AUTOMATIC REFINEMENT OF LARGE-SCALE CROSS-DOMAIN KNOWLEDGE GRAPHS

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

André de Oliveira Melo
aus Santiago, Brasilien

Mannheim, 2018

Dekan:	Professor Dr. Heinz Jürgen Müller, Universität Mannheim
Referent:	Professor Dr. Heiko Paulheim, Universität Mannheim
Korreferent*:	Professor Dr. Heiner Stuckenschmidt, Universität Mannheim

Tag der mündlichen Prüfung: 24. August 2018

*Anmerkung: Der Korreferent wird aus dem lateinischen "con" und "referre" abgeleitet und damit ist der "zweite Gutachter" der Korreferent - mir zwei "r".

Abstract

Knowledge graphs are a way to represent complex structured and unstructured information integrated into an ontology, with which one can reason about the existing information to deduce new information or highlight inconsistencies. Knowledge graphs are divided into the terminology box (TBox), also known as ontology, and the assertions box (ABox). The former consists of a set of schema axioms defining classes and properties which describe the data domain. Whereas the ABox consists of a set of facts describing instances in terms of the TBox vocabulary.

In the recent years, there have been several initiatives for creating large-scale cross-domain knowledge graphs, both free and commercial, with DBpedia, YAGO, and Wikidata being amongst the most successful free datasets. Those graphs are often constructed with the extraction of information from semi-structured knowledge, such as Wikipedia, or unstructured text from the web using NLP methods. It is unlikely, in particular when heuristic methods are applied and unreliable sources are used, that the knowledge graph is fully correct or complete. There is a trade-off between completeness and correctness, which is addressed differently in each knowledge graph's construction approach.

There is a wide variety of applications for knowledge graphs, e.g. semantic search and discovery, question answering, recommender systems, expert systems and personal assistants. The quality of a knowledge graph is crucial for its applications. In order to further increase the quality of such large-scale knowledge graphs, various automatic refinement methods have been proposed. Those methods try to infer and add missing knowledge to the graph, or detect erroneous pieces of information. In this thesis, we investigate the problem of automatic knowledge graph refinement and propose methods that address the problem from two directions, automatic refinement of the TBox and of the ABox.

In Part I we address the ABox refinement problem. We propose a method for predicting missing type assertions using hierarchical multilabel classifiers and ingoing/outgoing links as features. We also present an approach to detection of relation assertion errors which exploits type and path patterns in the graph. Moreover, we propose an approach to correction of relation errors originating from confusions between entities. Also in the ABox refinement direction, we propose a knowledge graph model and process for synthesizing knowledge graphs for benchmarking ABox completion methods.

In Part II we address the TBox refinement problem. We propose methods for

inducing flexible relation constraints from the ABox, which are expressed using SHACL. We introduce an ILP refinement step which exploits correlations between numerical attributes and relations in order to the efficiently learn Horn rules with numerical attributes. Finally, we investigate the introduction of lexical information from textual corpora into the ILP algorithm in order to improve quality of induced class expressions.

In Appendix A we present an empirical comparison of local and global feature selection for transformation based multilabel classification. This work consolidates the advantage of using local feature selection, which is exploited in our approaches for type prediction and detection of relation assertion errors.

Zusammenfassung

Wissensbasen stellen eine Möglichkeit dar, komplexe strukturierte und unstrukturierte Informationen als Ontologie und Instanzen darzustellen. Mit diesen Informationen können Schlussfolgerungen, Regeln und andere Formen von Logik verwenden werden, um neue Informationen abzuleiten oder Inkonsistenzen aufzudecken. Wissensbasen sind aufgeteilt in einen terminologischen Formalismus (Ontologie/T-Box) und in einen assertionalen Formalismus (ABox). Ersteres besteht aus einer Reihe von Axiomen, die Klassen und Eigenschaften definieren und die Datendomäne beschreiben. Wohingegen die ABox aus einer Menge von Fakten besteht, die die Instanzen in Bezug auf das TBox-Vokabular beschreiben.

In den letzten Jahren gab es mehrere Initiativen, um große domänenübergreifende Wissensbasen (sowohl freie als auch kommerzielle) zu erstellen, wobei DBpedia, YAGO und Wikidata zu den erfolgreichsten freien Datansätzen gehören. Diese Graphen werden oft aus semistrukturiertem Wissen, wie Wikipedia, oder unstrukturierten Texten aus dem Internet unter Verwendung von NLP-Methoden konstruiert. Insbesondere wenn heuristische Methoden und unzuverlässige Datenquellen verwendet werden, ist es unwahrscheinlich, dass die Wissensbasis völlig korrekt oder vollständig ist. Es gibt einen Kompromiss zwischen Vollständigkeit und Korrektheit, der in jedem Konstruktionsansatz der Wissensbasis unterschiedlich behandelt wird.

Es gibt eine große Vielfalt von Anwendungen für Wissensbasen, z.B. semantische Suche, Frage-Antwort Szenarios, Empfehlungssysteme, Expertensysteme und persönliche Assistenten. Die Qualität einer Wissensbasis ist entscheidend für ihre Anwendungen. Um die Qualität solcher umfangreichen Wissensbasen weiter zu erhöhen, wurden verschiedene automatische Verfeinerungsverfahren vorgeschlagen. Diese Methoden versuchen, fehlendes Wissen abzuleiten und hinzuzufügen, oder fehlerhafte Informationen zu erkennen. In dieser Arbeit untersuchen wir das Problem der automatischen Verfeinerung von Wissensbasen und schlagen Methoden vor, die das Problem aus zwei Richtungen angehen: automatischen Verfeinerung von TBox sowie ABox.

In Abschnitt I befassen wir uns mit dem ABox-Verfeinerungsproblem. Wir schlagen eine Methode zur Vorhersage fehlender Typ-Informationen vor, die hierarchische Multilabel-Klassifikatoren und eingehende/ausgehende Verbindungen als Merkmale/Features verwenden. Wir präsentieren auch einen Ansatz zur Erkennung von falschen Beziehungen zwischen zwei Entitäten, der die Typinformation und Pfadmuster im Graphen verwendet. Darüber hinaus schlagen wir einen Ansatz zur Korrektur dieser Fehler vor, die aus Verwechslungen zwischen Entitäten entstehen. Im Rahmen der ABox-Verfeinerung, schlagen wir ein Modell und einen Prozess zur Synthese von Wissensbasen vor, die zum Benchmarking von Methoden zur ABox-Vervollständigung verwendet werden.

In Abschnitt II sprechen wir das TBox-Verfeinerungsproblem an. Wir schlagen Methoden vor, um aus der ABox flexible Restrikionen von Beziehungen zu erzeugen, die mit SHACL ausgedrückt werden. Wir führen einen ILP-Verfeinerungsschritt ein, der Korrelationen zwischen numerischen Attributen und Beziehungen ausnutzt, um Hornregeln mit numerischen Attributen effizienter zu lernen. Abschließend untersuchen wir die Einführung von lexikalischen Informationen aus Textkorpora, um die Qualität erzeugter Klassenausdrücke von ILP-Verfahren zu verbessern.

Im Anhang A präsentieren wir einen empirischen Vergleich der lokalen und globalen Merkmalselektion für transformationsbasierte Multilabel-Klassifikation. Diese Arbeit konsolidiert den Vorteil der Verwendung lokaler Merkmalselektion, die in unseren Ansätzen für die Typvorhersage und die Erkennung von falschen Beziehungen ausgenutzt wird.

Contents

1 Introduction 1

1.1 Research Questions 4

1.2 Contributions 5

1.3 Structure 5

2 Fundamentals 8

2.1 Knowledge Graphs in the Semantic Web 8

2.2 Linked Open Data 9

2.3 Knowledge Graph Construction 12

2.4 Incompleteness and Noise 13

2.5 Knowledge Graph Lifecycle 14

2.6 Ontology Learning 16

2.7 Automatic ABox Refinement Approaches 17

2.7.1 Error Detection vs. Graph Completion 17

2.7.2 Internal vs. External Methods 17

2.7.3 Latent vs. Graph Feature Models 17

2.8 Evaluation of Knowledge Graph Refinement Methods 18

3 Related Work 20

3.1 Synthesis of Knowledge Graphs for Benchmarking 20

3.2 Type Prediction 21

3.3 Relation Assertion Error Detection 23

3.4 Correction of Confusions in Knowledge Graphs 24

3.5 Relation Constraints Learning 25

3.6 Inductive Lexical Learning of Class Expressions 26

3.7 Learning Rules With Numerical Attributes 27

3.8 Summary 28

I ABox Refinement 30

4 Synthesizing Knowledge Graphs for Refinement Benchmarking 31

4.1 Introduction 31

4.2 Knowledge Graph Model 32

4.3 Synthesis Process 35
4.4 Experiments 37
4.5 Conclusion 42

5 Type Prediction using Hierarchical Multilabel Classification 43
5.1 Introduction 43
5.2 Preliminaries 44
5.2.1 Multilabel Classification Approaches 45
5.2.2 Hierarchical Multilabel Classification Approaches 45
5.2.3 Evaluation Measures 47
5.3 Problem Definition 48
5.4 Approach 50
5.4.1 Algorithm 50
5.4.2 Features 53
5.5 Experiments 56
5.5.1 Datasets 56
5.5.2 SLCN Base Classifier and Parameter Settings 59
5.5.3 Graph Features vs. Latent Features 59
5.5.4 Scalability Experiments 62
5.5.5 Large-Scale Experiments on SW Datasets 63
5.6 Conclusion 65

6 Detection of Relation Assertion Errors 66
6.1 Introduction 66
6.2 Problem Definition 67
6.3 Approach 67
6.3.1 Extracted Features 69
6.3.2 Learning the Model 71
6.4 Experiments 71
6.4.1 Datasets 72
6.4.2 Evaluation Measures 73
6.4.3 Parameter Settings 74
6.4.4 Comparison 75
6.4.5 Manual Evaluation 77
6.5 Conclusion 79

