOD cloud. We use the top-k best-scored BOA patterns and retrieve the first n webpages from a Web search engine. For our example from Listing 10, an exemplary query sent to the search engine is “Albert Einstein” AND “was awarded the” AND “Nobel Prize in Physics”. We then crawl each webpage, remove HTML markup and try to extract *possible proofs* for the input triple, i.e. excerpts of these webpages which may confirm it. For the sake of brevity, we use proof and possible proof interchangeably.

5.5.3 BOA and NLP Techniques for Fact Confirmation

The extension of DeFacto uses the same approach for fact confirmation as the one presented for the baseline system in Section 5.4.1.3. Again we search for all combinations of co-occurrences of two surface forms. Note that we now search for co-occurrences for all configured languages. We then perform a noise removal and normalization step and score each proof using a support vector machine. We trained the classifier with the previously introduced features (see Section 5.4.1.3) and the following additional or updated features:

STRING SIMILARITY For the top-n BOA patterns of the given relation we determine the maximum string similarity between the normalized pattern and the proof phrase. As string similarity

	en _{20%}	en _{100%}	de _{20%}	de _{100%}	fr _{20%}	fr _{100%}
True	12414	79921	4419	29292	5724	36383
False	11705	17436	5488	8263	5231	7721
Total	24119	97357	9907	37555	10955	44104

Table 21: Proofs with language distribution used to train fact classifier.

we use Levenshtein, Smith-Waterman as well as the QGram Similarity²⁵.

WORDNET EXPANSION: We expand both the tokens of the normalized proof phrase as well as all of the tokens of the BOA pattern with synsets from Wordnet. Subsequently we apply the Jaccard-Similarity on the generated expansions. This is basically a fuzzy match between the BOA pattern and the proof phrase. Due to the language specificity of Wordnet to English, we will use BabelNet [Navigli and Ponzetto, 2012] in future iterations.

SYNTAX: We also calculate a number of numeric features for the proof phrases: the number of non-alpha-numeric and uppercase characters, the number of commas, digits and characters and the average token length.

PROPERTY: The property as a word vector.

LANGUAGE: The language of the webpage.

To train our classifier, we ran DeFacto on the *mix* train set (see Section 5.5.7) and extracted all proof phrases. We randomly sampled 20% of the 178337 proofs, trained the classifier on 66.6% and evaluated the learned model on the 33.3% unseen proofs. Both the train and the test set contained an equal amount of instances of both classes. A detailed overview of the proofs used to learn the fact classifier can be seen in 21. As expected, there is a skew towards proofs extracted in English (2.4 for English to German, 2.2 for English to French). This is not surprising, since English is the dominant language on the Web (see Figure 7). We chose an SVM as classifier since it is known to be able to handle large sets of features²⁶ and is able to work with numeric data and create confidence values. The ability to generate confidence values for proofs is useful as feedback for users and it also serves as input for the core classifiers described in Section 5.5.5. We achieved an F₁ score of 74.1%. We also performed preliminary work on fine-tuning the parameters of the above algorithms, which, however, did not lead to significantly different results. Therefore, the

²⁵ <http://sourceforge.net/projects/simmetrics/>

²⁶ Note that the majority of the features are word vectors.

English	German	French
R 's novel " D	R in seinem Roman " D	D " est un roman R
R 's book " D	R in seinem Buch " D	R dans son roman " D
R , author of " D	R in seinem Werk " D	R intitulé " D
R married D	D seiner Frau R	R épouse D
R , his wife D	D seiner Ehefrau R	R , veuve D
D 's marriage to R	R und seiner Gattin D	D , la femme de R

Table 22: Example list of patterns for relations *publication* (top) and *marriage* (bottom).

```
[0-9]{4}\\s*(/|-|--|-)\\s*[0-9]{4}
[Ff]rom [0-9]{4} until [0-9]{4}
[bB]etween (the years) [0-9]{4} and [0-9]{4}
[0-9]{4} bis einschließlich [0-9]{4}
[zZ]wischen (den Jahren) [0-9]{4} und [0-9]{4}
[dD]urant la période [0-9]{4} - [0-9]{4}
[eE]ntre les années [0-9]{4} et [0-9]{4}
```

Table 23: Example list of temporal patterns extracted from English, German and French Wikipedia.

reported measurements were carried out with default values of the mentioned algorithms in the Weka machine learning toolkit²⁷ version 3.6.6.

5.5.4 Trustworthiness Analysis of Webpages

The analysis of a webpage's trustworthiness for the extended multilingual version of DeFacto is similar to the one presented in Section 5.4.2. Whereas for the baseline version only English words could be topic terms, words from all languages can be topic terms in the multilingual version. Since many English entries from Wikipedia have corresponding articles in different languages, we can extract topic terms for a multitude of input languages. Also note that through the availability of multilingual labels for many resources in the LOD cloud, we can extract surface forms of resources in all available languages. Due to timely restrictions we chose to exclude the pagerank feature from the extended version of DeFacto.

²⁷ <http://www.cs.waikato.ac.nz/ml/weka/>

5.5.5 Features for Deep Fact Validation

Similar as for the fact confirmation features presented in Section 5.5.3 the DeFacto extension heavily relies on the features of the baseline system as shown in Section 5.4.3. Again, we start by extending the score of single proofs to a score of webpages as shown in Equation 23. We then combine the trustworthiness measures and the textual evidence as previously shown in Equation 24 and 25. In addition to the combinations of trustworthiness and fact confirmation described in Section 5.4.3 as well as the six additional features, we also defined the feature:

The pagerank measure is not used anymore.

7. Statistical triple evidence: Usually certain classes have a higher probability to cooccur as type of subject and object in a given triple, e.g. there might be a higher probability that instances of `dbo:Person` and `dbo:Film` are related via triples than for instance `dbo:Insect` and `dbo:Film`. This observation also holds for the cooccurrence of classes and properties, both for the types in subject and object position. This kind of semantic relatedness allows for computing a score for the statistical evidence STE of a triple $t = (s, p, o)$ by

$$\text{STE}(t) = \max_{\substack{c_s \in \text{cls}(s) \\ c_o \in \text{cls}(o)}} (\text{PMI}(c_s, c_o) + \text{PMI}(c_s, p) + \text{PMI}(p, c_o)) \quad (27)$$

where `cls` denotes the types of the resource and `PMI` denotes the Pointwise Mutual Information, which is a measure of association and defined by

$$\text{PMI}(a, b) = \log \left(N \cdot \frac{\text{occ}(a, b)}{\text{occ}(a) \cdot \text{occ}(b)} \right) \quad (28)$$

using `occ(e)` as number of occurrences of a given entity e in a specific position of a triple and N as the total number of triples in the knowledge base.

5.5.6 Temporal Extension of DeFacto

A major drawback of the previous version of DeFacto was the missing support of temporal validation. There was no way to check if a triple, e.g. `<Tom_Cruise> <spouse> <Katie_Holmes>`, is still true or if it only has been valid in the past. To overcome this drawback we introduce a temporal extension of DeFacto which is able to handle facts that happened on a particular date (*time points*) and facts which span a longer duration (*time periods*). The granularity of both time points and time periods is years. In this section we describe the two sources

for determining the correct time point/period: *a*) statistics, e.g. the (normalized) frequencies of years in proof phrases; and *b*) the combination of statistics and text understanding, by finding lexical patterns expressing temporal information in free text.

5.5.6.1 *Temporal Pattern Extraction*

Our temporal pattern extraction method takes a set of corpora as input. Each corpus is split and indexed on sentence level. After that, we perform index lookups for all sentences which contain at least one of all possible combinations of two years between 1900 and 2013. We apply a context window around the year match in the sentence which ensures that all text excerpts contain at least three (if possible) tokens before, between and after the two year occurrences. Furthermore, we replaced the year occurrences with placeholders and created a frequency distribution for all matching patterns. We then manually relaxed the patterns by generating regular expression versions and added further ones by examining the occurrences of the years in the DeFacto training set.

Specifically in our experiment, we used the Wikipedia dumps (generated in July 2013) as text corpora for the pattern search. After splitting the articles in sentences, the English index contained 64.7M sentences whereas the German with 27.6M and the French with 17.0M sentences are significantly smaller. An excerpt of the used patterns can be seen in Table 23 and the complete list of all patterns can be downloaded from the project homepage.

5.5.6.2 *Year Frequency*

We apply two different year extraction mechanisms for facts with associated time points and time periods. In the first case we create a frequency distribution for tokens which match the regular expression “[1-2][0-9]{3}”. To this end, we take all complex proof phrases extracted from DeFacto for a given triple and create a context window (of variable length) to the left and right of the surface form of the fact’s subject and object. All tokens from all context windows for all complex proofs are combined to compute the year frequency for an input fact. In the case of an associated time period we first try to find year pairs. To find those year pairs we extract a list of multilingual temporal regular expression patterns (see Subsection 5.5.6.1). We apply these patterns to the context of all complex proofs and create a frequency distribution for the start and end year of a given time period. We then choose the most frequent years from both distributions as start- or endpoint of the given fact. In case the temporal patterns do not return any year pairs, we apply a frequency analysis as explained for time points and select the first two years as start or end.

5.5.6.3 Normalizing Frequencies of Years

The extraction of time information is strongly influenced by the growing amount of digital content over time. This leads to more recent dates being more likely to be found on the Web, as shown in Figure 23, than less recent ones. Within DeFacto, we thus implemented two different approaches for normalizing the frequency of years by their popularity on the Web. Each approach defined a *popularity function* pop , which takes a year as input and returns a number representing its popularity. Values below 1 indicate that the year has less than average popularity, where values above 1 indicate an above average popularity. When collecting evidence in DeFacto, we can then divide the frequency of all years found by their popularity value to obtain a distribution, which takes popularity into account. When two years are equally often associated to a particular event, this means DeFacto will assign a higher probability to the year with lower pop -value.

GLOBAL NORMALIZATION Using the data from Figure 23, we have an estimate of the frequency of years, or more specifically a set Y of 4-digit-numbers, on the Web. Assuming that wf (Web frequency) is a function taking a year as input and returning its frequency on the Web, we can define the popularity function as follows:

$$\text{pop}_{\text{global}}(x) = \sqrt{\frac{\text{wf}(x)}{\frac{1}{|Y|} \sum_{y \in Y} \text{wf}(y)}}} \quad (29)$$

This function divides the frequency of a year on the Web by the average frequency of all years in Y on the Web. The square root is used to soften the effect of the normalization.

DOMAIN-SPECIFIC NORMALIZATION A second option to compute the popularity value of a year is to use the training set and actual evidence obtained by DeFacto. In Section 5.5.2, we described how search queries are generated and text excerpts (proofs) are extracted from the resulting webpages. Let pf (proof frequency) be a function taking a year as input and returning its frequencies in all proofs generated when running DeFacto over a training set. Furthermore, let tf (training set frequency) be the frequency of correct years in the training set.

Intuitively, we can expect years frequently in the training set to also frequently occur in the generated proofs. Therefore, we first divide pf by tf and then apply an analogous approach to the global normalization introduced above:

$$\text{pop}_{\text{domain}}(x) = \sqrt{\frac{\frac{\text{pf}(x)}{\text{tf}(x)}}{\frac{1}{|Y|} \sum_{y \in Y} \frac{\text{pf}(y)}{\text{tf}(y)}}}} \quad (30)$$

The fictional example below shows the results of our approach when using only three years as input:

	1990	2000	2010
pf	30	30	100
tf	5	3	3
pop _{domain}	0.36	0.61	2.03

In this example, 2000 is more popular than 1990, since it has the same frequency in proofs, but a lower frequency in the training set. 2010 is more popular than 2000, since it has a higher frequency in proofs, but the same frequency in the training set.

5.5.7 FactBench - A Fact Validation Benchmark

FactBench is a multilingual benchmark for the evaluation of fact validation algorithms. All facts in FactBench are scoped with a timespan in which they were true, enabling the validation of temporal relation extraction algorithms. FactBench currently supports English, German and French. The current release V1 is freely available (MIT License) at <http://github.com/AKSW/FactBench>. FactBench consists of a set of RDF models. Each of the 1500 models contains a singular fact and the time period in which it holds true. In addition, the FactBench suite contains the SPARQL and MQL queries used to query Freebase²⁸ and DBpedia, a list of surface forms for English, French and German as well as the number of incoming and outgoing links for the English wikipedia pages. FactBench provides data for 10 relations. The data was automatically extracted from DBpedia and Freebase. A detailed description on what facts the benchmark contains is shown in Figure 24. The granularity of FactBench’s time information is year. This means that a timespan is an interval of two years, e.g. 2008 - 2012. A time point is considered as a timespan with the same start and end year, e.g. 2008 - 2008.

FactBench is divided in a training and a testing set (of facts). This strict separation avoids the overfitting of machine learning algorithms to the training set, by providing unseen test instances.

²⁸ Since there are no incremental releases from Freebase we include the crawled training data.

Relation	Subjects	Objects	Type	Year _{min}	Year _{max}	Year _{avg}	Source	Comment
birth	75/75	67/65	point	1166/1650	1989/1987	1925/1935	DBpedia	birth place (city) and date of persons
death	75/75	54/48	point	1270/1677	2013/2012	1944/1952	DBpedia	death place (city) and date of persons
team	50/52	24/27	point	2001/2001	2012/2012	2007/2007	DBpedia	NBA players for a NBA team (after 2000)
award	75/75	5/5	point	1901/1901	2007/2007	1946/1952	Freebase	winners of nobel prizes
foundation	75/75	59/62	point	1865/1935	2006/2008	1988/1990	Freebase	foundation place and time of software companies
publication	75/75	75/73	point	1818/1918	2006/2006	1969/1980	Freebase	authors of science fiction books (one book/author)
spouse	74/74	74/74	point	2003/2003	2013/2013	2007/2007	Freebase	marriages between actors (after 2013/01/01)
starring	22/21	74/74	period	1954/1964	2009/2009	1992/1993	DBpedia	actors starring in a movie
leader	75/75	36/43	period	1840/1815	2013/2012	1973/1972	DBpedia	prime ministers of countries
subsidiary	54/50	75/75	period	1993/1969	2007/2007	2003/2002	Freebase	company acquisitions

Table 24: Overview of all correct facts of the training and testing set (train/test).

5.5.7.1 *Expression of Temporal Information in RDF Knowledge Bases*

There are several methods to model temporal information in the Data Web. According to Rula et al. [2012], we can distinguish the following main categories:

- Document-centric, e.g. time points are connected to documents via the last modified HTTP header.
- Fact-centric, e.g. temporal information refers to facts. This can be divided into sentence-centric and relationship-centric perspectives. In the sentence-centric perspective, the temporal validity of one or more statements is defined by annotating the facets. In the relationship-centric perspective, n-ary relations are used to encapsulate temporal information.

Popular knowledge bases show a variety of different modeling choices for temporal information: DBpedia uses a class `dbo:TimePeriod` and attaches various properties to it, e.g. `dbo:activeYearsEndDate` is used to associate an `xsd:date` to mark the end of some activity. Freebase is fact-centric, more specifically relationship centric, and uses n-ary relations. YAGO is sentence-centric and uses reification to attach temporal restrictions to statements. Furthermore, there exists a variety of ontologies and standards related to representing temporal information, e.g. the OWL time ontology²⁹, XML Schema Date Datatypes³⁰, ISO standard 8601³¹, Dublin Core time interval encoding³² and Linked Timelines [Correndo et al., 2010]. For FactBench, we adopt a temporal representation similar to the one used in DBpedia, although other options appear to be equally appropriate. Listing 10 shows an example fact.