7 Correction of Confusions Between Entities 80
7.1 Introduction 80
7.2 Proposed Approach 81
7.2.1 Type Prediction 81
7.2.2 Retrieving Candidates 82
7.2.3 Correcting Wrong facts 83
7.3 Experiments 84
7.4 Conclusion 86

II TBox Refinement 87

8 Generation of SHACL Relation Constraints 88
8.1 Introduction . . . 88
8.2 Generating SHACL constraints . . . 89
8.2.1 SHACL . . . 90
8.2.2 Generation Process . . . 90
8.3 Experiments . . . 94
8.4 Conclusion . . . 98

9 Inductive Lexical Learning of Class Expressions 99
9.1 Introduction . . . 99
9.2 Preliminaries . . . 100
9.3 Approach . . . 102
9.4 Evaluation . . . 104
9.4.1 Experimental Setup . . . 104
9.4.2 Results . . . 105
9.4.3 Discussion . . . 107
9.5 Conclusion . . . 108

10 Learning Rules With Numerical Attributes 109
10.1 Introduction . . . 109
10.2 Problem Definition . . . 111
10.3 Interestingness Measure . . . 112
10.4 Correlation Lattice . . . 113
10.4.1 Independence Test . . . 114
10.4.2 Scalability . . . 115
10.5 ILP Algorithm . . . 116
10.6 Experiments . . . 116
10.7 Conclusion . . . 120

11 Thesis Conclusion 121
11.1 Part I: ABox Refinement . . . 121
11.2 Part II: TBox Refinement . . . 122
11.3 Open Issues and Limitations . . . 123
11.4 Future Work . . . 124
11.4.1 General Perspective . . . 125

A Local vs. Global Feature Selection in Multilabel Classification 127
A.1 Introduction . . . 127
A.2 Background . . . 128
A.2.1 Multilabel Classification . . . 129
A.2.2 Hierarchical Multilabel Classification . . . 130
A.2.3 Evaluation Measures . . . 132

A.2.4 Feature Selection Methods 134
A.3 Related Work 134
A.4 Feature Selection on Transformed Multilabel Classification 135
A.4.1 Global Feature Selection 135
A.4.2 Local Feature Selection 136
A.4.3 Analysis of Local Feature Sets 137
A.5 Experiments 138
A.5.1 Datasets 139
A.5.2 Scalability 142
A.5.3 Results 143
A.5.4 Statistical Analysis 147
A.6 Conclusion and Future Work 149

Bibliography **151**

List of Publications

Parts of the work presented in this thesis have been previously published in international journals and proceedings of international conferences. For all publications the author of this thesis was a key contributor of the work presented in both the publications and this thesis.

- André Melo, Martin Theobald and Johanna Völker: *Correlation-Based Refinement of Rules with Numerical Attributes*. In Proceedings of FLAIRS'14, pages 345–250 (125): Chapter 10
- André Melo and Heiko Paulheim. *Detection of Relation Assertion Errors in Knowledge Graphs*. In Proceedings of K-CAP'17, pages 22:1–22:8 (121): Chapter 6.
- André Melo and Heiko Paulheim: *Type Prediction in RDF Knowledge Bases Using Hierarchical Multilabel Classification*. In Proceedings of WIMS'16, pages 14:1–14:10 (124): Chapter 5
- André Melo and Heiko Paulheim: *Synthesizing Knowledge Graphs for Link and Type Prediction Benchmarking*. In Proceedings of ESWC'17, pages 136–151 (123): Chapter 4
- André Melo and Heiko Paulheim: *Type Prediction in RDF Knowledge Bases Using Hierarchical Multilabel Classification With Graph and Latent Features*. International Journal on Artificial Intelligence Tools, Volume 26, 2017 (126): Chapter 5
- André Melo and Heiko Paulheim: *An Approach to Correction of Erroneous Links in Knowledge Graphs*. In Proceedings of QEKGraph colocated with K-CAP'17 (120): Chapter 7
- André Melo and Heiko Paulheim: *Learning SHACL Constraints for Validation of Relation Assertions in Knowledge Graphs*. Submitted to Semantic Web Journal (Tracking number: 1888-3101): Chapter 8
- Lorenz Bühmann, Daniel Fleischhacker, Jens Lehmann, André Melo and Johanna Völker: *Inductive Lexical Learning of Class Expressions*. In Proceedings of EKAW'14, pages 42–53 (23): Chapter 9

- André Melo and Heiko Paulheim: *Local and Global Feature Selection for Multilabel Classification With Binary Relevance*. Airtificial Intelligence Review, pages 1–28, 2017 (122): Appendix A

List of Figures

1.1 Example of movie recommendation using a knowledge base (28) . 2

2.1 Semantic Web stack 10
2.2 The Linking Open Data cloud diagram 11
2.3 The Linked Data lifecycle 15

4.1 Distances of performance measures to original datasets 39
4.2 Effect of scaling the replica sizes up and down 39
4.3 Nemenyi Critical distance diagrams for link and type prediction . 41
4.4 Synthesis process runtime over dataset size for the ESWC2015 model 42

5.1 A subset of the DBpedia type hierarchy 45
5.2 Hierarchical multilabel classification local classifier approaches . 47
5.3 Evaluation of the impact of the parameters n and k on hF and runtime. 60
5.4 Type prediction results for different number of HolE embeddings dimensions 61
5.5 Scalability in terms of number of instances, features and labels . . 63

6.1 Critical distance diagram comparing path selection heuristics . . . 75
6.2 Manual evaluation on DBpedia and NELL 78
6.3 Runtime comparison of the evaluated methods 79

7.1 Manual evaluation on DBpedia and NELL respectively 85

8.1 Example of decision tree pruning 91
8.2 Manual evaluation on DBpedia and YAGO 96
8.3 Manual evaluation of the differences between SHACL and SSI on DBpedia and YAGO 96

9.1 Outline of the general learning approach in ELTL: Class expressions taking the available background knowledge into account are generated and evaluated in a heuristic with respect to the target learning problem. Figure adapted from (77). 101
9.2 Illustration of a search tree in ELTL. 102

10.1 Confidence distribution over age Y 110
10.2 Confidence distribution of rule r_2 over the *age* attribute for the overall USA population (a) as well as refined by the states Florida (b) and South Dakota (c) 111
10.3 Example of a body support distribution (10.3a) with different positives distributions (10.3b) and (10.3d) 112
10.4 Example of correlation lattice for the numerical attribute *age* . . . 114
10.5 Example of a test for independence between the literals *maritalStatus(X,single)* and *employmentStatus(X,unemployed)* 115
10.6 Evaluation of the correlation lattice construction 118
10.7 Evaluation of the ILP extension 119

A.2 Wilcoxon test on flat and hierarchical multilabel datasets 148
A.1 Global vs local feature selection comparison with J48 and measures D_1 and D_2. 150

List of Tables

2.1 Profiling of major LOD datasets (166) 13

4.1 Statistics about the datasets used in the experiments 38
4.2 Summary of the link prediction results 40
4.3 Summary of the type prediction results 40

5.1 Statistics about the datasets used 57
5.2 Comparison of different local classifiers on SLCN 59
5.3 Comparison of hF for type prediction on different feature sets . . 61
5.4 Evaluation of different classification methods on large cross-domain SW datasets 64
5.5 Evaluation of different classification methods on smaller SW datasets 65

6.1 Comparison of local classifiers and number of selected features on generated errors of kind 1 74
6.2 Comparison of FMRR on generated errors of kind 1 75
6.3 Comparison of FMRR on generated errors of kind 2 76
6.4 Proportion of path and type features selected 77

8.1 PaTyBRED features translation into SHACL 93

9.1 Excerpt of the 50 class expressions that have been evaluated for the class `Astronaut`. The first column denotes the rank of the DL-Learner output without taking statistical measures into account. Only the class expressions have been shown in random order to the evaluators. 105
9.2 Results of relevance measure analysis. 106

10.1 Example of rules without numerical intervals 109
10.2 Example of rules refined with numerical intervals 110

A.1 Enron features ranked by information gain in descending order . . 137
A.2 Statistics about the flat datasets used 140
A.3 Statistics about the hierarchical datasets used 141

A.4 Comparison of local and global feature selection with *mean* aggregation on flat multilabel datasets 144
A.5 Comparison of local and global feature selection with *mean* aggregation on hierarchical multilabel datasets 145
A.6 Comparison of local and global feature selection with *max* aggregation on flat multilabel datasets 146
A.7 Comparison of local and global feature selection with *max* aggregation on. popular hierarchical multilabel datasets 147
A.8 Correlations between D_1 and D_2 measures and the ratio between global and local feature selection approaches for different evaluation measures 148

List of Acronyms

AI	Artificial Intelligence
IE	Information Extraction
IR	Information Retrieval
IRI	Internationalized Resource Identifier
KB	Knowledge Base
KG	Knowledge Graph
DAG	Directed Acyclic Graph
NLP	Natural Language Processing
SRL	Statistical Relational Learning
OIE	Open Information Extraction
POS	Part-of-speech
RDF	Resource Description Framework
RDFS	RDF Schema
OWL	Web Ontology Language
SHACL	Shapes Constraint Language
LOD	Linked Open Data
SVM	Support Vector Machine
HMC	Hierarchical Multilabel Classification
BR	Binary Relevance
idf	inverse document frequency
tf-idf	term frequency–inverse document frequency

Chapter 1

Introduction

Semantic Web knowledge graphs are an important information source for many intelligent systems that require access to structured knowledge. Those knowledge graphs contain descriptions of the data as well as factual knowledge about real world entities and their relations and attributes in a fully machine-readable format. Some of the most used Semantic Web knowledge bases are large-scale cross-domain datasets, such as DBpedia (106), YAGO (186), NELL (26), and Wikidata (56).

Generating such datasets is an extremely challenging task, which often involves a trade-off between coverage and accuracy. That means it is difficult to generate large datasets with high quality. Guaranteeing that every fact is correct is complicated and often requires a manual evaluation, which is unfeasible on the scale of the aforementioned datasets with tens or hundreds of million facts. YAGO has been estimated to have 95% of correct data (186), while DBpedia's correctness is estimated to be 88% (220) and NELL's 74% (26).

On the other hand, while these datasets are some of the largest existing knowledge graphs, it is practically impossible to achieve full completeness in such cross-domain datasets. That means incompleteness cannot be solved. However, it is important to reduce it as much as possible without sacrificing the data's correctness.

Both incompleteness and noise are common problems and solving, or at least reducing, them would be of great benefit for the applications which use such datasets. The process of reducing incompleteness and noise is called *refinement*. Developing automatic knowledge graph refinement methods which can do that on a large scale is a good way to reduce these problems therewith improving the quality of the data.

There is a wide variety of applications for knowledge graphs, e.g. semantic search and discovery, question answering, recommender systems, expert systems and personal assistants. For instance, knowledge graphs can play an important role addressing the cold-start problem, which is common in recommender systems and concerns the issue that the system cannot draw any inferences for users or items

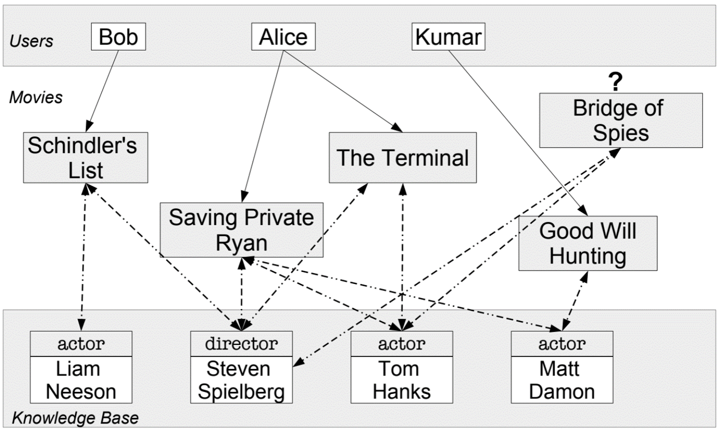


Figure 1.1: Example of movie recommendation using a knowledge base (28)

about which it has not yet gathered sufficient information.

Knowledge graphs are a great tool for leveraging external knowledge for improving content-based recommendations. Figure 1.1 illustrates how knowledge graphs can be exploited by recommender systems to reduce the cold-start problem (28). With the information from knowledge graphs it is possible to extend the movies with information about actors and directors, including how they are related. In the example, Bob who has only watched Schindler's List so far, can be recommended other popular movies directed by the Steven Spielberg, such as Saving Private Ryan and The Terminal.

If a knowledge graph has wrong or missing links, that can lead to worse recommendations. By improving the quality and coverage of a knowledge graph, i.e. removing wrong facts and adding missing facts, this and many other applications which use knowledge graphs can indirectly profit from such improvement.

There is a wide variety of refinement methods in the literature. Some approaches focus on refining the TBox, also known as schema or ontology, which describes a conceptualization, a set of concepts and properties for these concepts. Others approaches focus on refining the ABox, which consists of TBox-compliant statements about individuals belonging to those concepts. While there exist methods which can use external resources, such as web data and information extracted from textual corpora, the focus of this thesis in on the methods which rely exclusively on information already present in the knowledge graph.

The refinement of TBox is also known in the literature as ontology learning. Since ABox statements need to comply with the TBox, the refinement of the TBox can lead to the indirect detection of ABox errors.

Designing a good TBox can be a difficult task, especially on large-scale cross-domain datasets, where it is complicated to have a deep understanding of the data. Moreover, in crowdsource datasets, such as Wikidata, it is important that users – who help populate the ABox – can clearly understand the TBox vocabulary and use it adequately.

In many instances it can be interesting to consider a bottom-up approach, where the ABox information is used in order to learn patterns which can be integrated into the TBox. With that, unforeseen patterns can be induced from the data, and wrong patterns or often misused vocabulary can be identified and fixed.

Many ABox refinement methods do not use TBox information, relying exclusively on the relations between instances. However, since many knowledge graphs do have a schema, which provide valuable information about the data domain, it makes sense to exploit such information when available. The ABox refinement problem in the Semantic Web community has been traditionally divided into type prediction, i.e. prediction of missing instance types (concepts), and link prediction, i.e. prediction of relationships between instances.

Reasoning is one of the main premises of the semantic web, and it enables the deduction of new facts, thereby reducing incompleteness. Nevertheless, performing deductive reasoning on incorrect data can be problematic, since chains of several wrong facts can be deduced from a single wrong fact, resulting in the increase of incorrectness. It has been shown that classical ontology reasoning can propagate errors and lead to nonsensical results when applied on real-world datasets (150). Therefore, it is necessary to develop more robust statistical methods which can work on noisy data in order to mitigate this problem.

Some of those aforementioned challenges from automatic knowledge graph refinement are addressed in this thesis. In Part I we present ABox refinements methods, which include a type prediction approach based on hierarchical multilabel classification, an approach to detection of relation assertion errors and an approach to correction of wrong links originated from confusions between instances. In addition, we also propose a knowledge graph model and synthesis process for generating artificial knowledge graphs for benchmarking of ABox completion methods.

In Part II we present TBox refinement methods, which include the generation of relation constraints from the ABox. We also propose a method for induction of intuitive class expressions by incorporating lexical information into inductive logic programming (ILP), and an extension of the ILP algorithm to efficiently learn rules with numerical attributes, where correlations between numerical properties and literals are exploited by building a so-called correlation lattice.

In the Appendix we present a work about local feature selection in multilabel classification, which is one of the foundations of our type prediction and relation assertion error detection methods. There we perform an extensive empirical comparison of local and global feature selection for transformation based multilabel classification methods, and show that the local approach consistently outperforms the global approach in terms of predictive performance without sacrificing runtime.

In this thesis we study the automatic knowledge graph refinement problem from different perspectives and evaluate our proposed methods on a large set of popular publicly available knowledge graphs. It is important to notice that, as mentioned earlier, external information is also worth considering and can potentially further improve the quality of a knowledge graph. However, we restrict ourselves to the information already contained in the knowledge graph, such as the ontology axioms, type assertions, path patterns, textual and numerical information from data properties.

1.1 Research Questions

The general research question of this thesis is *How to automatically refine knowledge graphs using the information already contained in the graph?*. In order to answer this question, we divide it into a number of more specific questions which are addressed in the later chapters of this thesis.

RQ1: *How to automatically improve the quality of the ABox?*
The problem of automatic refinement of knowledge graphs can be divided into two subproblems. One is the refinement of the ontology (TBox), addressed by RQ2, and the other is the refinement of the assertions box (ABox). The ABox refinement problem includes the detection and correction of erroneous facts, addressed in Chapters 6 and 7, as well as the prediction of missing facts, addressed in Chapter 5.

RQ2: *How to automatically improve the quality of the TBox?*
Automatic improvement of the TBox is an important part of the knowledge graph refinement problem. One of the main approaches to ontology learning is the induction of TBox axioms from the ABox. This can aid the construction and maintenance of the ontology and provide valuable insights about the data. This question is addressed in Chapters 8 and 9.

RQ3: *How to efficiently apply refinement methods on large-scale data?*
Enabling automatic knowledge graph refinement methods to be applied on large-scale graphs, as well as across multiple interlinked datasets is also an important problem. Scalable methods and efficient heuristics to explore large search spaces are good means to enable such methods to be applied on large amounts of data. This question is addressed in Chapter 10 and Appendix A.

RQ4: *How to synthesize knowledge graphs for benchmarking knowledge graph completion methods?*
Artificially generated datasets are an important tool for comparing different algorithms for a given task. Being able to control characteristics of the data and analyzing how methods are affected by them are important to deeply understand the behavior of these methods. The challenge is to synthesize knowledge graphs from a given model, which should be able to convey characteristics important to the knowledge graph completion task. This question is addressed in Chapter 4.

1.2 Contributions

The contributions of this thesis are diverse and cover several different aspects of the automatic knowledge graph refinement, including methods for automatic TBox and ABox refinement. More specifically the contributions of this thesis are the following:

- A knowledge graph synthesis model for benchmarking of link and type prediction methods – Chapter 4.
- An approach to type prediction using hierarchical multilabel classification – Chapter 5.
- A machine learning approach to detection of relation assertion errors using path and type features – Chapter 6.
- An approach to correction of erroneous links originated from confusion between entities – Chapter 7.
- An approach to learning expressive relation constraints based on our proposed relation assertion error detection method – Chapter 8
- An investigation of how to incorporate lexical information in order to learn more intuitive class expressions – Chapter 9.
- A correlation-based approach for refinement of clauses with numerical attributes – Chapter 10.
- An empirical study comparing local with global feature selection multilabel classification transformation methods – Appendix A.

1.3 Structure

The contents of the next chapters of this thesis are summarized as follows:
Chapter 2: Fundamentals - A brief motivation for the thesis and an overview of some of the basics of knowledge graphs, semantic web, linked open data, knowledge graph construction, knowledge graph refinement methods and evaluation approaches is presented.
Chapter 3: Related Work - A comprehensive overview of existing knowledge graph refinement approaches is given. The overview covers works related to all of the thesis contributions listed earlier.
Chapter 4: Synthesizing Knowledge Graphs for Refinement Benchmarking - A model for synthesizing knowledge graphs for benchmarking of knowledge graph completion methods, more specifically type and link prediction, is proposed. The model is able to replicate different characteristics of a knowledge graph related to the completion task. An evaluation is conducted to find out how well the proposed

model is able to replicate knowledge completion results of popular datasets and to find out which model elements are the most relevant for a good replication.

Chapter 5: Type Prediction using Hierarchical Multilabel Classification - Hierarchical multilabel classification with local feature selection is applied for the prediction of missing types in knowledge graphs. It is shown that the structure of type hierarchies can be exploited to efficiently handle the cross-domain nature of large knowledge graphs and to improve scalability. Our evaluation show that the proposed approach outperforms previous state-of-the-art type prediction approaches.