5.5.7.2 *Data Generation*

The data generation of the FactBench benchmark has three objectives. First, we try to cover as many different domains as possible. The benchmark includes, amongst others, relations from the persons (marriage), places (birth, death) and organizations domain (subsidiary). In addition, it also contains relations which would usually fall into the miscellaneous domain, e.g. award or publication. Also the data is derived from multiple sources, e.g. DBpedia and Freebase. Second, we intended to not only cover relations for a fixed point in time (e.g. a person's birth date) but to also include relations which appear over a longer period of time (e.g. the marriage between two people). Lastly, we also tried to cover relations which appeared before the information age (before 1980). An overview of the most frequently occurring

²⁹ <http://www.w3.org/TR/owl-time/>

³⁰ www.w3.org/TR/xmlschema-2/

³¹ http://en.wikipedia.org/wiki/ISO_8601#Time_intervals

³² <http://dublincore.org/documents/dcmi-period/>

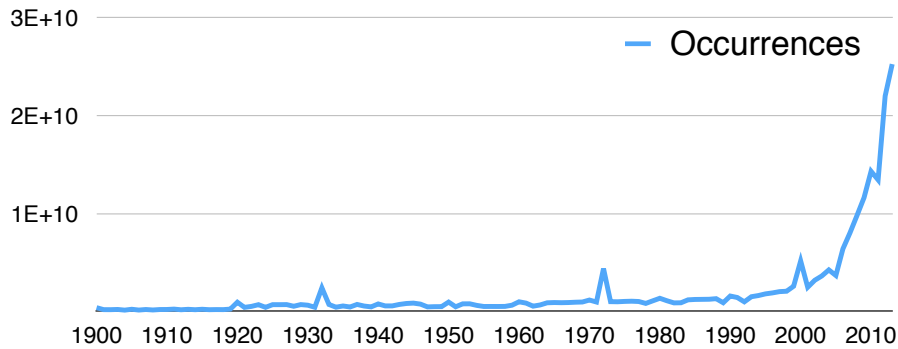


Figure 23: Distribution of year numbers in World Wide Web. Shows approximate number of Google search results. Outliers from left to right, 1931, 1972 and 2000. As comparison 'EU' has about 2.280.000.000 and 'Obama' 478.000.000 hits.

time points and time periods of the FactBench train set can be seen in Figure 24.

```

1  @prefix fbase: <http://rdf.freebase.com/ns/> .
2  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3  @prefix dbo: <http://dbpedia.org/ontology/> .
4  @prefix dbr: <http://dbpedia.org/resource/> .
5  @prefix frdbr: <http://fr.dbpedia.org/resource/> .
6  @prefix dedbr: <http://de.dbpedia.org/resource/> .
7  @prefix owl: <http://www.w3.org/2002/07/owl#> .
8  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
9  @prefix skos: <http://www.w3.org/2004/02/skos/core#> .
10
11 fbase:m.0dt39
12   rdfs:label
13     "Nobel Prize in Physics"@en,
14     "Prix Nobel de physique"@fr,
15     "Nobelpreis für Physik"@de ;
16   skos:altLabel
17     "Nobel Physics Prize"@en ,
18     "Nobel laureates in physics"@fr ,
19     "Physik-Nobelpreis"@de ...
20   owl:sameAs
21     frdbr: Prix_Nobel_de_physique ,
22     dedbr: Nobelpreis_für_Physik ,
23     dbr: Nobel_Prize_in_Physics ;
24
25 fbase:m.0jcx_24
26   dbo:startYear "1921"^^xsd:gYear ;
27   dbo:endYear "1921"^^xsd:gYear .
28   dbo:award fbase:m.0dt39 ;
29
30 fbase:m.0jcx
31   dbo:recievedAward fbase:m.0jcx_24 ;
32   rdfs:label
33     "Albert Einstein"@fr ,
34     "Albert Einstein"@en ,
35     "Albert Einstein"@de ;
36   skos:altLabel
37     "A. Einstein"@fr ,
38     "Einstein, Albert"@de ,
39     "Albert Einstein"@en ...
40   owl:sameAs
41     dbr: Albert_Einstein ,
42     frdbr: Albert_Einstein ,
43     dedbr: Albert_Einstein ;

```

Listing 10: Example of a fact in FactBench.

POSITIVE EXAMPLES In general, we use facts contained in DBpedia and Freebase as positive examples. Since for most relations there is far more data available than necessary we had to select a subset. For each of the properties we consider, we generated positive exam-

ples by issuing a SPARQL or MQL query and selecting the top 150 results. Note that the results in Freebase (MQL) are ordered by an internal relevance score³³. The results for the DBpedia SPARQL queries were ordered by the number of inbound-links of a given resources' Wikipedia page. We collected a total of 1500 correct statements (750 in test and train set). Each relation has 150 correct facts distributed equally in the test and train set.

NEGATIVE EXAMPLES The generation of negative examples follows the same idea as presented in Section 5.4.4.1 for the DeFacto baseline version since we still consider it essential that many of the negative examples are similar to true statements. In particular, most statements should be meaningful triples. Assume that the input triple $t = (s, p, o)$ and the corresponding time period $tp = (from, to)$ in a knowledge base K is given and let S be the set of all subjects, O the set of all objects of the given property p and P the set of all properties. For FactBench we generate the basic five training/test sets, *domain*, *range*, *domain-range*, *property* and *random*, as previously presented and generate the additional and updated training/test sets *date* and *mix* as follows:

DATE A triple $(s, p, o)(from', to')$ is generated. For time points $from'$ is a random year drawn from a gaussian distribution ($\mu = from$ and $\sigma^2 = 5$), $from' = to'$, $from' \neq from$ and $0 < from' \leq 2013$. For timespans $from'$ is a random year drawn from a gaussian distribution ($\mu = from$ and $\sigma^2 = 2$), the duration d' is generated by drawing a random number from a gaussian distribution ($\mu = to - from$ and $\sigma^2 = 5$), $to' = from' + d'$, $0 < d' \leq 2013$, $from \neq from'$, $to \neq to'$, $from \leq 2013$ and $to \leq 2013$.

MIX 1/6 of each of the above created negative training sets were randomly selected to create a heterogenous test set. Note that this set contains 780 negative examples.

5.5.8 Evaluation

The aim of our evaluation was three-fold. We wanted to quantify how well/much a) DeFacto can distinguish between correct and wrong facts; b) DeFacto is able to find correct time points or time periods for a given fact and if the year frequency distribution (1900 vs. 2013) does influence the accuracy; and c) the use of multilingual patterns boost the results of DeFacto with respect to fact validation and date detection. In the following, we describe how we set up our evaluation system, present the experiments we devised and discuss our findings.

³³ http://wiki.freebase.com/wiki/Search_Cookbook

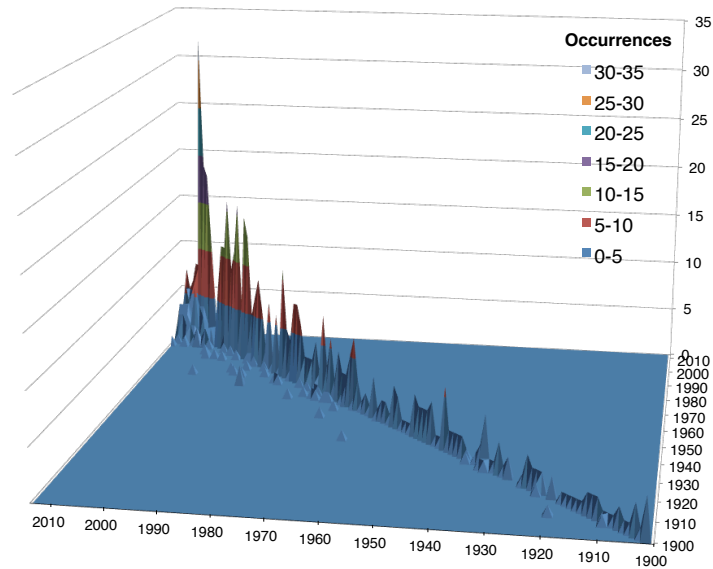


Figure 24: Overview of time points and time periods in the FactBench train set.

5.5.8.1 Experimental Setup

In a first step, we computed all feature vectors, described in Section 5.5.5 for the training and test sets. DeFacto relies heavily on web requests, which are not deterministic (i.e. the same search engine query does not always return the same result). To achieve deterministic behavior and to increase the performance as well as reduce load on the servers, all web requests were cached. The DeFacto runtime for an input triple was on average slightly below four seconds per input triple³⁴ when using caches.

We stored the features in the ARFF file format and employed the WEKA machine learning toolkit³⁵ for training different classifiers. In particular, we were interested in classifiers which can handle numeric values and output confidence values. Naturally, confidence values for facts such as, e.g. 95%, are more useful for end users than just a binary response on whether DeFacto considers the input triple to be true, since they allow a more fine-grained assessment. We selected popular machine-learning algorithms satisfying those requirements.

As mentioned in Section 5.5.1, we focused our experiments on the 10 relations from FactBench. The system can be extended easily to cover more properties by extending the training set of BOA to those properties. Note that DeFacto itself is also not limited to DBpedia or Freebase, i.e. while all of its components are trained on these datasets, the algorithms can be applied to arbitrary URIs and knowledge bases.

³⁴ The performance is roughly equal on server machines and notebooks, since the web requests dominate.

³⁵ <http://www.cs.waikato.ac.nz/ml/weka/>

5.5.8.2 Fact Scoring

For this evaluation task we used each FactBench training set to build an independent classifier. We then used the classifier on the corresponding test set to evaluate the built model on unseen data. The task’s results can be seen in Table 25. The J48 algorithm, an implementation of the C4.5 decision tree – shows the most promising results. Given the challenging tasks, F-measures up to 84.9% for the *mix* test set appear to be very positive indicators that DeFacto can be used to effectively distinguish between true and false statements, which was our primary evaluation objective. In general, DeFacto also appears to be stable against the various negative test sets given the F_1 values ranging from 89.7% to 91% for the *domain*, *range*, *domainrange* and *random* test set. In particular, the algorithms with overall positive results also seem less affected by the different variations. On the *property* test set, in our opinion the hardest task, we achieved an F_1 score of 68.7%. Due to the results achieved, we use J48 as the main classifier in DeFacto and, more specifically, its results on the mix sets as this covers a wide range of scenarios. We observe that the learned classifier has an error rate of 3% for correct facts, but fails to classify 55.3% of the false test instances as incorrect.

We also performed an evaluation to measure the performance of the classifier for each of the relations in FactBench. The results of the evaluation are shown in Figure 25. We used the precision of the main classifier (J48 on the *mix* models) on the correct subset for this figure.³⁶ The average precision for all relations is 89.2%. The worst precision for an individual relation, i.e 69%, is achieved on the *foundation* relation, which is by far the least frequent relation on the Web with respect to search engine results.

	Domain						
	C	P	R	F_1	AUC	RMSE	
J48	89.7%	0.898	0.897	0.897	0.904	0.295	
SimpleLogistic	89.0%	0.890	0.890	0.890	0.949	0.298	
NaiveBayes	81.2%	0.837	0.812	0.808	0.930	0.415	
SMO	85.4%	0.861	0.854	0.853	0.854	0.382	
	Range						
	C	P	R	F_1	AUC	RMSE	
J48	90.9%	0.909	0.909	0.909	0.954	0.271	
SimpleLogistic	88.0%	0.880	0.880	0.880	0.946	0.301	
NaiveBayes	83.3%	0.852	0.833	0.830	0.933	0.387	

³⁶ We are using the correct subset, since some negative examples are generated by replacing properties as described in Section 5.5.7.2. For those, it would not be clear, which property they refer to.

SMO	83.3%	0.852	0.833	0.830	0.833	0.409
DomainRange						
	C	P	R	F ₁	AUC	RMSE
J48	91.0%	0.910	0.910	0.910	0.953	0.270
SimpleLogistic	88.9%	0.889	0.889	0.889	0.950	0.296
NaiveBayes	84.5%	0.861	0.845	0.843	0.935	0.380
SMO	83.6%	0.853	0.836	0.834	0.836	0.405
Property						
	C	P	R	F ₁	AUC	RMSE
J48	70.8%	0.786	0.708	0.687	0.742	0.427
SimpleLogistic	64.9%	0.653	0.649	0.646	0.726	0.460
NaiveBayes	61.3%	0.620	0.613	0.608	0.698	0.488
SMO	64.6%	0.673	0.646	0.632	0.646	0.595
Random						
	C	P	R	F ₁	AUC	RMSE
J48	90.9%	0.910	0.909	0.909	0.933	0.283
SimpleLogistic	87.8%	0.879	0.878	0.878	0.954	0.293
NaiveBayes	84.1%	0.851	0.841	0.839	0.942	0.375
SMO	84.3%	0.864	0.843	0.841	0.843	0.396
Mix						
	C	P	R	F ₁	AUC	RMSE
J48	84.9%	0.850	0.849	0.849	0.868	0.358
SimpleLogistic	80.2%	0.810	0.802	0.799	0.880	0.371
NaiveBayes	78.7%	0.789	0.787	0.787	0.867	0.411
SMO	76.9%	0.817	0.769	0.756	0.754	0.480

Table 25: Classification results for FactBench test sets (C = correctness, P = precision, R = recall, F₁ = F₁ Score, AUC = area under the curve, RMSE = root mean squared error).