Chapter 6: Detection of Relation Assertion Errors - A supervised method for detection of relation assertion errors based on path and type features is proposed. The method consists of a binary classifier for each relation, which predicts whether a pair of subject-object entities belongs to the given relation. Moreover, an approach for the exploration of the graph paths search space involving heuristic path relevance measures is presented. It is shown that the proposed method is able to outperform state-of-the-art error detection and link prediction approaches on the error detection task.

Chapter 7: Correction of Confusions Between Entities - An approach to correction of confusions between entities is presented, where the focus is placed on detected relation assertion errors caused by such confusions. The proposed approach is able to detect such cases and propose corrections, based on Wikipedia disambiguation links and approximate string matching. It is shown that the approach is capable of correcting many of these cases, although the current accuracy achieved is not good enough to enable its application in a fully automated fashion.

Chapter 8: Generation of SHACL Relation Constraints - An approach to generation of relation constraints, based on the error detection method from Chapter 6, is proposed. The approach consists of learning a decision tree to detect error in each relation, which is then translated into a logical expression that represents the constraint. It is shown that the learned constraints can be more flexible and adapt better to incomplete data, being better at detecting errors than other state-of-the-art ontology learning approaches, such as statistical schema induction.

Chapter 9: Inductive Lexical Learning of Class Expressions - A method for generation of more intuitive class expressions based on lexical information is presented. It extends the existing DL-Learner heuristics, which rely on scores based on examples coverage obtained via logical inference, by adding relatedness information about the target class and the entities in its class expression obtained from textual corpora. A manual evaluation shows that a combination of relatedness measures is able to resemble the human perception of intuitiveness of a class expression best.

Chapter 10: Learning Rules With Numerical Attributes - A method for efficient refinement of rules with numerical attributes based on correlations between numerical attributes and categories is presented. The method exploits distributions of instances over numerical attributes on different subpopulations, and organize the information on a so-called correlation lattice. The lattice can be later used in the ILP refinement step to efficiently explore the search space thereby improving scalability.

Appendix A: Local vs. Global Feature Selection in Multilabel Classification - An empirical comparison between local and global feature selection strategies for transformation based multilabel classification methods. It is shown that local feature selection, where the selection is performed independently on each transformed datasets and local classifiers can work on specialized set of features, consistently outperforms the global approach.

Chapter 2

Fundamentals

In this chapter we present works which are fundamental to understanding of the thesis. It starts with the basics of knowledge graphs, semantic web and linked open data, then an overview of main knowldege graph ABox and TBox refinement methods are presented followed by evaluation approaches.

2.1 Knowledge Graphs in the Semantic Web

A knowledge graph[1], or knowledge base, is a technology used to store complex structured and unstructured information used by a computer system. It is composed by ABox and TBox statements. TBox statements describe a conceptualization, a set of concepts and properties for these concepts. ABox are TBox-compliant statements about individuals belonging to those concepts. For instance, “every musician is a person” and “musicians play for bands” are TBox statements, while “Trent Reznor is a musician”, “Nine Inch Nails is a band” and “Trent Reznor plays for Nine Inch Nails” are ABox statements.

In the Semantic Web the Resource Description Framework (RDF)[2] is the language of choice for describing knowledge graphs. It is represents statements about resources in the form of triples <subject–predicate–object>, where the resources are represented by a uniform resource identifier (URI). The subject denotes the resource, and the predicate expresses a relationship between the subject and the object or properties of the subject. Moreover, resources may be divided into groups called classes, whose members of a class are known as instances of the given class.

RDF standard representation uses standard XML notation, however, there are also more readable representations such as N-Triples, Turtle and Notation3. The ABox example mentioned earlier can be represented in RDF with Turtle notation as follows:

[1] The term Knowledge Graph was coined by Google in 2012 and has been recently also used to refer to Semantic Web knowledge bases.

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

```
@prefix ex: <http://example.org/> .

ex:Nine_Inch_Nails a ex:Band .
ex:Trent_Reznor a ex:Musician ;
     ex:playsFor ex:Nine_Inch_Nails .
```

RDF is also used as base for ontology languages, such as OWL[3] and RDFS[4], which are used to define the TBox, often also referred to as ontology or schema. Both the Web Ontology Language (OWL) and RDF Schema (RDFS) provide a data-modelling vocabulary for RDF. OWL is intended to be compatible with and extend RDFS.

With RDFS one can, for example, define that the domain of the ex:playsFor relation is ex:Musician and the range is ex:Band. It can also organize classes in a hierarchy determining subsumption relations between them. For instance, we can define a more general class ex:Agent which is the superclass of both ex:Musician and ex:Band, and a more general relation ex:memberOf which subsumes ex:playsFor and has domain and range ex:Agent.

Knowledge graphs represented in RDF can be accessed using the query language SPARQL[5]. The language has capabilities for querying across diverse RDF data sources. It is also capable of querying required and optional graph patterns, which can be combined with conjunctions and disjunctions. SPARQL also supports constraining queries by source RDF graph as well as many of the features supported by SQL.

```
@prefix ex: <http://example.org/> .

ex:Musician rdfs:subClassOf ex:Agent .
ex:Band rdfs:subClassOf ex:Agent .
ex:playsFor rdfs:domain ex:Musician ;
            rdfs:range  ex:Band .
ex:memberOf rdfs:domain ex:Agent ;
            rdfs:range  ex:Agent .
```

Figure 2.1 illustrates the architecture of the Semantic Web, showing how the different layers are stacked. Top of the stack (Trust, Proof and Unifying logic layers) is an essential part of the vision of the Semantic Web. However, at the moment there are still no implementation standards for those layers.

2.2 Linked Open Data

Linked Open Data is a method for publishing structured data so that it can be interlinked and become more useful through semantic queries. It builds upon standard

[3] https://www.w3.org/OWL/

[4] https://www.w3.org/TR/rdf-schema/

[5] https://www.w3.org/TR/rdf-sparql-query/

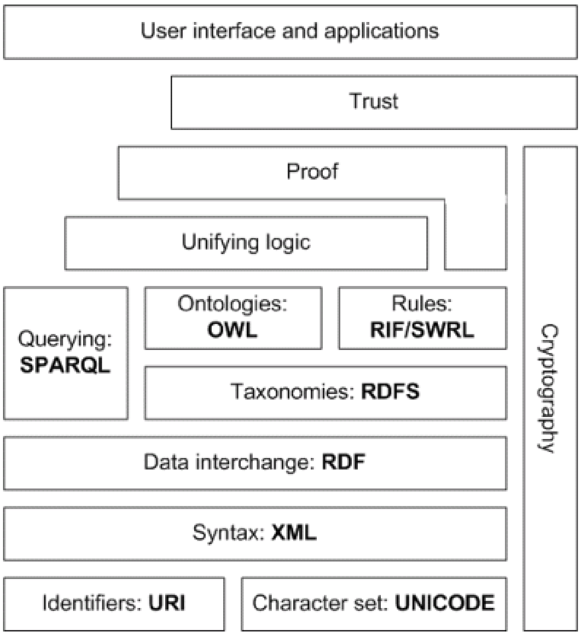


Figure 2.1: Semantic Web stack

web technologies, extending them to share information in a way that can be read automatically by computers. The four principles of linked data outlined by Tim Berners-Lee are the following:

1. Use URIs to name (identify) things.

2. Use HTTP URIs so that these things can be looked up (interpreted, "dereferenced").

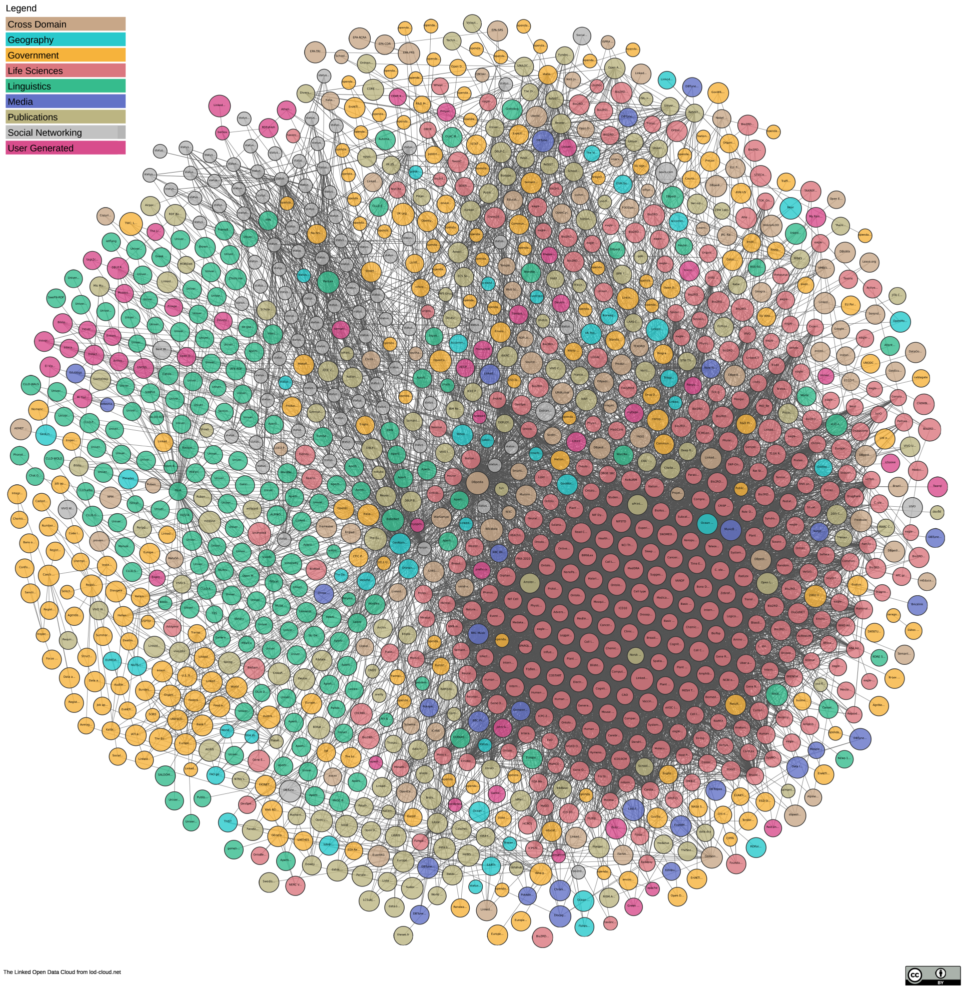
3. Provide useful information about what a name identifies when it's looked up, using open standards such as RDF, SPARQL, etc.