5.5.8.3 Date Scoring

To estimate time scopes, we first needed to determine appropriate parameters for this challenging task. To this end, we varied the context size from 25, 50, 100 and 150 characters to the left and right of the proofs subject and object occurrence. Additionally, we also varied the used languages which is discussed in more detail in Section 5.5.8.4. The final parameter in this evaluation was the normalization approach. As introduced in Section 5.5.6, we used the occurrence (number of occurrences of years in the context for all proofs of a fact),

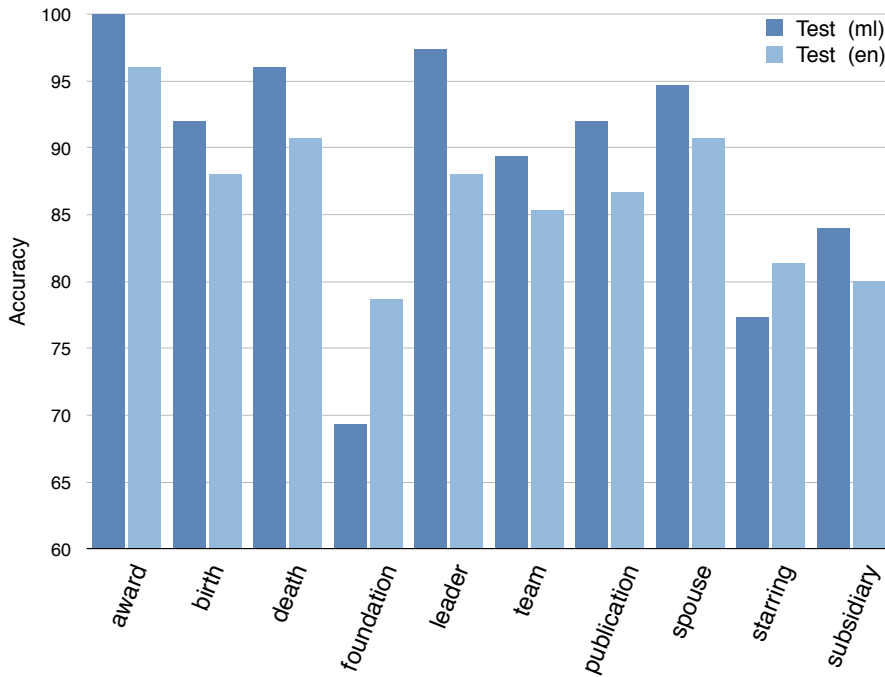


Figure 25: Accuracy results for learned J48 *mix* classifier on correct subset of the test set. The abbreviation ml indicates that multilingual (English, French, German) search results and surface forms were used, en is limited to English only.

the domain and range approach. We performed a grid search for the given parameters on the *correct* train set. As performance measures we choose precision³⁷ P (shown in Equation 31), recall R (shown in Equation 32) and F-measure, defined as $F_1 = 2 * \frac{P * R}{P + R}$.

$$P = \frac{|\text{relevant years} \cap \text{retrieved years}|}{|\text{retrieved years}|} \quad (31)$$

$$R = \frac{|\text{relevant years} \cap \text{retrieved years}|}{|\text{relevant years}|} \quad (32)$$

If for example, for a single fact the correct time period is 2008 (a time point), the F_1 score is either 0 or 1. However, if the correct time period is 2011 – 2013 and the retrieved results are 2010 – 2013, we would achieve a precision $P = \frac{3}{4}$ (three of the four retrieved years are correct) and a recall $R = 1$ (all of the relevant years were found), resulting in an F_1 score of $\frac{6}{7}$.

The final results for the train set are shown in Table 27. Please note that it is out of scope of this chapter to decide whether a given property requires a time period or a time point. As expected, facts with time points show a higher F_1 measure as facts with time periods. Calculating the average F_1 score for the individual relations leads to $F_1 = 70.2\%$ for time points and $F_1 = 65.8\%$ for relations associated with time periods. The relations performing well on fact scoring also

³⁷ Finding no year candidate for a given fact only influences the recall.

appear to be better suited for year scoping, e.g. the *award* relation. In general, the training results show that the *domain* normalization performs best and the optimal context size varies for each relation. We now applied the learned parameters for each relation on the FactBench *correct* test subset. The results are shown in Table 28. The average F_1 score decreases by 2.5% to 67.7% for time points and 4.6% to 61.2% for time period relations compared to the train set. Since it is not possible to determine a correct time point or time period for all facts (the context does not always include the correct year(s)) we also calculated DeFacto’s accuracy. We define the accuracy acc for a time period tp as follows:

$$\text{acc}(\text{tp}) = \begin{cases} 1 & \text{if } \text{tp}_{\text{from}} \text{ is correct } \wedge \text{tp}_{\text{to}} \text{ is correct} \\ 0 & \text{otherwise.} \end{cases} \quad (33)$$

The average accuracy for time point (from and to are equal) relations is 76%. Since for time periods we have to match both start and end year, which aggravates this task significantly, we achieved an accuracy of 44% on this dataset. Finally, we wanted to see if DeFacto’s performance is influenced by how recent a fact is. We grouped the time intervals in buckets of 10 years and plotted the proportion of correctly classified facts within this interval. We did this for the multilingual as well as the English-only setting of DeFacto. The results are shown in Figure 26. In general, all values are between 80% and 100% for the English version and between 93% and 100% for the multilingual version. While there is some variation, no obvious correlation can be observed, i.e. DeFacto appears to be able to handle recent and older facts. In this figure, it is interesting to note that the multilingual setting appears to be more stable and perform better. We performed a paired t-test using all 750 facts and obtained that the improvement of the multilingual setting is statistically very significant.

	C	P	R	F_1	AUC	RMSE
J48	83.4%	0.834	0.834	0.834	0.877	0.361
SimpleLogistic	80.6%	0.811	0.806	0.804	0.884	0.368
NaiveBayes	78.1%	0.788	0.781	0.782	0.872	0.428
SMO	78.6%	0.816	0.786	0.777	0.773	0.463

Table 26: Classification results for FactBench mix test set on English language only.

Set	occurrence						global						domain					
	C	P	R	F	P ₇₅	A	C	P	R	F	P ₇₅	A	C	P	R	F	P ₇₅	A
award _{en}	25	100	98.7	99.3	74	100	25	98.6	97.3	98	74	98.6	100	100	98.7	99.3	74	100
award _{ml}	25	100	98.7	99.3	74	100	25	100	98.7	99.3	74	100	25	100	98.7	99.3	74	100
birth _{en}	25	83.3	80	81.6	69	87	50	91.7	88	89.8	70	94.3	50	76.4	73.3	74.8	70	78.6
birth _{ml}	50	93.2	92	92.6	73	94.5	25	94.6	93.3	94	73	95.9	25	89.2	88	88.6	73	90.4
death _{en}	50	74.3	73.3	73.8	69	79.7	25	61.1	58.7	59.9	68	64.7	25	80.6	77.3	78.9	68	85.3
death _{ml}	25	77.3	77.3	77.3	75	77.3	25	66.7	66.7	66.7	75	66.7	25	84	84	84	75	84
foundation _{en}	150	14.1	12	12.9	28	32.1	150	17.2	14.7	15.8	28	39.3	150	25	21.3	23	28	57.1
foundation _{ml}	25	16.4	13.3	14.7	23	43.5	150	21.7	20	20.8	41	36.6	150	26.1	24	25	41	43.9
publication _{en}	100	58.3	56	57.1	63	66.7	150	60.3	58.7	59.5	67	65.7	100	51.4	49.3	50.3	63	58.7
publication _{ml}	25	70.8	68	69.4	68	75	150	74.7	74.7	74.7	72	77.8	50	60	60	60	70	64.3
starring _{en}	25	64.4	38.7	48.3	35	82.9	50	67.9	48	56.3	40	90	100	59.3	46.7	52.2	46	76.1
starring _{ml}	25	59.6	45.3	51.5	44	77.3	50	58.1	48	52.6	48	75	100	62.7	56	59.2	57	73.7
subsidiary _{en}	100	63.5	44	52	45	73.3	50	63	38.7	47.9	39	74.4	150	64.8	46.7	54.3	46	76.1
subsidiary _{ml}	25	70.8	45.3	55.3	43	79.1	25	68.8	44	53.7	43	76.7	25	70.8	45.3	55.3	43	79.1

Set	occurrence							global							domain						
	C	P	R	F	P ₇₅	A	C	C	P	R	F	P ₇₅	A	C	C	P	R	F	P ₇₅	A	C
spouse _{en}	100	67.5	68	67.7	53	50.9	25	75.5	64.4	69.5	37	78.4	25	77.1	65.2	70.6	37	78.4			
spouse _{ml}	25	69.6	66.5	68	49	59.2	25	70.8	65.6	68.1	49	55.1	25	75.2	67.2	71	49	61.2			
nbateam _{en}	100	54.2	47.4	50.6	44	34.1	100	57.8	47	51.9	44	34.1	150	59.1	48.4	53.2	53	28.3			
nbateam _{ml}	50	60.2	58.1	59.1	58	25.9	100	62.1	55.4	58.6	63	23.8	25	65.2	58.7	61.8	53	32.1			
leader _{en}	100	42.6	65.1	51.5	55	41.8	100	42.6	63.1	50.9	55	41.8	100	46.7	64.4	54.1	55	43.6			
leader _{ml}	100	53.6	75.4	62.6	72	44.4	100	53.3	75.6	62.5	72	44.4	100	55.9	76.7	64.7	72	45.8			
timepoint _{en}	25	61	48	53.7	277	78	25	60.2	47.3	53	277	76.9	100	57.8	50	53.6	317	71			
timepoint _{ml}	25	65.9	56.7	60.9	326	78.2	25	64.1	55.1	59.3	326	76.1	150	61.6	58.2	59.9	373	70.2			
timeperiod _{en}	100	54.7	60.2	57.3	152	42.8	100	54.9	60.3	57.4	152	42.8	100	58.7	60.6	59.7	152	44.7			
timeperiod _{ml}	100	59	67.2	62.8	198	38.9	100	59.4	67.5	63.2	198	39.4	100	63	69	65.9	198	40.9			
all _{en}	50	61.3	56.2	58.6	496	72	25	64	54	58.6	460	75.4	100	62.7	58.1	60.3	543	67.6			
all _{ml}	25	67.1	63.2	65.1	568	70.1	25	66.3	62.4	64.3	568	68.7	100	66.1	65.2	65.7	635	65.4			

Table 27: Overview of the time-period detection task for the FactBench training set with respect to the different normalization methods. ml (multilingual) indicates the use of all three languages (en,de,fr).

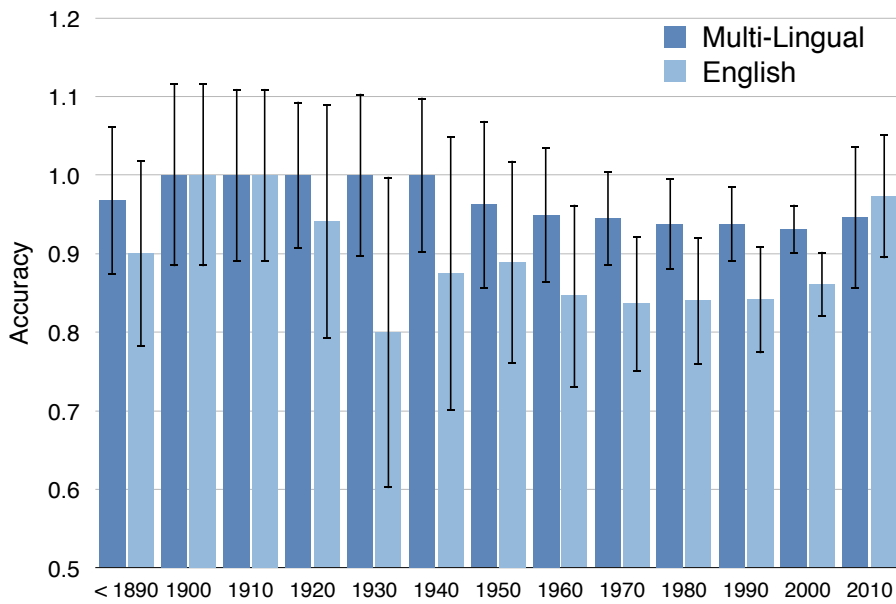


Figure 26: A plot showing the proportion of correctly classified facts (y-axis) for the FactBench *mix-correct*-test-set using the J48 classifier. The time intervals (x-axis) are buckets of ten years, e.g. 1910 stands for all years from 1910 to 1919. Results for the multilingual and English-only setting of DeFacto are shown.

5.5.8.4 Effect of Multilingual Patterns

The last question we wanted to answer in this evaluation is how much the use of the multilingual patterns boosts the evidence scoring as well as the date scoping. For the fact scoring we trained different classifiers on the *mix* training set. We only used English patterns and surface forms to extract the feature vectors. As the results in Table 26 on the test set show, J48 is again the highest scoring classifier, but is outperformed by the multilingual version shown in Table 25 by 1.5% F₁ score. The detailed analysis for the different relations in Figure 25 indicates a superiority of the multilingual approach. We also performed the grid search as presented in Section 5.5.8.3 for English patterns and surface forms only. As shown in Table 27 the multilingual date scoping approach outperforms the English one significantly on the training set. The multilingual version achieved an average 4.3% on the time point and a 6.5% better F₁ measure on time period relations. The difference is similar on the test set, where the difference is 6.5% for time points and 6.9% for time period relations.

Finally, as shown in Figure 26, the English version performs equally well on recent dates, but performs worse for less recent dates, which is another indicator that the use of a multilingual approach is preferable to an English-only setting.

Set _{language} ^{context}	P	R	F	MRR	C _S	C _E	P ₇₅	Acc
award _{en} ¹⁰⁰	93.3	93.3	93.3	100	70	-	75	93.3
award _{ml} ²⁵	93.3	93.3	93.3	100	70	-	75	93.3
birth _{en} ⁵⁰	77.8	74.7	76.2	81.6	56	-	69	81.2
birth _{ml} ²⁵	93.2	92	92.6	93.3	69	-	73	94.5
death _{en} ²⁵	72	72	72	84.5	54	-	69	78.3
death _{ml} ²⁵	81.3	81.3	81.3	87.1	61	-	74	82.4
foundation _{en} ¹⁵⁰	22.2	18.7	20.3	66.1	14	-	20	70
foundation _{ml} ¹⁵⁰	20.3	18.7	19.4	48.1	14	-	33	42.4
publication _{en} ¹⁵⁰	62	58.7	60.3	77.8	44	-	68	64.7
publication _{ml} ¹⁵⁰	67.6	66.7	67.1	75.5	50	-	74	67.6
starring _{en} ⁵⁰	57.1	48	52.2	87.1	36	-	44	81.8
starring _{ml} ¹⁰⁰	61.4	57.3	59.3	73.6	43	-	60	71.7
subsidiary _{en} ¹⁵⁰	60.7	49.3	54.4	79.3	37	-	53	69.8
subsidiary _{ml} ²⁵	70.2	53.3	60.6	87.5	40	-	50	80
spouse _{en} ²⁵	69.2	59	63.6	-	34	35	34	76.5
spouse _{ml} ²⁵	73.7	61.4	67	-	36	36	42	59.5
team _{en} ¹⁵⁰	52.7	42.7	47.2	-	25	16	51	23.5
team _{ml} ²⁵	59.9	49.6	54.3	-	28	16	45	26.7
leader _{en} ¹⁰⁰	46.3	60.8	52.5	-	29	29	56	44.6
leader _{ml} ¹⁰⁰	55	71.5	62.2	-	38	37	72	45.8
timepoint _{en} ²⁵	62	52	56.6	85.8	273	273	356	76.7
timepoint _{ml} ²⁵	66.9	60.6	63.6	87.1	318	318	404	78.7
timeperiod _{en} ¹⁰⁰	55.7	55.6	55.7	-	92	82	159	41.5
timeperiod _{ml} ¹⁰⁰	59.6	60.1	59.8	-	102	91	195	38.5
all _{en} ¹⁰⁰	58.2	54.4	56.2	-	375	365	563	62
all _{ml} ¹⁰⁰	61.5	59.6	60.5	-	414	403	634	61

Table 28: Overview of the *domain*-normalization on the FactBench test set. ml (multilingual) indicates the use of all three languages (en,de,fr). C_(S|E) shows the number of correct start and end years, P₇₅ is the number of time-periods possible to detect correctly and A is the accuracy on P₇₅.

5.6 CONCLUSION

In this chapter, we presented DeFacto, an approach for checking the validity of RDF triples using the Web as corpus. We introduced the DeFacto baseline version in Section 5.4. DeFacto uses English natural language patterns of formal relations for the translation of input triples into natural language to query the Web for evidence. The presented approach is able to combine textual evidence of facts with the trustworthiness of websites in which the fact occurs. We evaluated our approach for 60 properties and showed that our approach achieves an average F_1 measure (J48 for all 6 datasets) of 0.843 on DBpedia.

In Section 5.5, we presented the extension of the DeFacto baseline, a multilingual and temporal approach for checking the validity of RDF triples using the Web as corpus. In more detail, we explicated how multilingual natural language patterns for formal relations can be used for fact validation. In addition, we presented an extension for detecting the temporal scope of RDF triples with the help of pattern and frequency analysis. We support the endeavor of creating better fact validation algorithms (and to that end also better relation extraction systems) by providing the full-fledged benchmark FactBench. This benchmark consists of one training and several test sets for fact validation as well as temporal scope detection. We showed that our approach achieves an F_1 measure of 84.9% on the most realistic fact validation test set (FactBench *mix*) on DBpedia as well as Freebase data. The temporal extension shows a promising average F_1 measure of 70.2% for time point and 65.8% for time period relations. The use of multilingual patterns increased the fact validation F_1 by 1.5%. Moreover, it raised the F_1 for the date scoping task of up to 6.9%. It is also important that our approach can now be used on non-English knowledge bases.

TEMPLATE-BASED QUESTION ANSWERING OVER RDF DATA

As an increasing amount of RDF data is published as Linked Data, intuitive ways of accessing this data become more and more important. Question answering approaches have been proposed as a good compromise between intuitiveness and expressivity. Most question answering systems translate questions into triples which are matched against the RDF data to retrieve an answer, typically relying on some similarity metric. However, in many cases, triples do not represent a faithful representation of the semantic structure of the natural language question, with the result that more expressive queries cannot be answered. To circumvent this problem, we present a novel approach that relies on a parse of the question to produce a SPARQL template that directly mirrors the internal structure of the question. This template is then instantiated using statistical entity identification and predicate detection. We show that this approach is competitive and discuss cases of questions that can be answered with our approach but not with competing approaches.

This chapter is mainly based on Unger et al. [2012] and Höffner et al. [2013].

6.1 INTRODUCTION

As more and more RDF data is published as Linked Data, developing intuitive ways of accessing this data becomes increasingly important. One of the main challenges is the development of interfaces that exploit the expressiveness of the underlying data model and query language, while hiding their complexity. As a good compromise between intuitiveness and expressivity, question answering approaches allow users to express arbitrarily¹ complex information needs in natural language without requiring them to be aware of the underlying schema, vocabulary or query language. Several question answering systems for RDF data have been proposed in the past, for example, Aqualog [Lopez and Motta, 2004; V. Lopez and Pasin, 2007], PowerAqua [V. Lopez and Motta, 2009], NLP-Reduce [E. Kaufmann, 2007], FREyA [Damljanovic et al., 2010] and Pythia Unger and Cimiano [2011]. Many of these systems map a natural language question to a triple-based representation. For example, consider the simple question *Who wrote The Neverending Story?*. PowerAqua² would map this question to the triple representation

\langle [person,organization], wrote, Neverending Story \rangle .

Then, by applying similarity metrics and search heuristics, it would retrieve matching subgraphs from the RDF repository. For the above query, the following triples would be retrieved from DBpedia, from which the answer “Michael Ende” can be derived:

\langle Writer, IS_A, Person \rangle
 \langle Writer, author, The_Neverending_Story \rangle

While this approach works very well in cases where the meaning of the query can be captured easily, it has a number of drawbacks, as in many cases the original semantic structure of the question can not be faithfully captured using triples. For instance, consider the questions [1a](#) and [2a](#) below. PowerAqua would produce the triple representations in [1b](#) and [2b](#), respectively. The goal, however, would be SPARQL queries³ like [1c](#) and [2c](#), respectively.

1. a) Which cities have more than three universities?
 b) \langle [cities], more than, universities three \rangle
 c)

```
SELECT ?y WHERE {
    ?x rdf:type onto:University .
    ?x onto:city ?y .
  }
  HAVING (COUNT(?x) > 3)
```

¹ At least as complex as can be represented in the query language.

² Accessed via the online demo at <http://poweraqua.open.ac.uk:8080/poweraqualinked/jsp/index.jsp>.

³ Assuming a DBpedia namespace with `onto` as prefix `<http://dbpedia.org/ontology/>`.

2. a) Who produced the most films?
- b) \langle [person,organization], produced, most films \rangle
- c)

```
SELECT ?y WHERE {
    ?x rdf:type onto:Film .
    ?x onto:producer ?y .
}
ORDER BY DESC(COUNT(?x)) OFFSET 0 LIMIT 1
```

Such SPARQL queries are difficult to construct on the basis of the above mentioned triple representations, as aggregation and filter constructs arising from the use of specific quantifiers are not faithfully captured. What would be needed instead is a representation of the information need that is much closer to the semantic structure of the original question. Thus, we propose a novel approach to question answering over RDF data that relies on a parse of the question to produce a SPARQL template that directly mirrors the internal structure of the question and that, in a second step, is instantiated by mapping the occurring natural language expressions to the domain vocabulary. For example, a template produced for Question 2a would be:

3.

```
SELECT ?x WHERE {
    ?x ?p ?y .
    ?y rdf:type ?c .
}
ORDER BY DESC(COUNT(?y)) LIMIT 1 OFFSET 0
```

In this template, *c* stands proxy for the URI of a class matching the input keyword *films* and *p* stands proxy for a property matching the input keyword *produced*. In a next step, *c* has to be instantiated by a matching class, in the case of using DBpedia `onto:Film`, and *p* has to be instantiated with a matching property, in this case `onto:producer`. For instantiation, we exploit an index as well as the BOA pattern library (see Chapter 3) that links properties with natural language predicates.

We show that this approach is competitive and discuss specific cases of questions that can be precisely answered with our approach but not with competing approaches. Thus, the main contribution of this paper is a domain-independent question answering approach that first converts natural language questions into queries that faithfully capture the semantic structure of the question and then identifies domain-specific entities combining NLP methods and statistical information.

In the following section we present an overview of the system's architecture. In Section 6.3 we report our evaluation results, exemplify the impact of the usage of the BOA Pattern Library and point to an online interface of the prototype in Section 6.4. In Section 6.5 we compare our approach to existing question answering systems on RDF data, before concluding in Section 6.6.

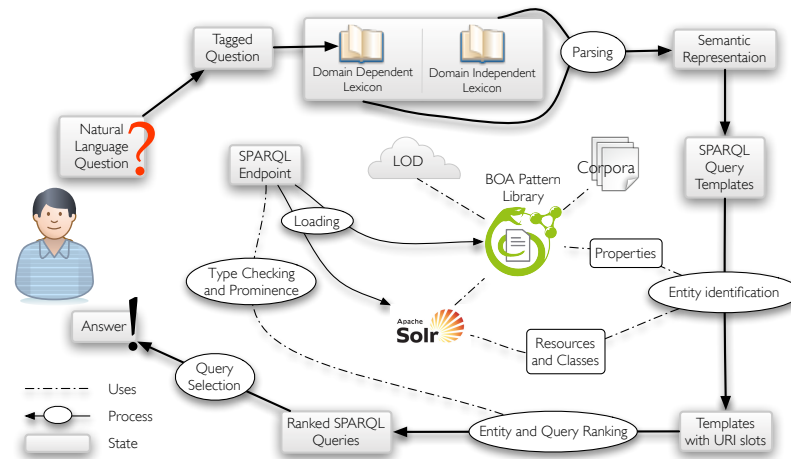


Figure 27: Overview of the template based SPARQL query generator.

AUTHOR CONTRIBUTIONS The author of this thesis was a co-author of Unger et al. [2012]; Lehmann et al. [2012a]; Höffner et al. [2013] and carried out the extension of the BOA architecture to also support OWL datatype properties as well as trained the BOA framework for the properties contained in the QALD benchmark.

6.2 OVERVIEW

Figure 27 gives an overview of our approach. The input question, formulated by the user in natural language, is first processed by a POS tagger. On the basis of the POS tags, lexical entries are created using a set of heuristics. These lexical entries, together with pre-defined domain-independent lexical entries, are used for parsing, leading to a semantic representation of the natural language query, which is then converted into a SPARQL query template.

The query templates contain *slots*, which are missing elements of the query that have to be filled with URIs. In order to fill them, our approach first generates natural language expressions for possible slot fillers from the user question using WordNet expansion. In a next step, sophisticated entity identification approaches are used to obtain URIs for those natural language expressions. While the detection of resources and classes can be reduced to a retrieval task based on string similarities, the detection of predicates from natural language is a difficult task. This is mostly due to the large number of expressions that can be used to denote the same predicate. For example, the expressions *X*, *the creator of Y* and *Y is a book by X* are difficult to match by using synset expansion but they both imply that *X* is the author of *Y*. To address this problem, we make use of the pattern library extracted by the BOA framework as presented in Chapter 3 in addition to string matching to detect properties.

This yields a range of different query candidates as potential translations of the input question. It is therefore important to rank those query candidates. To do this, we combine string similarity values, prominence values and schema conformance checks into a score value. The highest ranked queries are then tested against the underlying triple store and the best answer is returned to the user.

6.3 EVALUATION AND DISCUSSION

The evaluation is based on the QALD⁴ benchmark on DBpedia⁵ [Lehmann et al., 2009]. It comprises two sets of 50 questions over DBpedia, annotated with SPARQL queries and answers. Each question is evaluated w.r.t. precision and recall defined as follows:

$$\text{Recall} = \frac{\text{number of correct resources returned by system}}{\text{number of resources in gold standard answer}} \quad (34)$$

$$\text{Precision} = \frac{\text{number of correct resources returned by system}}{\text{number of resources returned by system}} \quad (35)$$

Before we turn to the evaluation results, one important preliminary remark: The reported results are based on natural language questions tagged with ideal part-of-speech information. The reason is that questions often lead to POS tagging errors. For example, in Which films did Leonardo di Caprio star in, the infinitive verb form star is tagged as a noun by the Stanford POS tagger as well as the Apache OpenNLP⁶ POS tagger, which leads to a parse failure. The same holds for a range of infinitives such as play, border, die, cross and start. In order to separate such external errors from errors internal to our approach, we manually corrected erroneous POS tags in seven questions, that otherwise would not have been parsed. This is only a temporal solution, of course; the next step is to train a POS tagger model on a corpus containing a sufficient amount of questions.

EVALUATION RESULTS Of the 50 training questions provided by the QALD benchmark, 11 questions rely on namespaces which we did not incorporate for predicate detection: FOAF⁷ and YAGO⁸. Especially the latter poses a challenge, as YAGO categories tend to be very specific and complex (e.g. FemaleHeadsOfGovernment and HostCitiesOfTheSummerOlympicGames). We did not consider these questions, thus only 39 questions are processed by our approach. Of these 39 questions, 5 questions cannot be

⁴ <http://www.sc.cit-ec.uni-bielefeld.de/qald>

⁵ <http://dbpedia.org>

⁶ <http://incubator.apache.org/opennlp/>

⁷ <http://www.foaf-project.org/>

⁸ <http://www.mpi-inf.mpg.de/yago-naga/yago/>

parsed due to unknown syntactic constructions or uncovered domain-independent expressions. This mainly concerns the noun phrase conjunction as well as and ordinals (the 5th, the first). These constructions will be added in the future; the only reason they were not implemented yet is that they require significant additional effort when specifying their compositional semantics.

Of the remaining 34 questions, 19 are answered exactly as required by the benchmark (i.e. with precision and recall 1.0) and another two are answered almost correctly (with precision and recall > 0.8). Figure 29 (see Appendix) lists the results of each of the 39 processed questions.

The mean of all precision scores is therefore 0.61 and the mean of all recall scores is 0.63, leading to an F-measure⁹ of 0.62. These results are comparable with those of systems such as FREyA and PowerAqua. The key advantage of our system is that the semantic structure of the natural language input is faithfully captured, thus complex questions containing quantifiers, comparatives and superlatives pose no problem, unlike in PowerAqua. Moreover, our system does not need any user feedback, as FREyA does.

PROPERTY DETECTION The utilization of the BOA Pattern Library, as compared to simple string similarity measures, had a huge impact on the overall performance of our approach. Of the 21 questions which were answered (almost) correctly by our system, four questions:

1. Who was Tom Hanks married to?
2. Which people were born in Heraklion?
3. Which books were written by Danielle Steel?
4. Who wrote the book The pillars of the Earth?

could not have been answered without the use of the library. This leads to a significant performance improvement of 19%. In detail, with help of the BOA Pattern Library we are able to close the gap between different word classes/phrases, e.g. “born in” (VP) and “birth place” (NP) or “married to” (VP) and “spouse” (NP), mapping to the same relation. This mapping would not have been possible with for example a Wordnet expansion. Another advantage of the proposed approach is, that it is able to incorporate statistical information into the property detection. For example, it is possible to map “wrote” to *dbo:author* instead of *dbo:writer*, since *dbo:author* occurs statistically more frequently with books and *dbo:writer* with screenplays.

⁹ $(2 \times \text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$

Result	title	author	isbn	literaryGenre	edic	publisher	subsequentWork	author	country
✓	Deception Point	Dan Brown	0-551-15176-4 (US) / 9780551159172 (UK)	Thriller %2B%20Action	52912546	Putnam	The Da Vinci Code	Dan Brown	United States
✗	Digital Fortress	Dan Brown	ISBN 0312180074 (first edition hardcover)	Technothriller	55045700	St. Martin's Press	Angels & Demons	Dan Brown	United Kingdom
✓	The Da Vinci Code	Dan Brown	0-385-50420-9 (US) / 9780385159110 (UK)	Conspiracy Fiction	50000659	Ballantine Books	The Lost Symbol	Dan Brown	United States

Figure 28: Screenshot of prototype available at <http://autosparql-tbsl.dl-learner.org>.

6.4 PROTOTYPE

A prototype for the described algorithm was implemented and deployed, see Figure 28. It is a freely accessible web application, which allows users to enter natural language questions. The answers are shown in a tabular view if appropriate. The view allows users to enrich the generated answers by displaying further appropriate property values for the returned resources. Interesting queries can be saved and reused by other users.

For the prototype, we used DBpedia as underlying knowledge base. To be able to use the mentioned techniques, some components were created offline: Separate indices were created for resources, properties and classes by querying for the labels of those elements in the used DBpedia triple store. Additionally, a BOA index was created for properties, since it vastly improves the mapping of properties in natural language queries compared to using a text index. The same approach can be applied to other knowledge bases and we plan to evaluate this in future work.

6.5 RELATED WORK

Several approaches have been developed for the purpose of question answering.