4. Refer to other things using their HTTP URI-based names when publishing data on the Web.

The number of linked open datasets available has quickly grown in the last few years, going from a few dozen of datasets in 2007 to more than a thousand nowadays. Figure 2.2[6] shows the Linked Open Data cloud diagram (2018-05-30) featuring 1168 datasets.

This gives an idea of the rapid adoption of the LOD principles. However, it is also worth noting that there are still several challenges to be addressed. What Figure 2.2 does not show is the fact that many of those datasets do not adequately

[6] "Linking Open Data cloud diagram 2018, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. `http://lod-cloud.net/`"



The Linked Open Data Cloud from lod-cloud.net

Figure 2.2: The Linking Open Data cloud diagram

apply the linked data best practices (76). These best practices can be grouped into three areas:

- *Linking*: By setting RDF links, data providers connect their datasets into a single global data graph which can be navigated by applications and enables the discovery of additional data by following links.
- *Vocabulary Usage*: Publishers should use terms from widely-used vocabularies in order to ease the interpretation of their data. If data providers use their own vocabularies, the terms of such proprietary vocabularies should be dereferencable, and their definitions should contain RDF links pointing to widely-used vocabularies.
- *Metadata Provision*: Linked Data should be as self-descriptive as possible, and therefore include metadata. For instance, information about licensing, provenance and accessibility should be provided. With that, users and applications can easily know whether they are licensed to use the data, assess its quality and access the data via alternative methods such as SPARQL endpoints or data dumps.

An evaluation of the adoption of linking, vocabulary, metadata best practices (176), shows that most datasets fail to fully apply these practices. Provenance information is provided for roughly a third of all datasets. Only 10% of all datasets provide machine-readable licensing information and only 14.69% provide VoID[7] metadata. This results there is still a long way for the vision of linked data

2.3 Knowledge Graph Construction

Most of the popular large-scale knowledge graphs, because of their magnitude, cannot be fully manually created and maintained. They rely, at least partially, on automatic methods for knowledge extraction which transform unstructured or semi-structured data, such as natural language text and web tables, into RDF triples. In this section we provide a brief overview of how some of the most popular publicly available large-scale cross-domain knowledge graphs are constructed.

DBpedia: DBpedia is extracted from Wikipedia's structured data. The infoboxes of articles are the main source of information, with types of infoboxes mapped into classes, and keys into properties in the DBpedia ontology.

YAGO - Yet Another Great Ontology: YAGO's extraction is also based on Wikipedia. However, its classes are derived from the Wikipedia's category system instead of infobox type, and infobox keys are manually mapped into a smaller set of properties. YAGO extracts and fuses knowledge extracted from various Wikipedia language definitions, while DBpedia creates different KGs for each language.

NELL - Never Ending Learning Language: Instead of just using Wikipedia, NELL extracts information from a large scale corpus of diverse websites. It is a

[7] `http://www.w3.org/TR/void/`

Dataset Version	DBpedia 2016-04	YAGO YAGO3	Wikidata 2016-08-01	OpenCyc 2016-09-05	NELL 08m.995
#instances	5 109 890	5 130 031	17 581 152	118 125	1 974 297
#facts	397 831 457	1 435 808 056	1 633 309 138	2 413 894	3 402 971
#classes	754	576 331	30 765	116 822	290
#relations	3555	93 659	11 053	165	1334
avg indegree	13.52	17.44	9.83	10.03	5.33
avg outdegree	47.55	101.86	41.25	9.23	1.25

Table 2.1: Profiling of major LOD datasets (166)

research project that attempts to create a computer system that learns over time to read the web. Since it uses unstructured text as source, it relies on more complex information extraction approaches which learn text patterns that correspond to type and relation assertions.

Wikidata: Wikidata is a collaboratively edited knowledge graph, operated by the Wikimedia foundation that also hosts Wikipedia. Data is entered and maintained by crowdsourced editors as well as automated bots. After the shutdown of Freebase, the data contained in Freebase was subsequently moved to Wikidata. A particularity of Wikidata is that for each axiom, provenance metadata can be included

OpenCyc: OpenCyc is a publicly available and reduced version of Cyc, which is one of the oldest knowledge graphs developed and curated by CyCorp.

There also exist other important private knowledge graphs such as Google's Knowledge Vault, Yahoo's Knowlege Graph, Facebook's Entities Graph and Microsoft's Satori. However, since these datasets are not publicly available and therefore cannot be used in our experiments, we do not get into more details.

Table 2.1 (166) shows some statistics about the aforementioned datasets, including the number of instances, facts, classes, relations, and average in-degree and out-degree. The latter two values indicate the average number of ingoing links per instance, i.e. relation assertions where the instance is the object, and the average number of outgoing links, i.e. relation assertions where the instance is the subject.

Given the quality of the sources, scale of the datasets, and complexity of the extraction process it is common that these datasets have several missing facts as well as erroneous data. In the next sections we discuss these problems in more details.

2.4 Incompleteness and Noise

All of the knowledge graphs mentioned in Section 2.3 have two problems in common: incompleteness and noise. Incompleteness refers to the absence of true facts in the data, while noise refers to the existence of false facts.

There are two main possible sources of noise: extraction errors and source data errors. The former refers to erroneous facts generated from correct input which was corrupted by problems in the information extraction process. The latter refers to errors which were already existent in the source data, e.g., a wrong entry in Wikipedia infobox, or a typo in textual data.

It has been estimated that Wikipedia, which is the source from DBpedia and YAGO, has 2.8% of wrong statements (209). YAGO's and DBpedia's correctness have been estimated to be 95% and 88% respectively (186; 220), while NELL's is estimated to be only 74% correct (26). These figures have been calculated based on manual evaluation of data samples.

Estimating incompleteness is a much more complicated task. The high level of incompleteness of most graphs can be illustrated with the fact that in Wikidata only 2% of all people have a father, in YAGO the average number of children per person is 0.02 (65), and in Freebase 71% of people have no known place of birth and 75% no nationality (51). In general, between 69% and 99% of instances in popular KGs lack at least one property that other entities in the same class have (185).

Refining knowledge graphs, i.e. reducing both incompleteness and noise, is an extremely relevant task in such datasets. Given the large size of the cross-domain datasets, which often contain several millions of triples, automatic methods are essential to enable the refinement of datasets on such scale. In the next sections we give an overview of automatic knowledge graph refinement methods, starting with ontology learning, i.e. the refinement of the TBox, and then with automatic ABox refinement approaches.

2.5 Knowledge Graph Lifecycle

Once a knowledge graph is created and deployed as linked data, it needs to be adequately maintained. Additionally, the issues previously discussed should be addressed in order to progressively improve the quality of the knowledge graph. The lifecycle of linked data on the web (7), which comprises of eight stages shown in Figure 2.3, provides a framework for maintaining the data and continuously improving it.

The eight stages of the linked data lifecycle are defined as follows:

- *Extraction*: Information represented in unstructured form or adhering to other structured or semi-structured representations must be extracted and converted into RDF.

- *Storage/Querying*: Mechanisms have to be in place to store, index and query this RDF data efficiently.

- *Authoring*: Users must have the opportunity to create new structured information or to correct and extend existing ones.

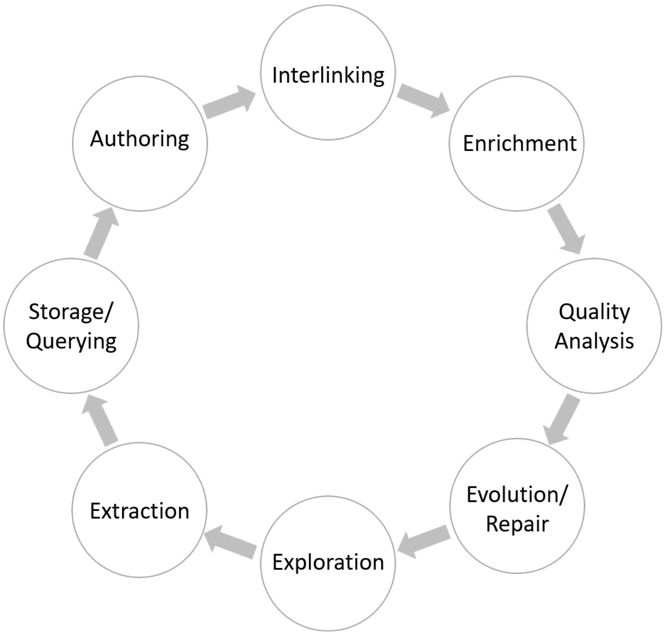


Figure 2.3: The Linked Data lifecycle

- *Interlinking*: Information about the or related entities in different datasets needs to be linked.
- *Enrichment*: Since Linked Data primarily comprises instance data, we observe a lack of classification, structure and schema information. This deficiency can be tackled by approaches for enriching data with higher-level structures in order to be able to aggregate and query the data more efficiently.
- *Quality Analysis*: Datasets contain a variety of information of different quality. The quality of the information provided by different sources should be adequately assessed.
- *Evolution/Repair*: Knowledge and information are dynamic in various domains. Changes and modifications to knowledge graphs should be considered and methods should be able to spot problems and to automatically suggest repairs.
- *Exploration*: Applications and users have to be empowered to browse, search and explore the information available in a fast, reliable and user friendly manner.

This thesis addresses mainly the enrichment and evolution stages. The ABox refinement methods proposed in the thesis can be used to detecting and correcting errors as well as predict missing facts. The TBox refinement methods, on the other hand, can induce TBox axioms from the ABox, thereby spotting problems and indirectly improving the quality of the ABox. The combination of the two kinds of approaches enables the TBox and ABox to evolve together in a virtuous cycle.