PowerAqua is a question answering system over Linked Data that is not tailored towards a particular ontology; in particular, it does not make any assumptions about the vocabulary or structure of datasets. The main focus of the system is to combine and merge data from different sources, focusing on scalability, and using iterative algorithms, filtering and ranking heuristics to limit the search space. *PowerAqua* is therefore very strong on large, heterogeneous datasets, although it does struggle on complex mappings such as the aforementioned YAGO categories. For a detailed explanation of the system's architecture and an evaluation see, e.g. [Lopez et al. \[2010, 2011\]](#). The major

shortcoming of PowerAqua is its limited linguistic coverage. In particular, PowerAqua fails on questions containing the most (such as question 31), and more than (such as question 12), which pose no problem for a system with a deeper linguistic analysis of the input question.

Pythia [Unger and Cimiano, 2011] is such a system. It relies on a deep linguistic analysis (on which the approach based in this paper is based) and can therefore handle linguistically complex questions, in particular questions containing determiners such as the most and more than. Pythia's major drawback is that it requires a lexicon, which up to this moment has to be created manually. It therefore fails to scale to very large datasets.

The approach proposed in this paper tries to combine both a deep linguistic analysis with the flexibility of approaches focusing on matching natural language questions to RDF triples. The triple structure is derived from the semantic structure of the question.

Another possibility to determine the triple structure is by exploration of the dataset, as in the question answering system *FREyA* [Damjanovic et al., 2010, 2011]. However, *FREyA* partly relies on the user's help in selecting the entity that is most appropriate as match for some natural language expression. The drawback of such an approach is that the naive end-user is often not informed about the modeling and vocabulary of the data and thus is not able to help.

Further approaches related to question answering over Linked Data include, e.g. *Treo* [Freitas et al., 2011], which combines entity search, semantic relatedness and spreading activation for exploring RDF data, and *Ontolook* [Li et al., 2007], which focuses on relation-based search. In addition to question answering, keyword-based approaches have been gaining momentum over the past years. This led to semantic search engines, such as Swoogle [Ding et al., 2004], Watson [d'Aquin et al., 2008], Sigma [Tummarello et al., 2010] and Sindice [Tummarello et al., 2007], which aim at indexing RDF across the Web and making it available for entity search. The approaches described by Shekarpour et al. [2011] and Tran et al. [2010] extend upon the paradigm of simple entity search and try to generate interpretations of keyword queries which exploit the semantics available on the Linked Data Web. Especially, Tran et al. [2010] implement a graph exploration approach to detect subgraphs of the input knowledge base that can be used to compute an answer to the user's query. On the other hand, Shekarpour et al. [2011] use schema knowledge to infer SPARQL queries that represent possible interpretations of the user-given keywords.

6.6 CONCLUSION

We presented a novel approach to question answering over Linked Data that relies on a deep linguistic analysis yielding a SPARQL template with slots that need to be filled with URIs. In order to fill those

slots, possible entities were identified using string similarity as well as natural language patterns extracted from structured data and text documents. The remaining query candidates were then ranked and, on the basis of scores attached to the entities, one of them was selected as final result.

One of the strengths of this approach is that the generated SPARQL templates capture the semantic structure of the natural language input. Therefore questions containing quantifiers like *the most* and *more than*, comparatives like *higher than* and superlatives like *the highest* do not pose a problem – in contrast to most other question answering systems that map natural language input to purely triple-based representations.

MAPPING TEXT TO ONTOLOGY WITH DBPEDIA LEMON AND BOA

With the huge amount of knowledge available as Linked Data, there is a growing need to make this data accessible for humans in an easy and intuitive way, preferably by means of natural language. One of the main challenges in the development of NLP systems over Linked Data is to determine the exact meaning of a natural language expression. More specifically, there is the problem of entity linking – finding the named entity a mention in a text refers to – as well as the more challenging problem of ontology linking – finding what vocabulary element from an OWL ontology is the target of referral. This work focuses on the second problem. We automatically extend lexical mappings for an OWL ontology from a high-quality seed lexicon. To achieve this goal our approach uses an existing library of natural language patterns of formal relations to extract new lexical mappings. In Section 7.1, we describe the background, i.e. the existing high-quality seed lexicon and the utilized pattern library. Then, we provide insides on how we find, extract and score newly found lexical mappings. Finally, we present preliminary evaluation results, conclude and propose future steps to improve the approach described here.

*This chapter is
mainly based on
Lukovnikov et al.
[2014].*

AUTHOR CONTRIBUTIONS The author of this thesis was a co-author of Lukovnikov et al. [2014] and carried out the training of the BOA framework for the properties contained in the FactBench benchmark used in the evaluation of the proposed approach.

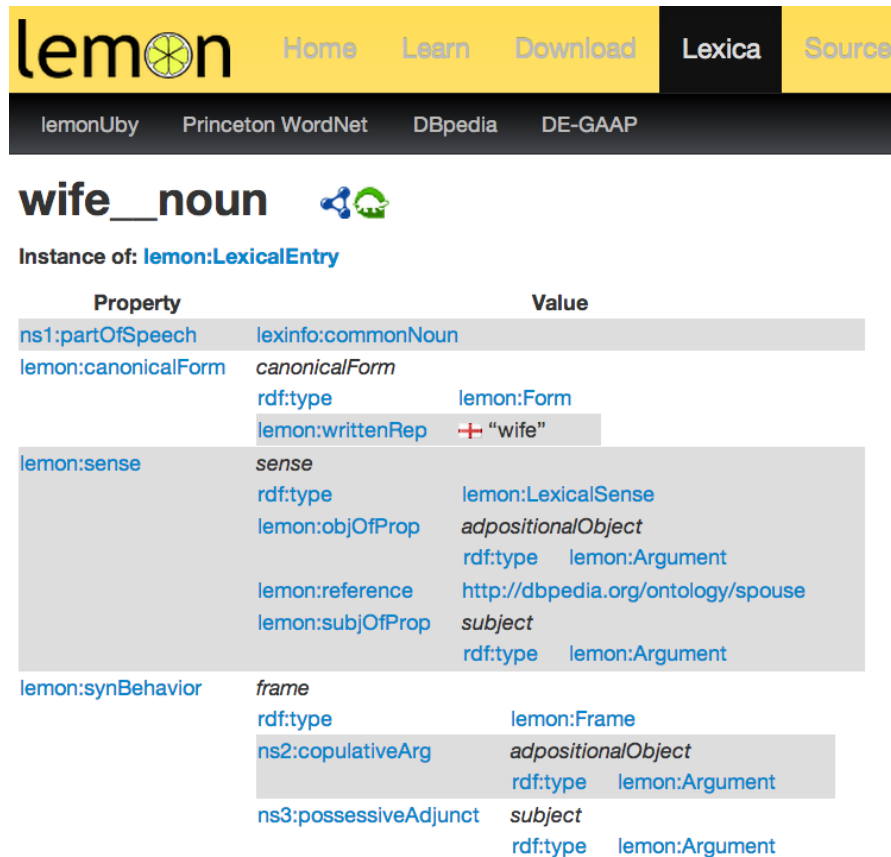
7.1 LEXICAL PATTERN LIBRARY AND SEED LEXICON

As pattern library we use the BOA pattern library, generated by the BOA framework [Gerber and Ngonga Ngomo, 2011, 2012] as presented extensively in Chapter 3, as source for lexical patterns. As seed lexicon we use *lemon*¹ [Unger et al., 2013], a vocabulary for extending an ontology with lexical information describing (among other things) how the elements of the ontology are verbalized in natural language. This information is represented as RDF data, thus making the lexical information a part of the respective ontology and the Linked Data Cloud. The English *lemon* lexicon for DBpedia 3.8² was constructed

¹ Lemon is an abbreviation for LEXicon Model for ONtologies. More information on *lemon* can be found at <http://lemon-model.net/>.

² http://lemon-model.net/lexica/dbpedia_en/

manually. The first version covers about 98% of the ontology classes and 20% of the properties. An example of DBpedia lexicon can be seen in Figure 29.



The screenshot shows the 'lemon' website interface. At the top, there is a navigation bar with 'Home', 'Learn', 'Download', 'Lexica', and 'Source'. Below this, there are links for 'lemonUby', 'Princeton WordNet', 'DBpedia', and 'DE-GAAP'. The main content area displays the entry for 'wife__noun' with a small icon. Below the title, it states 'Instance of: lemon:LexicalEntry'. The core of the page is a table with two columns: 'Property' and 'Value'. The table lists various properties and their corresponding values, including 'lexinfo:commonNoun', 'canonicalForm', 'rdftype', 'lemon:Form', 'lemon:writtenRep', 'sense', 'adpositionalObject', 'lemon:Argument', 'http://dbpedia.org/ontology/spouse', 'subject', 'frame', 'lemon:Frame', 'ns2:copulativeArg', 'ns3:possessiveAdjunct', and 'adpositionalObject'.

Property	Value
ns1:partOfSpeech	lexinfo:commonNoun
lemon:canonicalForm	canonicalForm
rdftype	lemon:Form
lemon:writtenRep	+ "wife"
lemon:sense	sense
rdftype	lemon:LexicalSense
lemon:objOfProp	adpositionalObject
rdftype	lemon:Argument
lemon:reference	http://dbpedia.org/ontology/spouse
lemon:subjOfProp	subject
rdftype	lemon:Argument
lemon:synBehavior	frame
rdftype	lemon:Frame
ns2:copulativeArg	adpositionalObject
rdftype	lemon:Argument
ns3:possessiveAdjunct	subject
rdftype	lemon:Argument

Figure 29: Overview of DBpedia lemon lexicon for word “wife”.

Whereas entity linking is a straightforward problem in NLP as systems like AIDA [Hoffart et al., 2011] and AGDISTIS [Usbeck et al., 2014] show, ontology linking poses more challenges. A baseline approach for ontology linking is using text similarity of a given token with the label of an ontology element to estimate the probability that a word is referring to this ontology element. However, a drawback of this method is, that it is not able to deal with synonyms. For example, the word “spouse” can easily be matched to the property `dbpedia-owl:spouse`, since the label of the property is “spouse”. But “wife”, “husband” and “married to” also refer to this property but textual similarity-based approaches often fail to find those links.

A more sophisticated approach for ontology linking is using a lexicalization dictionary, such as the DBpedia lemon lexicon. With the help of such a lexicon, it is simpler (polysemous words excluded) to generate mappings from surface forms to ontology elements. However, there are several issues hindering the widespread usage of the first version of the DBpedia lexicon for ontology linking:

1. it only covers 20% of the properties available in DBpedia,
2. the manual generation of this lexicon is very expensive,
3. it does not cover properties that are not in the DBpedia OWL ontology but can be found in DBpedia (especially OWL properties in the <http://dbpedia.org/property/> namespace) and
4. there are no statistics about ontology verbalization, which can be helpful to improve ontology linking.

7.2 COMBINING LEMON SEEDS AND BOA

BOA patterns can be used to automatically expand the coverage of the DBpedia-lemon lexicon to verbalize more DBpedia OWL properties. Additionally, the patterns can be used to generate a lexicon for ontology elements found on DBpedia (and BOA patterns) that are not in the DBpedia ontology. Furthermore we can leverage the statistical information associated with BOA patterns to add statistical information to lexicon elements. The input of our approach is a set of BOA patterns as well as the DBpedia lemon lexicon. We then use the lexicon entries to find mapping extraction patterns which will subsequently be used to extract new lexical information.

7.2.1 Finding mapping extraction patterns

Consider the *lemon* lexicon for DBpedia as a set \mathcal{L} of pairs of mappings from text to DBpedia ontology properties. For example, $l = (\text{word}, \text{uri}) \in \mathcal{L}$, where *word* is an entry of the lexicon and *uri* is the URI of a given property, could look as follows:

$$l = (\text{wife}, \text{spouse}).$$

Consider the BOA patterns as a set of patterns \mathfrak{P} , where each pattern p maps a natural language text to a RDF property. For example, $p = (\text{uri}, \theta, \text{lang})$, where *uri* is the URI of the property, θ is its natural language representation and *lang* is the language of the NLR, could be instantiated like this:

$$p = (\text{spouse}, \text{"R is the wife of D"}, \text{en})$$

We iterate over all mappings $l \in \mathcal{L}$ and for each l , we search \mathfrak{P} to find all patterns matching the mapping l , giving us \mathfrak{P}_l . The matching function currently used is simply checking whether the surface form of the mapping ("wife") occurs in the BOA pattern string ("R is the wife of D") and whether the property provided by the mapping is the same as the property of the BOA pattern. For each BOA pattern in \mathfrak{P}_l , we extract the meta-pattern by substituting the matched surface form l in the pattern by a property variable X .

Note that lang is always "en" (English) in our experiments.

Then, if not already present, each meta-pattern is added to the set of meta-patterns $\mathfrak{M}\mathfrak{P}$. If the meta-pattern is already in $\mathfrak{M}\mathfrak{P}$, we update its support. An example meta-pattern that could be extracted from the above example lexicalization of the `dbo:spouse` property is:

R is the X of D.

7.2.2 *Extracting new lexical mappings*

In this phase, we want to extract new lexical mappings and score existing ones. Here, we take $\mathfrak{M}\mathfrak{P}$ and \mathfrak{P} and search for new elements for \mathcal{L} .

To this end, for every BOA pattern $p \in \mathfrak{P}$ we find the matching meta-patterns using simple pattern matching with regular expressions. From the possibly applicable meta-patterns, the best meta-pattern is selected and used to extract the lexical mapping. If the mapping is already present in \mathcal{L} , we update the scores, otherwise the mapping is added to \mathcal{L} . The selection of the best meta-pattern from all meta-patterns applicable to BOA patterns is done by choosing the most surface form-specific meta-pattern. We define the most specific pattern to be the pattern producing the shortest surface form in the lexical mapping.

As an example illustrating the generation of new mappings in this phase, consider the BOA pattern:

$$p = (\text{spouse}, \text{"R is the consort of D"}, \text{en})$$

and the meta-pattern extracted from the example of the previous subsection:

$$\text{mp}_1 = (\text{spouse}, \text{"R is the X of D"}, \text{en})$$

However, another (more general) meta-pattern could have been extracted in the first phase:

$$\text{mp}_2 = (\text{spouse}, \text{"R X D"}, \text{en})$$

Both meta-patterns could match the BOA pattern, but the first one is more specific because it produces a shorter surface form ("consort") than the second meta-pattern would produce ("is the consort of"). Therefore, the first meta-pattern is applied to the BOA pattern to produce the mapping:

$$l_1 = (\text{consort}, \text{spouse}).$$

Notice that the second (more general) meta-pattern would produce the mapping

$$\text{"is the consort of"} \rightarrow \text{dbo:spouse } l_2 = (\text{is the consort of}, \text{spouse}),$$

which is worse than the mapping produced by the more specific meta-pattern, if not wrong for the purposes of lexicon construction.

7.3 EVALUATION

The methodology described above was tested using a sample dataset containing BOA patterns of 10 properties (spouse, award, team, author, subsidiary, starring, deathPlace, birthPlace, leaderName and foundationPlace). Only 7 of the 10 properties in the sample BOA dataset had verbalizations in the DBpedia lexicon. In total, the lexicon contains 17 relevant verbalizations. Below, the preliminary evaluation results of the prototype implementation are described.

Applied to the given data, our method found 119 meta-patterns and 5,274 lexical mappings in one iteration. This initial set of found lexical mappings contains a lot of noisy mappings. A fraction of the noisy mappings could still be useful in NLP applications (for example "goalkeeper" → `dbo:team`), but it would be wrong to include them in the lexicon. In the future, we will implement a filtering strategy to discard noisy mappings.