2.6 Ontology Learning

Automatic ontology learning methods can be extremely helpful on the process of creation, maintenance and extension of ontologies. While the quality of the learned ontologies might not be perfect, especially when learning on noisy and incomplete data, in general the set of learned axioms is small and can be manually evaluated and improved by specialists. It can also be used to spot differences between intended and actual uses of ontologies, which are difficult to be predicted by ontology designers and laborious to be manually detected.

The impact of learned axioms can be significant, potentially allowing the detection of several errors or prediction of several new facts in the ABox with a single axiom. Moreover, when compared with ABox refinement approaches, ontology learning approaches have a clear decision process, which can be analyzed and improved by humans.

Ontology learning approaches can be substantially different, going into various research directions. Lehmann et al. (107) roughly classify such approaches into four main areas: ontology learning from text, linked data mining, concept learning, and crowdsourcing.

Ontology Learning from text mostly focuses on the generation of ontologies with text mining and information extraction methods. A prominent example is NELL project, which reads the web to add statements to its knowledge base and improves its performance over time, by taking user input about the quality of the extracted facts.

Linked Data Mining refers to the process of learning meaningful patterns in RDF graphs. Being able to detect the structure within the knowledge graphs can support the later creation of schemata and enable the detection of interesting associations between elements in the graph. Statistical schema induction (203) or statistical relational learning methods are examples of methods in this area.

Concept Learning is a direction of research that aims at learning schema axioms, such as class expressions, from existing ontologies and instance data. Most methods in this area are based on Inductive Logic Programming (113). While many algorithms, such as DL-FOIL (100) and OCEL (105) are generic supervised machine learning approaches for description logics, there are also specific adaptations to ontology learning (102), e.g., in terms of performance and usability.

An interesting alternative to purely automatic approaches is crowdsourcing ontologies. Formulating the task to be completed and providing incentives for people to contribute are additional challenges of this kind of approach. Examples of crowdsourcing in the field of ontology learning include taxonomy construction via Amazon mechanical turk, and games with a purpose for ontology population (35; 83).

2.7 Automatic ABox Refinement Approaches

In this section we give a brief overview of knowledge graph refinement approaches that focus on directly improving the ABox. We present some of the state-of-the-art methods and categorize them according to (137; 149).

2.7.1 Error Detection vs. Graph Completion

There are two main goals of knowledge graph refinement: adding missing knowledge to the graph, i.e., completion, and identifying wrong information in the graph, i.e., error detection. The former addresses the incompleteness problem, while the latter addresses the noise problem.

Both completion and error detection approaches can be further distinguished by the targeted kind of information in the knowledge graph. For example, some approaches are targeted towards completing/correcting entity type information, while others are targeted to (either specific or any) relations between entities, interlinks between different knowledge graphs, or literal values, such as numbers. The problem of completing a graph with relations between entities is often referred to in the literature as link prediction, while the problem of completing a graph with entity type information is often referred to as type prediction.

2.7.2 Internal vs. External Methods

Internal methods use only the knowledge contained in the knowledge graph itself to predict missing information. External methods use sources of knowledge – such as text corpora, other knowledge graphs or crowdsourcing – which are not part of the knowledge graph itself. Those external sources can be linked from the knowledge graph, such as knowledge graph interlinks or links to web pages (e.g., Wikipedia pages describing an entity), or exist without any relation to the knowledge graph at hand, such as large text corpora.

Some popular external methods focus on type prediction. E.g., Giovanni et al. (71) exploit Wikipedia links to predict new types, Aprosio et al. (5) use type information from DBpedia in different languages, while Gangemi et al. (67) and Kliegr (88) use textual information from DBpedia abstracts. In link prediction, some methods predict new relation assertions with learned text patterns from Wikipedia text (97; 214). Other approaches using question answering based on web search engines (210) as well as HTML web tables (170) have also been proposed.

Amongst the internal methods, there are mainly two kinds: latent and graph feature models, which will be discussed in more details next.

2.7.3 Latent vs. Graph Feature Models

Graph feature models rely on features which can be directly observed in a graph. The intuition behind these methods is that similar entities are likely to be related and that the similarity of entities can be derived from the neighborhood of nodes or from the existence of paths between nodes.

Latent features are those features which are not directly observed in the data. The main task of all latent feature models is to infer these features automatically from the data, while maximizing a triple score function over the whole dataset. In these models entities and relations are represented in lower-dimensional spaces. The intuition behind the learned representations is that the relationships between entities can be derived from interactions of their latent features.

Recently graph feature models have received a lot of attention and have become some of the best performing models on the link prediction task. There are different approaches to learning graph embeddings. There are tensor factorization models (e.g., RESCAL (139), TRESCAL (31)), translation models (e.g., TransE (15), TransH (208), TransR (111)), multilayer perceptrons (e.g., E-MLP (181), ER-MLP (51)) and neural tensor networks (NTN) (181). Other methods are based on word embedding methods which are trained on random walks of a knowledgraph (169; 38), however they are not conceived for the link prediction task and cannot score triples.

Graph feature models have also been used for link prediction, e.g. path ranking algorithm (PRA) (98) and subgraph feature extraction (SFE) (69), as well as for error detection, e.g. SDValidate (151). On the type prediction problem SDType (150) is an example of a graph feature model.

2.8 Evaluation of Knowledge Graph Refinement Methods

There are three main kinds of evaluation approaches for ABox automatic refinement (149): partial gold standard, silver standard and retrospective evaluation.

Partial Gold Standard: In this evaluation methodology a subset of graph entities or relations are selected and labeled manually. Other evaluations use external data as partial gold standards. For completion tasks, this means that facts that should exist in the knowledge graph are collected, whereas for correction tasks, a set of facts in the graph is manually labeled as correct or incorrect. Crafting a gold standard can be extremely costly, and the fact that the gold standard is created for a relatively small sample can significantly affect the quality of the evaluation.

Silver Standard: Another evaluation strategy is to use the given knowledge graph itself as a test dataset. For completion, that means a subset of the knowledge graph is removed from the training dataset and used as test. For error detection, wrong facts can be generated and added to the knowledge graph, then later be used for reference in the evaluation. Since the knowledge graph is not perfect, it cannot be considered as a gold standard, therefore we call it a silver standard. However, assuming that the given knowledge graph is already of reasonable quality, the evaluation results should be a good approximation of the actual results, with the advantage of being fully automatable.

Retrospective Evaluation: For retrospective evaluations, the output of a given approach is given to human judges for annotation, who then label suggested completions or identified errors as correct and incorrect. The quality metric is usually accuracy or precision, along with a statement about the total number of completions or errors found with the approach, and ideally also with a statement about the agreement of the human judges. Recall and other measures which rely on it cannot be calculated since the number of false positives cannot be calculated. One advantage of retrospective evaluations is that they allow a very detailed analysis of an approach's results. However the annotations are specific for a given method and cannot be reused.

For the evaluation of ontology learning approaches there are mainly gold standard and manual evaluation approaches. Given that the TBox is generally orders of magnitude smaller than the ABox, in many cases the manual evaluation is feasible. At the same time, the creation of a gold standard can be difficult since the definition of a good ontology is often debatable and application dependent.

Chapter 3

Related Work

In this section we present previous works related to the contents of this thesis. We cover related works in the area of knowledge graph synthesis (c.f. Section 3.1), ABox refinement – including type prediction, detection and correction of relation assertion errors (c.f. Sections 3.2, 3.3 and 3.4) – as well as TBox refinement – more specifically learning relation constraints, class expressions and rules with numerical attributes (c.f. Sections 3.5, 3.6. In the rest of this section we discuss more extensively and deeply the works related to those areas.

3.1 Synthesis of Knowledge Graphs for Benchmarking

There have been works which address the synthesis of knowledge graphs for benchmarking purposes. However, most efforts were focused on synthesizing A-box assertions for a specific T-box. Moreover, these works generate benchmarking datasets for various tasks in the Semantic Web, but none of them focus on the evaluation of link and type prediction methods.

Guo et al. (73) propose a method for benchmarking Semantic Web knowledge base systems on large OWL applications. They present the Lehigh University Benchmark (LUBM), which has an ontology for the university domain and includes the Univ-Bench artificial data generator (UBA), as well as a set of queries and performance measures. The data generator synthesizes A-boxes of arbitrary size to evaluate scalability. The data contains information about universities, which are artificially created based on some predefined restrictions, e.g. minimum and maximum number of departments, student/faculty ratio, which are based on arbitrarily defined ranges.

SP2Bench (177) is a SPARQL performance benchmark based on DBLP data. It features a data generator, which can create datasets of any given size. Similarly to UBA, the authors synthesize the A-box based on an existing T-box, in this case the DBLP ontology, and a dataset specific model used to generate the synthetic data. The model uses logistic curves and simple intervals to describe characteristics of the DBLP data, such as the number and types of publications, distribution of citations, and level of incompleteness over years.

Morsey et al. (130) created a SPARQL query benchmark based on DBpedia to evaluate knowledge base storage systems. They gather a set of real world queries extracted with query log mining, and run them on datasets of different sizes generated from DBpedia. Their "data generation" process consists of sampling the original DBpedia dataset and changing the entities namespace. Two sampling methods are considered: *rand*, which basically randomly selects a fraction of the triples, and *seed*, which first sample a subset of the classes, then instances of these classes are also sampled and added to a queue, with this process repeated until the target dataset size is reached.

Linked Data Benchmark Council (LDBC) (4) developed the social network benchmark (SNB) and the semantic publishing benchmark (SPB). The SNB which includes a data generator that enables the creation of synthetic social network data representative of a real social network. The data generated includes properties occurring in real data, e.g. irregular structure, structure/value correlations and power-law distributions. The benchmark covers main aspects of social network data management, including interactive, business intelligence and graph analytics workload. The SPB is similar to the SNB, but it concerns the scenario of a media organization that maintains RDF descriptions of its catalogue of creative works.

3.2 Type Prediction

The problems of inference on noisy data in the Semantic Web has been identified by Ji et al. (81) and Polleres et at. (156). Reasoning on noisy data can result on the propagation of errors and further damage the quality of the data. There have been solutions proposed for the specific problem of type inference in (general or particular) RDF datasets in the recent past, using strategies such as machine learning, statistical methods, and exploitation of external knowledge such as links to other data sources or textual information. One of the first approaches to type classification in relational data is discussed by Neville et al. (136). The authors train a machine learning model on instances that already have a type, and apply it to the untyped instances in an iterative manner.

Some works address slightly different inference problems. Instead of predicting instance types, Oren et al. (143) predict possible predicates for resources based on co-occurrence of properties. The approach discussed by Pohl (155) addresses the problem of mapping DBpedia entities to the category system of OpenCyc. They use DBpedia specific information – infoboxes, textual descriptions, Wikipedia categories and instance-level links to OpenCyc – and apply an a posteriori consistency check using Cyc's own consistency checking mechanism.

HYENA (219) is a multi-label classifier for named entity types based on hierarchical taxonomies derived from YAGO. Textual features extracted from the mentions of the named entity Wikipedia articles are used by the classifier, which consists of the SCN approach with siblings negative examples selection. There are several works on type prediction which exploit specific aspects of DBpedia. Aprosio et al. (5) introduced an approach which first exploits cross-language links between DBpedia in different languages to increase coverage. They use nearest neighbor classification based on different features, such as templates, categories, and bag of words of the corresponding Wikipedia article.

The *Tipalo* system (67) leverages the natural language descriptions of DBpedia entities to infer types, exploiting the fact that most abstracts in Wikipedia follow similar patterns. Those descriptions are parsed and mapped to the WordNet and DOLCE ontologies in order to find appropriate types. Giovanni et al. (71) exploit types of resources derived from linked resources, where links between Wikipedia pages are used to find linked resources (which are potentially more than the resources actually linked in DBpedia). For each resource, they use the classes of related resources as features, and use k-nearest neighbors for predicting types based on those features. However, none of those approaches can be trivially applied to datasets other than DBpedia.

SDType (150) uses links between resources as indicators for types, namely the ingoing and outgoing properties of instances. The method requires the prior distribution of types, as well as, for every property, a conditional probability distribution of object and subject types. Every property is assigned a weight, where maximum weight is given to properties that appear with a single type only, while the minimum weight is given to properties which are equally present in all types. Based on that, when predicting the types of an instance, SDType computes a confidence value for every type possible. Those types whose confidence value satisfies an arbitrarily defined minimum confidence threshold are assigned to the instance's prediction.

SDType is a simple and highly scalable method, whose complexity grows linearly with the number of statements in the knowledge base. According to the algorithm categorization by Silla Jr. et al. (179), SDType can be considered a global hierarchical multilabel classifier with multipath, non-mandatory leaf-nodes. SDType also generates predictions consistent with the type hierarchy. That is because the confidence of any non-root class will be always smaller or equal to that of its superclass if the model is learned on data with all subsumed type assertions.

LIFT (222) relies on the idea of class specific features for local classifiers on multilabel classification. The authors propose a method for generation of class specific features based on the distance of instances to the centroid of clusters computed for the positive and negative examples. Since this approach requires a clustering algorithm to be executed twice for each class, scalability is major issue.

Amongst the approaches discussed above and in (150), SDType is reportedly the best performing type predictor (150; 151), also outperforming RDFS reasoning. Therefore, in the experiments of Chapter 5 we have restricted ourselves to comparing hierarchical classification methods against SDType.

Statistical relational learning is an area which has a lot in common with type prediction. In fact, type prediction can be considered a special case of the link prediction problem, discussed earlier in this section, where instances are linked with types. However, there are no reported results for type prediction using link prediction approaches in the literature, therefore we cannot directly compare the results presented in the respective papers with our method. Link prediction works are presented in details in Section 3.3.

3.3 Relation Assertion Error Detection

The problem of relation assertion error detection in knowledge graphs has been researched by the Semantic Web community. A few methods have been proposed for cleansing large-scale LOD knowledge graphs, such as DBpedia and NELL, which contain many relation assertion errors that cannot be detected by reasoning methods (151). Absence of domain and range restrictions of relations or too general restrictions is one of the main causes of such problems.

SDValidate (151) exploits statistical distributions of types and relations, and (48) applies outlier detection on type-based entity similarity measures to detect erroneous relation assertions. These methods can effectively detect errors on DBpedia, however they require the existence of informative type assertions. Moreover, more complex errors containing wrong entities with correct types cannot be detected. A detailed survey including link prediction and error detection methods for knowledge graphs can be found in (149).

Link prediction, or knowledge graph completion (KGC), is a problem highly related to error detection. Despite being a different problem, KGC methods can also be used on the error detection problem. This kind of methods can be divided into graph-based, which relies on features which can be directly observed in the graph, and embedding methods, which learn embeddings that represent entities and relations in a latent feature space.

The Path Ranking Algorithm (PRA) (98) has shown that a logistic regression classifier using path features generated with random walks can be used for learning and inference in KGs and outperforms N-FOIL horn-clause inference on NELL (98). The approach has been improved with Sub-graph Feature Extraction (SFE) (69), which also simplifies some aspects of PRA. For instance, while PRA uses real-value features which correspond to the probabilities to reach o from s with a given path, SFE simply uses binary features which indicate if o can be reached from s or not. SFE not only reduces runtime by an order of magnitude when compared with PRA, but it also improves the qualitative performance.

In the recent years, knowledge graph embedding models, i.e., representations of knowledge graphs into lower-dimensional dense vector spaces, have received a lot of attention (207). Several different models have been developed for the knowledge graph completion problem and have brought improvements in performance.

There is a plethora of different embeddings models for knowledge graphs. One of the earliest embedding models is RESCAL (139), which performs tensor factorization on the knowledge graph's adjacency tensor, with the resulting eigenvectors corresponding to the entity embeddings and the core tensor the relations matrices. TRESCAL (31) extends RESCAL by exploiting entity types as well as domain and range restrictions of relations to improve the data quality and speed up the tensor factorization process. Neural Tensor Model (NTN) (181) represents each relation as a bilinear tensor operator followed by a linear matrix operator. Other early embedding models include Structure Embeddings (SE) (16), Semantic Matching Energy (SME) (14) and Latent Factor Model (LFM) (80).

Translation-based embeddings represent relations as translations between subject and object entities. TransE (15) was the first translation-based model proposed. In TransE entities and relations share the same embeddings space, while in TransH (208) and TransR (112) the translations are performed in the relations space, which is different from the entities space, therefore requiring projection matrices to map the entities onto the relations space. TransG (216) and CTransR (112) incorporate multiple relation semantics, where a relation may have multiple meanings determined by the entities pair associated with the relation. PTransE (110) extends TransE by considering relation paths as regular relations, which makes the number of relations considered grow exponentially.

Other approaches include DistMult (217), which uses dot product instead of translations to compute the triple scores. HolE (138) used circular correlation as an operator to combine the subject and object embeddings, Complex Embeddings (193) represents a triple score as the hermitian dot product of the relation, subject and object embeddings, which consist of real and imaginary vector components. ProjE (178) formulates the knowledge graph completion as a ranking problem, and it optimizes the ranking of candidate entities collectively. Discriminative Gaifman Models (141) embeds subgraph patterns incorporating first-order rules in the model. Some embedding models, such as RDF2Vec (169) and Global RDF vectors (39), are not conceived for the KGC task and cannot generate triple scores. Thus they cannot be directly used for error detection in the same way the other models mentioned earlier can.

Recently some works have raised doubts about the performance of new KGC embeddings models. Most of the experiments rely exclusively on two datasets (WN18 and FB15k), which contain many inverse relations (191). Therefore some of the models may exploit this characteristic and not necessarily perform as well on other KGs. It has also been shown that the presence of relations between candidate pairs can be an extremely strong signal in some cases (191). Moreover, recent works showed that a hyperparameter tuning has been overlooked and that a simple method, such as DistMult, can achieve state-of-the-art performance when well tuned (82).

3.4 Correction of Confusions in Knowledge Graphs

In the knowledge graph context, there are mainly error detection and link prediction approaches. Both are closely related to our problem: while error detection deletes wrong triples, link prediction aims at adding new triples. In both cases, the approaches learn a KG model which is capable of assigning confidence values to triples.

It is important to note that none of the link prediction approaches mentioned address the problem of covering the candidate triples space.Error detection approaches, such as SDValidate and PaTyBRED, focus on the detection of already existing erroneous triples. However, they do not address the problem of what to do with the detected wrong facts. No knowledge graph refinement approach exploits the assumption that some erroneous facts, because of confusions between entities, can be corrected by fixing such confusions. This assumption can be used reduce the candidate triples search space.

Rule-based systems, such as AMIE (66), cannot assign scores to arbitrary triples. However, they could also be used to restrict the candidate triples search space by identifying high confidence soft rules and using the missing facts from instances where the rule does not hold as candidates. Combining them with previously mentioned KG models would be an interesting line of research, however, it is out of the scope of this work.

Wang et al. (206) studied the problem of erroneous links in Wikipedia, which is also the source of many errors of DBpedia. They model the Wikipedia links as a weighted directed mono-relational graph, and propose the LinkRank algorithm which similar to PageRank, but instead of ranking the nodes (entities), it ranks the links. They use LinkRank to generate candidates for the link correction and use textual features from the description of articles to learn an SVM classifier that can detect errors and choose the best candidate for correction. While this is a closely related problem, which can help reduce the number of relation errors by improving the quality of the source data, their method cannot be directly applied on arbitrary knowledge graphs. Moreover, our approach takes advantage of the multi-relational nature of KGs, entity types, ontological information and the graph structure.

3.5 Relation Constraints Learning

As discussed in Section 3.3, most works on detection of relation assertion errors in knowledge graphs address the problem on the level of individual assertions. There are few works which attempt to derive reusable, higher-level artifacts, apart from domain and range restrictions.