We performed a preliminary evaluation of the method by looking at how many of the verbalizations present in the DBpedia lexicon were found by our method. We used the sample BOA dataset with 10 properties which give us 17 mappings from the DBpedia lexicon, as discussed in the first paragraph of this section. With stemming on the surface forms to counter minor variations (verb conjugations, multiplicity), 8 of the 17 mappings were also found by our method. Thus, we can state that the recall of the prototype implementation applied to BOA patterns of 10 properties is 47%.

7.4 CONCLUSION AND FUTURE WORK

The proposed method is an interesting use case of BOA patterns. The mappings extracted from the manual DBpedia lexicon can already be useful for NLP purposes. The approach for automatic generation of new mappings can extend the coverage of these kinds of lexicons. The proposed method for automatic extraction of mappings can be extended to an approach for automatic lexicon generation.

Adding the scores derived from BOA and our method improves the usability of the extended lexical mappings for NLP purposes. For example, the commonness score of a mapping indicates how often the surface form is used to refer to the given ontology element, with respect to other ontology elements that can be referred to by the same surface form.

Preliminary evaluation of the proposed method using a sample BOA pattern dataset indicates acceptable recall. However, it should be noted that the preliminary evaluation was done on a small dataset. The method should perform better on bigger datasets.

Here, the prototype implementation and its preliminary evaluation are presented. Future work includes improving the scoring system for

better estimation of the relevance of (meta-)patterns and mappings and improvement of the specificity criterium for meta-pattern selection in the second phase. Using POS tags in the pattern matching process should reduce the number of generated noisy mappings dramatically. The ultimate goal is a method for automated expansion of ontology lexica.

CONCLUSION AND FUTURE WORK

The goal of the algorithms and applications presented within this thesis was to provide solutions towards closing the gap between the Semantic Web and the Web 2.0. We identified four main drawbacks of the Semantic Web: *completeness*, *actuality*, *provenance* and *quality* (see Section 1.2). The algorithms shown in the first part of this thesis focused on the completeness and actuality problems, whereas the applications depicted in the second part – building on top of the proposed algorithms – dealt with provenance and quality issues. In the following paragraphs, which also correspond to this thesis' structure, we will summarize the proposed algorithms and applications and highlight the impact on bridging the aforementioned gap between Web 2.0 and the Semantic Web.

8.1 SUMMARY

BOA - BOOTSTRAPPING LINKED DATA To tackle the problem of completeness, i.e. that most data available on the Data Web was extracted from semi- or structured knowledge sources, we implemented a relation extraction algorithm to be able to also extract knowledge from unstructured sources. Relation extraction is an enormously difficult task, since it typically builds upon a machine learning pipeline where the erroneous output of a module in the pipeline being able to significantly deteriorate the performance of the next module within that pipeline and so on. We built our algorithms on top of state-of-the-art algorithms for Sentence Boundary Disambiguation, Named Entity Recognition, Part-of-Speech Tagging, Named Entity Disambiguation and others. The task gets even more complex due to the prerequisite of being able to process data at web-scale. To solve these requirements, we proposed the BOA framework, an iterative distant supervision algorithm, where we use instance data already available on the Data Web to generate new instance and schema knowledge. This knowledge can be fed back into the Data Web and be used as background knowledge in subsequent runs of the framework. BOA generates a mapping between formal relations found on the Data Web and their natural language representation found in the Document Web by extracting sentences from natural language text that contain both named entities and patterns. After the implementation and evaluation we continuously improved our approach to cover a broader range of languages (English, German, French), updated the pattern scoring algorithms and the Named Entity Disambiguations

techniques, incorporated surface forms for named entities and finally implemented a multilingual pattern generalization approach in order to further increase precision and recall. Note that our approach can thus be used on most languages whose grammar adheres roughly to the subject - predicate - object sentence structure. The potential of the approach presented herein is immense, as it could promote the Data Web to the lingua franca for a large number of applications including machine translation, named entity recognition and question answering.

RDFLIVENEWS So far, up-to-date data has, with exceptions like DBpedia Live, been mostly neglected by the Semantic Web research community. RDF on the Data Web has mostly been extracted from database dumps, which typically are weeks or even months old (see Section 1.2). With RdfLiveNews, we presented a framework for the extraction of RDF from unstructured data streams on web scale. In our experimental setting we used newspaper RSS feeds as data streams and sentences as data stream elements. We performed an Open Relation Extraction task on those sentences and extracted patterns representing relations between entities. Then, these patterns were grouped and could be refined with the help of the information already available on the Data Web (finding a suitable *rdfs:range* and *rdfs:domain* with different strategies). We implemented and evaluated multiple similarity functions for the clustering and have been able to link clusters, or the properties within these clusters respectively, to properties available on the Data Web. We were able to disambiguate resources with a precision of 85%, cluster patterns with an accuracy of 82.5% and extract RDF with a total accuracy of around 90%. Additionally, RdfLiveNews can handle two hour time slices with around 300.000 sentences within 20 minutes on a small server.

DEFACTO The number of knowledge bases on the Data Web and the triples contained therein are continuously rising. One of the main tasks of curators and maintainers of these knowledge bases is to (in)validate facts and provide sources for them in order to ensure correctness and traceability of the provided knowledge. With an estimated number of 89 billion triples on the Data Web, this task cannot be carried out by humans alone. To support human curators in this task we presented DeFacto, an approach for checking the validity of RDF triples using the Web as corpus. The DeFacto baseline uses English natural language representations generated by the BOA framework to create and execute search engine queries. These queries return websites containing both entities of the given triple *and* the relation in natural language. We extract all of those phrases and apply a two-step machine learning task, in which we first test if the given phrase validates the input fact and then aggregate all phrases, com-

bined with the trustworthiness of the websites they are contained in, to a total evidence score. The evaluation of DeFacto for 60 properties found on the Data Web lead to a promising F_1 measure of 0.843. Despite English being the dominant language on the Web (see Figure 7), a large part of information is only available in some languages. In order to also extract these highly localized information, we extended the baseline version of DeFacto to recognize German and French natural language representations as well. Additionally, a large number of facts is only valid for a certain time span. For example, the fact that Albert Einstein was the spouse of Mileva Marić, does not hold true today but only for the time span between 1903 and 1919. The extended version of DeFacto implements three strategies to determine the time scope of a given fact. We support the endeavor of creating better fact validation algorithms (and to that end also better relation extraction systems) by providing the full-fledged and timely scoped FactBench benchmark. We showed that our approach achieves an F_1 measure of 84.9% on the most realistic fact validation test set (FactBench *mix*) on DBpedia as well as Freebase data. The temporal scoping module shows a promising average F_1 measure of 70.2% for time point and 65.8% for time period relations. Finally, we were able to show that multilingual fact validation increased the overall and significantly increased the date scoping performance with respect to the English only version.

TEMPLATE-BASED QUESTION ANSWERING To enable lay users to tap into the wealth of data becoming available on the Data Web every day, developing intuitive ways for accessing this data becomes increasingly important. Natural language question answering systems can be considered as a compromise between intuitiveness and expressivity since these approaches are able to exploit the expressiveness of the underlying data model and query language, while hiding their complexity from the user. Therefore we presented a novel approach to question answering over Linked Data that relies on a deep linguistic analysis yielding a SPARQL template with slots that need to be filled with URIs. This captures the semantic structure of the natural language input and allows to deal with quantifiers (*the most, more than*) comparatives (*higher than*) and superlatives (*the highest*) easily. We have exploited the BOA pattern library to fill the property slots of the SPARQL templates with URIs from the Data Web. This was especially useful if string or WordNet similarity metrics could not provide the correct mapping (e.g. *was married to* and *spouse*). The utilization of the BOA pattern library lead to a significant increase of the overall performance of the QA-System by 19% F_1 measure.

8.2 FUTURE WORK

There is a long way to go for the Semantic Web to be as successful as the Web 2.0 currently is. The algorithms and applications presented within this thesis help to bridge this gap significantly. They can be extended in manifold ways to bring the Semantic Web even closer to the vision postulated by its inventor.

BOA - BOOTSTRAPPING LINKED DATA In future work, we will aim at applying our approach to Asian languages whose grammar differ completely from that of the language we processed so far. Preliminary work on the Korean language showed very promising results. In addition, we will consider the use of crowd-sourcing to improve our scoring approach. Furthermore, we can deploy our approach on larger data sets such as ClueWeb09/12¹ to discover even more facts and to extract more confident patterns. Finally, we will integrate the temporal scoping module of the DeFacto framework to take temporal markers into consideration so as to be able to process predicates such as `dbpedia:formerTeam` or `dbpedia:formerBandMember`.

RDFLIVENEWS For future versions of RdfLiveNews we plan to go beyond text streams and apply our work to audio and video streams. We will extend our approach to support additional languages and to also cover datatype properties. For example, from the sentence "... , Google said Motorola Mobility contributed revenue of US\$ 1.25 billion for the second quarter" the triple `dbpedia:Motorola_Mobility rln:revenue 1.250.000.000` can be extracted. Furthermore, we could extract the triple `dbpedia:Google rln:says "Motorola Mobility contributed revenue of US$ 1.25 billion for the second quarter"` which would lead to a worldwide machine readable library of quotes from persons or organizations respectively. Additionally, we plan to integrate DeFacto to be able to verify/falsify the extracted triple from other news sources and enrich it with additional provenance information. Finally, we will extend our approach with temporal logics to explicate the temporal scope of the triples included in our knowledge base.

DEFACTO Our approach can be extended in manifold ways. First, we could run the experiments on a Web crawl such as ClueWeb09/ClueWeb12 or CommonCrawl.² This would drastically increase recall, since we could execute all combinations of subject/object surface forms and patterns as well as precision, since we could also query for exact matches like "Albert Einstein was awarded the Nobel Prize in Physics" as opposed to querying for fragments (see Section 5.5.2).

¹ <http://lemurproject.org/clueweb12>

² <http://commoncrawl.org/>

Second, we could work on efficient disambiguation of (parts of) the webpage’s text before extracting proof phrases. This would be useful for, e.g. differentiating between Winston Churchill, the American novelist, and Winston Churchill, the British prime minister. Furthermore, we could extend our approach to support data type properties or try to search for negative evidence for facts, therewith allowing users to have a richer view of the data on the Web through DeFacto. Finally, we could extend the user interface (see Figure 22) to improve classifier performance by incorporating a feedback loop allowing users to vote on overall results, as well as proofs found on webpages. This feedback can then be fed into our overall machine learning pipeline and improve DeFacto on subsequent runs.

TEMPLATE-BASED QUESTION ANSWERING One of the strengths of this approach is that the generated SPARQL templates capture the semantic structure of the natural language input. However, in some cases the semantic structure of the question and the triple structure of the query do not coincide. Thus, faithfully capturing the semantic structure of the input question sometimes leads to too rigid templates. We are currently exploring two approaches to solve this problem. The first one concentrates on more flexible processing. On the one hand, we are considering a preprocessing step that can detect complex (especially YAGO) categories before parsing the natural language question. On the other hand, we are investigating the relaxation of templates, in such a way that the triple structure is not completely fixed but is discovered through exploration of the RDF data.

The second approach concerns incorporating a more flexible fallback strategy in case no successful SPARQL query is found. In particular, we are working on combining our approach with active learning methods as described by [Lehmann and Bühmann \[2011\]](#). Ultimately, our goal is to provide robust question answering for large scale heterogeneous knowledge bases. Our vision is that this robustness can help to make the usage of question answering systems a standard task in everyday life in a similar but more powerful way as web search.

Henry Ford, the famous American industrialist and founder of the Ford Motor Company, is said to have once said:

“If I had asked people what they wanted, they would have said faster horses.”

If we alter this quote to fit the problems postulated in Section 1.2, then *Tim Berners-Lee* could have said something along the lines of: “*If I had asked people what they wanted, they would have said more intelligent text search.*”. So if the Semantic Web research community is able to solve the problems highlighted in this thesis, the Semantic Web might very well be the next automobile.

Part III

APPENDIX



APPENDIX

A.1 TEMPLATE BASED QUESTION ANSWERING

id	question	precision	recall
2	Who has been the 5th president of the United States of America		
4	Who was Tom Hanks married to	1.0	1.0
5	Which people were born in Heraklion	0.91	1.0
7	Which companies work in the aerospace industry as well as on nuclear reactor technology		
8	Which people have as their given name Jimmy		
9	Who developed the video game World of Warcraft	1.0	1.0
10	Who was the wife of president Lincoln	1.0	1.0
12	Which caves have more than 3 entrances	1.0	1.0
13	Which cities have more than 2000000 inhabitants	0.04	0.26
14	Who owns Aldi		
16	Give me all soccer clubs in the Premier League	0.5	0.86
17	In which programming language is GIMP written	1.0	1.0
18	What languages are spoken in Estonia	1.0	0.14
20	Which country does the Airedale Terrier come from	1.0	1.0
21	What is the highest mountain	1.0	1.0
24	Which organizations were founded in 1950	0.0	0.0
25	Which genre does DBpedia belong to	1.0	1.0
26	When was DBpedia released	1.0	1.0
27	Who created English Wikipedia	1.0	1.0
28	Which companies are located in California USA	0.8	0.76

30	How many films did Leonardo DiCaprio star in	1.0	1.0
31	Who produced the most films	1.0	1.0
32	Is Christian Bale starring in Batman Begins	1.0	1.0
33	Which music albums contain the song Last Christmas		
34	Give me all films produced by Hal Roach	1.0	1.0
35	Give me all actors starring in Batman Begins	1.0	0.86
36	Give me all movies with Tom Cruise	0.08	0.75
37	List all episodes of the first season of the HBO television series The Sopranos		
38	Which books were written by Danielle Steel	1.0	1.0
39	Who wrote the book The pillars of the Earth	0.5	1.0
40	Which mountains are higher than the Nanga Parbat	0.0	0.0
41	When was Capcom founded	1.0	1.0
42	Which software has been published by Mean Hamster Software	1.0	1.0
43	Is there a video game called Battle Chess	0.0	0.0
44	Which software has been developed by organizations founded in California		
45	Which country has the most official languages	0.0	0.0
47	Is Natalie Portman an actress	1.0	1.0
48	Who produced films starring Natalie Portman	1.0	1.0
49	In which films did Julia Roberts as well as Richard Gere play		

Table 29: This table shows precision and recall values for each processed question (i.e. all questions that do not require the YAGO or FOAF namespace). For questions with no precision and recall specified, no query was constructed. Questions printed in **cells with red background** were not parsed, questions in **white cells** succeeded and for questions in **lightgray cells** queries with quality equal or close to the Gold query were built, while questions in **yellow cells** fail due to a query selection problem and questions in **orange cells** fail due to some entity identification problem.

BIBLIOGRAPHY

- Adrian, B., Hees, J., Herman, I., Sintek, M., and Dengel, A. (2010). Epiphany: Adaptable RDFa Generation Linking the Web of Documents to the Web of Data. In *EKAW*, pages 178–192. (Cited on page 23.)
- Agichtein, E. and Gravano, L. (2000). Snowball: Extracting Relations from Large Plain-Text Collections. In *In ACM DL*, pages 85–94. (Cited on pages 22, 25, and 79.)
- Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2008). DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer. (Cited on page 80.)
- Auer, S., Lehmann, J., and Hellmann, S. (2009). LinkedGeoData - Adding a Spatial Dimension to the Web of Data. In *Proceedings of 8th International Semantic Web Conference*. (Cited on pages 4 and 19.)
- Auer, S., Lehmann, J., and Ngomo, A.-C. N. (2011). Introduction to Linked Data and Its Lifecycle on the Web. In *Reasoning Web*, pages 1–75. (Cited on pages 19 and 55.)
- Augenstein, I., Padó, S., and Rudolph, S. (2012). LODifier: Generating Linked Data from Unstructured Text. In *ESWC*, volume 7295, pages 210–224. (Cited on pages 68 and 79.)
- Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open Information Extraction from the Web. In *International Joint Conferences on Artificial Intelligence*, pages 2670–2676. (Cited on page 26.)
- Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2004). OWL Web Ontology Language Reference. W3C Recommendation. Last access on Dez 2008 at: <http://www.w3.org/TR/owl-ref/>. (Cited on page 12.)
- Beckett, D. (10. Februar 2004). RDF/XML Syntax Specification (Revised). W3C Recommendation. (Cited on pages 11 and 12.)
- Beckett, D. (24. Februar 2014). RDF 1.1 XML Syntax. W3C Recommendation. (Cited on page 12.)
- Beckett, D. and Berners-Lee, T. (2008). Turtle - Terse RDF Triple Language. W3C Team Submission. (Cited on page 12.)

- Beckett, D. and Broekstra, J. (2008). SPARQL Query Results XML Format. W3C Recommendation. (Cited on page 13.)
- Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. (2012). PROV-O: The PROV Ontology. Technical report. (Cited on pages 6 and 82.)
- Berners-Lee, T. (1998). Cool URIs don't change. (Cited on page 11.)
- Berners-Lee, T., Fielding, R., and Masinter, L. (2005). RFC 3986, Uniform Resource Identifier (URI): Generic Syntax. Request For Comments (RFC). (Cited on pages xvi and 10.)
- Berners-Lee, T. and Fischetti, M. (1999). *Weaving the web: The original design and ultimate destiny of the world wide web by its inventor*. Harper, San Francisco. (Cited on page 3.)
- Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34–43. (Cited on pages 3, 9, 10, and 55.)
- Biemann, C., Heyer, G., Quasthoff, U., and Richter, M. (2007). The Leipzig Corpora Collection – Monolingual Corpora of Standard Size. In *Proceedings of the 4th Conference on Corpus Linguistics (CL)*. (Cited on pages 29 and 35.)
- Blumberg, R. and Atre, S. (2003). The problem with unstructured data. *DM Review*, 13(February 2003):42–49. (Cited on page 19.)
- Brickley, D. and Guha, R. (10. Februar 2004). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. (Cited on page 11.)
- Brin, S. (1999). Extracting Patterns and Relations from the World Wide Web. In *WebDB*, pages 172–183. (Cited on pages 22, 25, and 79.)
- Brohée, S. and van Helden, J. (2006). Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*. (Cited on page 65.)
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., and Mitchell, T. M. (2010). Toward an Architecture for Never-Ending Language Learning. In *AAAI*. (Cited on pages 20, 23, 26, 34, 38, 48, and 79.)
- Carnielli, W. A. and Marcos, J. (2001). Ex contradictione non sequitur quodlibet. In *Proceedings of the II Annual Conference on Reasoning and Logic, held in Bucharest, RO, July 2000*, pages 89–109. (Cited on page 6.)
- Carothers, G. and Seaborne, A. (2014). RDF 1.1 N-Triples. W3C recommendation, W3C. <http://www.w3.org/TR/2014/REC-n-triples-20140225/>. (Cited on page 12.)

- Clark, K. G., Feigenbaum, L., and Torres, E. (2008). SPARQL Protocol for RDF. W₃C Recommendation. (Cited on page 13.)
- Correndo, G., Salvadores, M., Millard, I., and Shadbolt, N. (2010). Linked Timelines: Temporal Representation and Management in Linked Data. In *COLD*, volume 665. (Cited on page 105.)
- Curran, J. R. and Clark, S. (2003). Language independent NER using a maximum entropy tagger. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, pages 164–167, Morristown, NJ, USA. Association for Computational Linguistics. (Cited on page 23.)
- Damljanovic, D., Agatonovic, M., and Cunningham, H. (2010). Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010), Heraklion, Greece, May 31-June 3, 2010*. Springer. (Cited on pages 119 and 125.)
- Damljanovic, D., Agatonovic, M., and Cunningham, H. (2011). FREyA: An interactive way of querying Linked Data using natural language. In *Proceedings of the 1st Workshop on Question Answering over Linked Data (QALD-1), ESWC 2011*. (Cited on page 125.)
- d’Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V., and Guidi, D. (2008). Toward a New Generation of Semantic Web Applications. *Intelligent Systems, IEEE*, 23(3):20–28. (Cited on page 125.)
- Davidov, D. and Rappoport, A. (2008). Classification of Semantic Relationships between Nominals Using Pattern Clusters. ACL. (Cited on page 69.)
- DCMI Usage Board (2006). DCMI Metadata Terms. DCMI recommendation, Dublin Core Metadata Initiative. Published online on December 18th, 2006 at <http://dublincore.org/documents/2006/12/18/dcmi-terms/>. (Cited on page 6.)
- Demter, J., Auer, S., Martin, M., and Lehmann, J. (2012). LODStats – An Extensible Framework for High-performance Dataset Analytics. Lecture Notes in Computer Science (LNCS) 7603. Springer. (Cited on page 6.)
- Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V., and Sachs, J. (2004). Swoogle: a search and metadata engine for the Semantic Web. In Grossman, D. A., Gravano, L., Zhai, C., Herzog, O., and Evans, D. A., editors, *CIKM*, pages 652–659. ACM. (Cited on page 125.)

- Dividino, R., Sizov, S., Staab, S., and Schueler, B. (2011). Querying for Provenance, Trust, Uncertainty and other Meta Knowledge in RDF. *Web Semantics: Science, Services and Agents on the World Wide Web*. (Cited on page 78.)
- Dong, X. L., Berti-Equille, L., and Srivastava, D. (2009). Truth Discovery and Copying Detection in a Dynamic World. *PVLDB*, 2:562–573. (Cited on page 78.)
- E. Kaufmann, A. Bernstein, L. F. (2007). NLP-Reduce: A "naive" but domain-independent natural language interface for querying ontologies. In *Proceedings of the 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria*. (Cited on page 119.)
- Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2004). Web-scale Information Extraction in KnowItAll. In *Proceedings of World Wide Web Conference 2004*. (Cited on pages 22 and 25.)
- Exner, P. and Nugues, P. (2012). Entity extraction: From unstructured text to DBpedia RDF triples. In Rizzo, G., Mendes, P., Charton, E., Hellmann, S., and Kalyanpur, A., editors, *Web of Linked Entities Workshop (WoLE 2012)*. (Cited on page 68.)
- Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying Relations for Open Information Extraction. In *In Proceedings Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. (Cited on pages 26, 46, and 69.)
- Fellbaum, C., editor (1998). *Wordnet, an Electronic Lexical Database*. MIT Press. (Cited on pages 5, 14, and 46.)
- Ferrucci, D. A., Brown, E. W., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J. M., Schlaefel, N., and Welty, C. A. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79. (Cited on page 5.)
- Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL, ACL '05*, pages 363–370. (Cited on page 23.)
- Finkel, J. R. and Manning, C. D. (2010). Hierarchical joint learning: improving joint parsing and named entity recognition with non-jointly labeled data. In *ACL '10*, pages 720–728. (Cited on page 22.)
- Freitas, A., de Oliveira, J., O’Riain, S., Curry, E., and da Silva, J. P. (2011). Querying Linked Data using Semantic Relatedness: A Vocabulary Independent Approach. In *Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems (NLDB)*. (Cited on page 125.)

- Gaag, A., Kohn, A., and Lindemann, U. (2009). Function-based solution retrieval and semantic search in mechanical engineering. In *IDEC '09*, pages 147–158. (Cited on pages 19 and 55.)
- Galland, A., Abiteboul, S., Marian, A., and Senellart, P. (2010). Corroborating information from disagreeing views. In *WSDM*, pages 131–140. ACM. (Cited on page 78.)
- Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngonga Ngomo, A.-C., and Speck, R. (2015). DeFacto - Temporal and Multilingual Deep Fact Validation. *Journal of Web Semantics*. (Cited on pages vii, 21, 73, and 77.)
- Gerber, D., Hellmann, S., Bühmann, L., Soru, T., and Ngomo, A.-C. N. (2013). Real-time RDF extraction from unstructured data streams. In *Proceedings of ISWC*. (Cited on pages vii, 27, 54, and 56.)
- Gerber, D. and Ngonga Ngomo, A. (2012). Extracting Multilingual Natural Language Patterns for RDF Predicates. In *EKAW, Lecture Notes in Computer Science*, pages 87–96. Springer. (Cited on pages vii, 18, 21, 79, 81, and 127.)
- Gerber, D. and Ngonga Ngomo, A. (2013). *From RDF to Natural Language and Back*. Springer. (Cited on pages viii, 18, and 21.)
- Gerber, D. and Ngonga Ngomo, A.-C. (2011). Bootstrapping the Linked Data Web. In *1st Workshop on Web Scale Knowledge Extraction @ ISWC*. (Cited on pages viii, 18, 21, 27, 51, 69, 81, and 127.)
- Goldhahn, D., Eckart, T., and Quasthoff, U. (2012). Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. In *LREC*. (Cited on page 63.)
- Grant, J. and Beckett, D. (10. Februar 2004). RDF Test Cases. W3C Recommendation. (Cited on page 11.)
- Grishman, R. and Yangarber, R. (1998). Nyu: Description of the Proteus/Pet system as used for MUC-7 ST. In *MUC-7*. Morgan Kaufmann. (Cited on page 79.)
- Guha, R. and Brickley, D. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. W3C recommendation, W3C. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>. (Cited on page 12.)
- Guha, R. and Brickley, D. (2014). RDF Schema 1.1. W3C recommendation, W3C. <http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>. (Cited on page 12.)
- Gutierrez, C., Hurtado, C., and Vaisman, A. (2005). Temporal RDF. In *The Semantic Web: Research and Applications*, pages 93–107. Springer. (Cited on page 79.)

- Hartig, O. (2008). Trustworthiness of Data on the Web. In *Proceedings of the STI Berlin & CSW PhD Workshop*. (Cited on page 78.)
- Hartig, O. (2009). Provenance Information in the Web of Data. In *Proceedings of Linked Data on the Web Workshop at World Wide Web Conference*. (Cited on pages 5 and 74.)
- Hartig, O. and Zhao, J. (2010). Publishing and Consuming Provenance Metadata on the Web of Linked Data. In *IPAW*, pages 78–90. (Cited on page 78.)
- Hayes, P. and McBride, B. (10. Februar 2004). RDF Semantics. (Cited on page 11.)
- Hearst, M. A. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545. (Cited on pages 22 and 25.)
- Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. (Cited on page 19.)
- Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating NLP using Linked Data. In *12th International Semantic Web Conference*. (Cited on page 82.)
- Hendler, J. and Golbeck, J. (2008). Metcalfe’s law, Web 2.0, and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):14–20. (Cited on page 7.)
- Hitzler, P., Krötzsch, M., Rudolph, S., and Sure, Y. (2008). *Semantic Web, Grundlagen*. Springer. (Cited on page 12.)
- Hoffart, J., Suchanek, F. M., Berberich, K., and Weikum, G. (2013). YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. In Rossi, F., editor, *IJCAI*. IJCAI/AAAI. (Cited on page 15.)
- Hoffart, J., Yosef, M. A., Bordino, I., Fürstenaу, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP ’11*, pages 782–792. (Cited on pages 64, 80, and 128.)
- Höffner, K., Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.-C. N., Gerber, D., and Cimiano, P. (2013). TBSL Question Answering System Demo. In *Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web*. (Cited on pages vii, 118, and 121.)
- Hogan, A., Harth, A., Passant, A., Decker, S., and Polleres, A. (2010). Weaving the Pedantic Web. In *Linked Data on the Web Workshop*

- (LDOW2010) at WWW'2010, volume 628, pages 30–34. CEUR Workshop Proceedings. (Cited on page 6.)
- Huynh, D., Mazzocchi, S., and Karger, D. R. (2005). Piggy Bank: Experience the Semantic Web Inside Your Web Browser. In *ISWC*, pages 413–430. (Cited on page 23.)
- Jiang, J. J. and Conrath, D. W. (1997). Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *International Conference Research on Computational Linguistics*, pages 9008+. (Cited on page 46.)
- Kim, S. N., Medelyan, O., Kan, M.-Y., and Baldwin, T. (2010). SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *SemEval '10*, pages 21–26. (Cited on page 22.)
- Kleinberg, J. M. (1999). Hubs, authorities, and communities. *ACM Comput. Surv.* (Cited on page 77.)
- Klyne, G. and Carroll, J. J. (10. Februar 2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. (Cited on page 11.)
- Krause, S., Li, H., Uszkoreit, H., and Xu, F. (2012). Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web. In *International Semantic Web Conference*, volume 7649 of *Lecture Notes in Computer Science*, pages 263–278. (Cited on pages 23, 27, and 80.)
- Lanthaler, M., Cyganiak, R., and Wood, D. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C recommendation, W3C. <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>. (Cited on page 11.)
- Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). DBpedia - A Crystallization Point for the Web of Data. *Journal of Web Semantics*, 7(3):154–165. (Cited on pages 14, 85, 96, and 122.)
- Lehmann, J. and Böhmann, L. (2011). AutoSPARQL: Let Users Query Your Knowledge Base. In *Proceedings of ESWC 2011*, volume 6643 of *Lecture Notes in Computer Science*, pages 63–79. (Cited on page 137.)
- Lehmann, J., Furche, T., Grasso, G., Ngonga Ngomo, A.-C., Schallhart, C., Sellers, A., Unger, C., Böhmann, L., Gerber, D., Höffner, K., Liu, D., and Auer, S. (2012a). DEQA: Deep Web Extraction for Question Answering. In *Proceedings of ISWC*. (Cited on pages vii and 121.)
- Lehmann, J., Gerber, D., Morsey, M., and Ngonga Ngomo, A.-C. (2012b). DeFacto - Deep Fact Validation. In *ISWC*. (Cited on pages vii, 73, 77, 84, and 85.)

- Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2013). DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*. Under review. (Cited on pages 4, 14, 19, and 96.)
- Lenat, D. (1995). CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38. (Cited on pages 5 and 14.)
- Li, Y., Wang, Y., and Huang, X. (2007). A Relation-Based Search Engine in Semantic Web. *IEEE Trans. Knowl. Data Eng.*, 19(2):273–282. (Cited on page 125.)
- Lin, D. (1998). An Information-Theoretic Definition of Similarity. In Shavlik, J. W. and Shavlik, J. W., editors, *ICML*, pages 296–304. Morgan Kaufmann. (Cited on page 60.)
- Lopez, V., Fernandez, M., Motta, E., and Stieler, N. (In Press (2011)). PowerAqua: Supporting Users in Querying and Exploring the Semantic Web. *Semantic Web Journal*. (Cited on page 124.)
- Lopez, V. and Motta, E. (2004). Ontology Driven question answering in AquaLog. In *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004), Manchester, England*. (Cited on page 119.)
- Lopez, V., Nikolov, A., Sabou, M., Uren, V., and Motta, E. (2010). Scaling up Question-Answering to Linked Data. In *Proceedings of Knowledge Engineering and Knowledge Management by the Masses (EKAW-2010), Lisboa, Portugal*. (Cited on page 124.)
- Lukovnikov, D., Hellmann, S., Gerber, D., and Unger, C. (2014). Mapping text to ontology with DBpedia Lemon and BOA. (Cited on pages viii and 127.)
- Manola, F. and Miller, E. (10. Februar 2004). RDF Primer. W3C Recommendation. (Cited on page 11.)
- Matsuo, Y. and Ishizuka, M. (2004). Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169. (Cited on page 22.)
- Meiser, T., Dylla, M., and Theobald, M. (2011). Interactive Reasoning in Uncertain RDF Knowledge Bases. In Berendt, B., de Vries, A., Fan, W., and Macdonald, C., editors, *CIKM'11*, pages 2557–2560. (Cited on page 78.)
- Mendes, P. N., Jakob, M., Garcia-Silva, A., and Bizer, C. (2011). DBpedia Spotlight: Shedding Light on the Web of Documents. In

- I-SEMANTICS*, ACM International Conference Proceeding Series, pages 1–8. ACM. (Cited on pages [23](#), [42](#), [59](#), [80](#), [85](#), and [86](#).)
- Miller, A. (1995). Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):39–41. (Cited on page [46](#).)
- Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. *ACL*, pages 1003–1011. (Cited on pages [22](#), [23](#), [24](#), [26](#), [38](#), and [39](#).)
- Morsey, M., Lehmann, J., Auer, S., and Ngonga Ngomo, A.-C. (2011). DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data. In *International Semantic Web Conference (1)*, pages 454–469. (Cited on page [60](#).)
- Morsey, M., Lehmann, J., Auer, S., Stadler, C., and Hellmann, S. (2012). DBpedia and the Live Extraction of Structured Data from Wikipedia. *Electronic Library and Information Systems*, page 27. (Cited on page [85](#).)
- Motik, B., Patel-Schneider, P. F., and Parsia, B. (2. Dezember 2008). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation. (Cited on page [12](#).)
- Nakamura, S., Konishi, S., Jatowt, A., Ohshima, H., Kondo, H., Tezuka, T., Oyama, S., and Tanaka, K. (2007). Trustworthiness Analysis of Web Search Results. In *Research and Advanced Technology for Digital Libraries*, volume 4675, pages 38–49. (Cited on pages [77](#) and [88](#).)
- Nakashole, N., Theobald, M., and Weikum, G. (2011). Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining, WSDM '11*, pages 227–236, New York, NY, USA. ACM. (Cited on pages [23](#), [27](#), [34](#), [38](#), [48](#), and [79](#).)
- Nakashole, N. and Weikum, G. (2012). Real-time population of knowledge bases: opportunities and challenges. In *Proceedings of AKBC-WEKEX*. (Cited on page [69](#).)
- Navigli, R. and Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, pages 217–250. (Cited on page [98](#).)
- Ngonga Ngomo, A.-C. (2012). On Link Discovery using a Hybrid Approach. *J. Data Semantics*, 1(4):203–217. (Cited on page [57](#).)
- Ngonga Ngomo, A.-C., Bühmann, L., Unger, C., Lehmann, J., and Gerber, D. (2013a). Sorry, I don't speak SPARQL — Translating

- SPARQL Queries into Natural Language. In *Proceedings of WWW*. (Cited on page [viii](#).)
- Ngonga Ngomo, A.-C., Bühmann, L., Unger, C., Lehmann, J., and Gerber, D. (2013b). SPARQL2NL - Verbalizing SPARQL queries. In *Proc. of WWW 2013 Demos*. (Cited on page [viii](#).)
- Ngonga Ngomo, A.-C., Heino, N., Lyko, K., Speck, R., and Kaltenböck, M. (2011a). Scms - semantifying content management systems. In *ISWC 2011*. (Cited on page [23](#).)
- Ngonga Ngomo, A.-C., Heino, N., Lyko, K., Speck, R., and Kaltenböck, M. (2011b). SCMS - Semantifying Content Management Systems. In *ISWC*. (Cited on page [69](#).)
- Ngonga Ngomo, A.-C. and Schumacher, F. (2009). BorderFlow: A Local Graph Clustering Algorithm for Natural Language Processing. In *CICLing*, pages 547–558. (Cited on page [60](#).)
- Nguyen, D. P. T., Matsuo, Y., and Ishizuka, M. (2007). Relation Extraction from Wikipedia using Subtree Mining. In *AAAI*, pages 1414–1420. (Cited on page [79](#).)
- Nguyen, T. and Kan, M.-Y. (2007). Keyphrase Extraction in Scientific Publications. pages 317–326. (Cited on page [23](#).)
- Pasternack, J. and Roth, D. (2011a). Generalized fact-finding. In *WWW '11*, pages 99–100. (Cited on page [77](#).)
- Pasternack, J. and Roth, D. (2011b). Making Better Informed Trust Decisions with Generalized Fact-Finding. In *IJCAI*, pages 2324–2329. (Cited on page [77](#).)
- Pasternack, J. and Roth, D. (2013). Latent Credibility Analysis. In *Proceedings of the 22Nd International Conference on World Wide Web, WWW '13*, pages 1009–1020. (Cited on page [77](#).)
- Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). WordNet: : Similarity - Measuring the Relatedness of Concepts. In *AAAI*. (Cited on page [60](#).)
- Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL query language for RDF. W3C Recommendation. (Cited on page [13](#).)
- Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *CONLL*, pages 147–155. (Cited on page [22](#).)
- Rizzo, G., Troncy, R., Hellmann, S., and Bruemmer, M. (2012). NERD meets NIF: Lifting NLP extraction results to the Linked Data Cloud. In *LDOW, 5th Workshop on Linked Data on the Web, April 16, 2012, Lyon, France*, Lyon, FRANCE. (Cited on page [62](#).)

- Röder, M., Usbeck, R., Hellmann, S., Gerber, D., and Both, A. (2014). N₃ - A Collection of Datasets for Named Entity Recognition and Disambiguation in the NLP Interchange Format. In *Submitted to The 9th edition of the Language Resources and Evaluation Conference, 26-31 May, Reykjavik, Iceland*. (Cited on pages [viii](#) and [54](#).)
- Ruiz-Casado, M., Alfonseca, E., and Castells, P. (2007). Automatising the learning of lexical patterns: An application to the enrichment of WordNet by extracting semantic relationships from Wikipedia. (Cited on page [69](#).)
- Rula, A., Palmonari, M., Harth, A., Stadtmüller, S., and Maurino, A. (2012). On the Diversity and Availability of Temporal Information in Linked Open Data. In *The 11th International Semantic Web Conference (ISWC2012)*. (Cited on page [105](#).)
- Rula, A., Palmonari, M., Ngonga Ngomo, A., Gerber, D., Lehmann, J., and Bühmann, L. (2014). Hybrid Acquisition of Temporal Scopes for RDF Data. In *In Proceedings of the 11th Extended Semantic Web Conference*. (Cited on page [viii](#).)
- Sarawagi, S. (2008). Information Extraction. *Found. Trends databases*. (Cited on page [68](#).)
- Sauermann, L. and Cyganiak, R. (2008). *Cool URIs for the Semantic Web*. W3C Interest Group Note, W3C. (Cited on page [11](#).)
- Seaborne, A. and Harris, S. (2013). SPARQL 1.1 Query Language. W3C recommendation, W3C. <http://www.w3.org/TR/2013/REC-sparql11-query-20130321/>. (Cited on page [14](#).)
- Seco, N., Veale, T., and Hayes, J. (2004). An Intrinsic Information Content Metric for Semantic Similarity in WordNet. *Proceedings of European Conference on Artificial Intelligence*, pages 1089–1090. (Cited on page [46](#).)
- Shadbolt, N., Berners-Lee, T., and Hall, W. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101. (Cited on page [3](#).)
- Shapiro, C. and Varian, H. R. (1998). *Information rules: a strategic guide to the network economy*. Harvard Business School Press, Boston, MA, USA. (Cited on page [7](#).)
- Shekarpour, S., Auer, S., Ngonga Ngomo, A.-C., Gerber, D., Hellmann, S., and Stadler, C. (2011). Keyword-driven SPARQL Query Generation Leveraging Background Knowledge. In *International Conference on Web Intelligence*. (Cited on pages [21](#) and [125](#).)
- Stadler, C., Lehmann, J., Höffner, K., and Auer, S. (2012). LinkedGeo-Data: A Core for a Web of Spatial Open Data. *Semantic Web Journal*, 3:333–354. (Cited on pages [4](#) and [19](#).)

- Stern, R. and Sagot, B. (2012). Population of a knowledge base for news metadata from unstructured text and web data. In *Proceedings of the AKBC-WEKEX*. (Cited on page 69.)
- Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In *WWW*, New York, NY, USA. ACM Press. (Cited on page 15.)
- Suchanek, F. M., Kasneci, G., and Weikum, G. (2008). YAGO: A Large Ontology from Wikipedia and WordNet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217. (Cited on page 15.)
- Suchanek, F. M., Sozio, M., and Weikum, G. (2009). SOFIE: A Self-Organizing Framework for Information Extraction. In *International World Wide Web conference*. (Cited on pages 23 and 26.)
- Talukdar, P. P., Wijaya, D., and Mitchell, T. (2012a). Acquiring Temporal Constraints between Relations. In *Proceedings of the Conference on Information and Knowledge Management (CIKM 2012)*, Hawaii, USA. Association for Computing Machinery. (Cited on page 79.)
- Talukdar, P. P., Wijaya, D., and Mitchell, T. (2012b). Coupled Temporal Scoping of Relational Facts. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM)*, Seattle, Washington, USA. Association for Computing Machinery. (Cited on page 79.)
- Theoharis, Y., Fundulaki, I., Karvounarakis, G., and Christophides, V. (2011). On Provenance of Queries on Semantic Web Data. *IEEE Internet Computing*, 15:31–39. (Cited on page 78.)
- Tran, T., Mathäß, T., and Haase, P. (2010). Usability of Keyword-Driven Schema-Agnostic Search. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., and Tudorache, T., editors, *ESWC (2)*, volume 6089 of *Lecture Notes in Computer Science*, pages 349–364. Springer. (Cited on page 125.)
- Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M. R., Weissenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D., Almirantis, Y., Pavlopoulos, J., Baskiotis, N., Gallinari, P., Artières, T., Ngonga, A., Heino, N., Gaussier, É., Barrio-Alvers, L., Schroeder, M., Androutsopoulos, I., and Paliouras, G. (2015). An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16:138. (Cited on page 5.)
- Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., and Decker, S. (2010). Sig.ma: Live views on the Web of Data. *Journal of Web Semantics*, 8(4):355–364. (Cited on page 125.)

- Tummarello, G., Delbru, R., and Oren, E. (2007). Sindice.com: Weaving the Open Linked Data. pages 552–565. (Cited on page 125.)
- Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., and Cimiano, P. (2012). Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. (Cited on pages vii, 21, 118, and 121.)
- Unger, C. and Cimiano, P. (2011). Pythia: Compositional meaning construction for ontology-based question answering on the Semantic Web. In *Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems (NLDB 2011)*. (Cited on pages 119 and 125.)
- Unger, C., Mccrae, J., Walter, S., Winter, S., and Cimiano, P. (2013). A lemon lexicon for DBpedia. In *Proceedings of 1st International Workshop on NLP and DBpedia, October 21-25, Sydney, Australia*, volume 1064 of *NLP & DBpedia 2013*, Sydney, Australia. CEUR Workshop Proceedings. (Cited on page 127.)
- Usbeck, R., Ngonga Ngomo, A.-C., Röder, M., Gerber, D., Athaide Coelho, S., Auer, S., and Both, A. (2014). AGDISTIS - Agnostic Disambiguation of Named Entities Using Linked Open Data. In *In Proceedings of the 13th International Semantic Web Conference*. (Cited on pages viii, 80, and 128.)
- V. Lopez, V. Uren, E. M. and Pasin, M. (2007). AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Journal of Web Semantics*, 5(2):72–105. (Cited on page 119.)
- V. Lopez, M. Sabou, V. U. and Motta, E. (2009). Cross-Ontology Question Answering on the Semantic Web – an initial evaluation. In *Proceedings of the Knowledge Capture Conference, 2009, California*. (Cited on page 119.)
- Wang, Y., Yang, B., Qu, L., Spaniol, M., and Weikum, G. (2011). Harvesting Facts from Textual Web Sources by Constrained Label Propagation. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM), Glasgow, Scotland, UK, October 24-28, 2011*, pages 837–846. (Cited on page 79.)
- Wang, Y., Zhu, M., Qu, L., Spaniol, M., and Weikum, G. (2010). Timely YAGO: harvesting, querying, and visualizing temporal knowledge from Wikipedia. In *EDBT*, volume 426 of *ACM International Conference Proceeding Series*, pages 697–700. ACM. (Cited on page 79.)
- Wu, F. and Weld, D. S. (2010). Open Information Extraction Using Wikipedia. In *ACL*, pages 118–127. The Association for Computer Linguistics. (Cited on page 26.)

- Wu, Z. and Palmer, M. S. (1994). Verb Semantics and Lexical Selection. In Pustejovsky, J., editor, *ACL*, pages 133–138. Morgan Kaufmann Publishers / ACL. (Cited on page 60.)
- Xu, F., Uszkoreit, H., and Li, H. (2007). A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. In *ACL*. (Cited on pages 22 and 25.)
- Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., and Ishizuka, M. (2009). Un-supervised relation extraction by mining Wikipedia texts using information from the web. In *ACL, ACL '09*, pages 1021–1029. (Cited on pages 22 and 79.)
- Yin, X., Han, J., and Yu, P. S. (2007). Truth discovery with multiple conflicting information providers on the web. In *KDD '07*, pages 1048–1052. (Cited on pages 77 and 78.)
- Yoichiro, H. (2006). Method for using Wikipedia as Japanese Corpus. *Doshisha studies in language and culture*, 9(2):373–403. (Cited on page 29.)
- Zaveri, A., Kontokostas, D., Sherif, M. A., Bühmann, L., Morsey, M., Auer, S., and Lehmann, J. (2013). User-driven quality evaluation of DBpedia. In Sabou, M., Blomqvist, E., Noia, T. D., Sack, H., and Pellegrini, T., editors, *I-SEMANTICS*, pages 97–104. ACM. (Cited on page 6.)
- Zhou, G. and Su, J. (2002). Named entity recognition using an HMM-based chunk tagger. In *ACL '02*, pages 473–480. (Cited on page 23.)

DECLARATION

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Universität Leipzig, Oktober 2015

Daniel Gerber