One such approach has been proposed in (190). The authors provide means of learning additional domain and range restrictions for relations, which can then facilitate more fine-grained fact checking. The domain and range axioms learned are a reusable artifact, but are not always suitable for the complex scenarios induced by modern knowledge graphs.

Paulheim and Gangemi (153) have introduced an approach that clusters similar relation assertion errors. Those clusters can be more easily inspected by experts (e.g., by presenting them one typical, prominent example as a proxy for a class of errors), but the expert still needs to identify the cause and come up with a suitable fix manually.

The work presented in (148) aims at closing that gap by precisely pinpointing the cause of an error. For DBpedia, it is able to identify single axioms in the ontology or single mapping elements (i.e., the smallest building blocks of the creation process) that are responsible for a class of errors. It is, however, tightly tangled to the DBpedia creation process and cannot be trivially transferred to other knowledge graphs built with different methods.

Since we discuss the learning of constraints to be used for validating a knowledge graph, we target a problem which is similar to that of ontology learning or enrichment; a field in which quite a bit of related work exists. Rudolph (171) uses a class of OWL axioms that generalize domain and range restrictions, which support the conjunction of concepts. Statistical schema induction (SSI) (204) uses association rule mining to learn OWL 2 EL axioms, such as class and relation subsumptions, relation's domain and range restrictions, relation transitiveness. Bühmann and Lehmann (24) propose a method for enriching ontologies with OWL 2 axioms implemented in the DL-Learner framework. Regarding relation assertion constraints, domain and range restrictions and relation cardinalities (133) are the only kind of constraints which can be learned by these methods. A brief introduction to ontology learning and overview of the main approaches can be found in (107).

Gayo et al. (70) uses SHACL and ShEX to define constraints to validate and describe linked data portals. Arndt et al. (6) uses rule mining to learn RDF-CV (RDF Constraints Vocabulary). Swift Linked Data Miner (SLDM) (157) is the only system at the moment which can automatically learn SHACL constraints. However, it does not learn relation constraints, only class expressions.

Rule learning approaches, such as AMIE (66) and DL-learner (104), could in principle have some of their rules converted into SHACL constraints. Since they were not originally conceived for learning relation constraints, these approaches would need to be extended in order to support it. As of now there are no works in that direction.

3.6 Inductive Lexical Learning of Class Expressions

Since we presented the first approach towards unifying logical (data-based) and lexical (text-based) ontology learning, we describe related work in both areas.

Early data-based works essentially focused on demonstrating the PAC-learnability for various terminological languages derived from CLASSIC. In particular, Cohen and Hirsh investigate the CORECLASSIC DL proving that it is not PAC-learnable (40) as well as demonstrating the PAC-learnability of its sub-languages, such as C-CLASSIC (41), through the bottom-up LCSLEARN algorithm. These approaches tend to cast supervised concept learning to a structural generalizing operator working on equivalent graph representations of the concept descriptions. Recently, many approaches have been proposed that adopt the idea of *generalization as search* (128) performed through suitable operators that are specifically designed for DL languages (9; 79; 103) on the grounds of the previous experience in the context of ILP. There is a body of research around the analysis of such operators (105; 99) and studies on the practical scalability of algorithms using them (77). Supervised learning systems, such as YINYANG (79) and DL-Learner (101) have been developed and adoptions implemented for the ontology learning use case (102; 25). Also techniques from the area of data mining have been used for unsupervised ontology learning (203). As an alternative model, a new version of the FOIL algorithm (160) has been implemented, resulting in the DL-FOIL system (58). The general framework has been extended to cope with logical representations designed for formal Web ontologies (59).

Unlike logical approaches which have been developed to generate ontologies from structured data, lexical or NLP-based methods (213) draw upon the huge amounts of unstructured text available, e.g., on the web. Many of these methods combine lexico-syntactic patterns (e.g., Hearst patterns (75)) and linguistic resources like WordNet with machine learning techniques. While a growing number of ontology learning methods also leverages linked data or ontologies (e.g. FRED (158)), the results are mostly limited to atomic entities and simple axioms. An exception to this are pattern-based approaches to translating natural language definitions into class expressions such as LExO (202).

Altogether, we can see that attempts have been made to integrate semantic web data and logical inference into lexical approaches to ontology learning. However, there has been little if any work on integrating lexical evidence into logics-based ontology learning algorithms so far.

3.7 Learning Rules With Numerical Attributes

Related work consists of approaches for learning rules with numerical attribute intervals (also called quantitative rules) in association rule mining as well as Inductive Logic Programming (ILP). To the best of our knowledge, there is no prior work which focuses on the efficient exploration of the rules search space in ILP. The works presented in this section addresses different problems which are tightly related to our proposed approach.

The related works on association rule mining focus on learning rules of the form $(A_1 \in [l_1, u_1]) \wedge C_1 \Rightarrow C_2$, where A_1 is an uninstantiated numerical attribute, l_1 and u_1 are the lower and upper boundaries of A_1, and C_1 and C_2 are instantiated conditions. Srikant and Agrawal (183) use a priori discretization of the numerical attribute domain into fine partitions in order to treat them as regular categorical attributes. They learn rules for the individual partitions and then try to merge rules with adjacent partitions into wider intervals, but their approach does not guarantee the optimality of the generated intervals. Brin et al. (21) propose an algorithm with linear complexity for finding optimal intervals. It features a supervised bucketing technique, which collapses instances together, reducing input size without sacrificing optimality. Mata et al. (118) employ evolutionary techniques which do not require the discretization of the numerical attributes, and Salleb-Aouissi et al. (173) employ a genetic-base algorithm in order to find the optimal interval boundaries. Note that these algorithms could be integrated into our ILP extension for finding the optimal lower and upper boundaries, after we have detected an interesting rule with a numerical attribute with no intervals.

Inductive Logic Programming (ILP), introduced by (131) , is a state-of-the-art technique for learning concepts from examples, which we base our approach on. There are two main induction methods: top-down (which starts with specific clauses and searches for generalizations), and bottom-up (which starts with a general clause and searches for specializations). We use the top-down approach of FOIL (160) , which essentially consists of a covering and a specialization loop. The former ensures completeness, i.e. all positive examples are covered, whereas the latter ensures consistency, i.e. no negative examples are covered. In order to handle noisy data, we use a minimum expected accuracy (namely the probability that an example covered by the clause is positive) as a stopping criterion in the specialization loop. We also restrict the hypothesis space to safe datalog rules, which support arithmetical predicates providing the expressiveness required to define the intervals of numerical attributes.

3.8 Summary

This thesis addresses a variety of challenges involving the automatic refinement of knowledge graphs. The next chapters answer the research questions discussed earlier at Section 1.1 and fill the gaps in the literature, which are summarized below.

- Firstly we identified that there are no knowledge graph synthesizers capable of artificially generating data with characteristics relevant to link and type prediction tasks.

- Previous type prediction methods were quite limited and did not take the hierarchical structure of types. Moreover, type prediction has not been formulated as a hierarchical multilabel classification problem.

- Despite the wide body of research on the link prediction problem, there were previously no works addressing the detection of erroneous links, or which evaluated link prediction methods on the error detection task.

- There's no work on correcting detected erroneous relation assertions originated from confusions between entities, which is common type of error on datasets extracted from Wiki sources.

- Relation constraint learning methods are restricted to domain and range restrictions, which in many cases is not expressive enough to cope with complex domains.

- Logical class expressions learning methods, which uses facts in the knowledge graph to induce the expressions, have not been combined with lexical methods, which relies on textual data.

- There are very few works on learning rules with numerical attributes, and no work on efficiently learning such rules with inductive logic programming.

With the works presented in Chapters 4-10 those gaps are addressed one by one in the same order. Finally, in Chapter 11 we conclude the thesis, discuss future works and present important current challenges of knowledge graph refinement.

Part I

ABox Refinement

Chapter 4

Synthesizing Knowledge Graphs for Refinement Benchmarking

4.1 Introduction

Benchmarking is an important way of evaluating and comparing different methods for a given task. Having datasets with various characteristics is a crucial part of designing good benchmarking tests, allowing to thoroughly analyze the performance of a method under various conditions.

With the growing adoption and usage of Web-scale knowledge graphs, the data quality of those graphs has drawn some attention, and methods for improving the data quality, e.g., by predicting missing types and links, have been proposed. While there are a few benchmarking datasets for other tasks in the Semantic Web community, like SPARQL query performance (130; 177), ontology matching (34), entity linking (200), machine learning (167), and question answering (114), benchmarks for the task of type and link prediction are still missing. In contrast, the majority of approaches is only tested on one or few datasets, most prominently different versions of DBpedia, which makes it difficult to compare the approaches (149). Thus, it would be desirable to have benchmarking datasets with different characteristics, such as the number of entities, relation assertions, number of types, the taxonomy of types, the density of the knowledge graph, etc. Furthermore, it would be interesting to be able to have some control over these characteristics, vary them if necessary, and generate a knowledge graph following defined settings.

Generating data artificially for evaluation purposes is not something new. Data synthesizers have been used in various research areas. IBM Quest Synthetics Data Generator[1] is probably the most famous of them. It generates transaction tables for frequent pattern mining. There are also generators, e.g., for spatial-temporal data (188), clustering and outlier detection (33), data for information discovery and analysis systems (174), and high-dimensional datasets (3).

[1] http://www.philippe-fournier-viger.com/spmf/datasets/IBM_Quest_data_generator.zip

The overall goal is to synthesize a multitude of knowledge graphs to design benchmarkings for the tasks of link and type prediction. A first step to achieve this goal is to be able to replicate already existing datasets. In this chapter, we propose knowledge graph models, and a synthesis process that is able to generate data based on the models. To show the validity of the synthesis approach, our main goal is to replicate the evaluation results when performing link and type prediction with various state-of-the-art methods. We want to minimize the distance between the original dataset and the synthesized replicas for these measures, and also preserve method rankings. In our case, we select five methods for each task.

In order to be able to run systematic scalability tests with different approaches, we also explore the possibility to generate replicas of different sizes (number of entities and facts). The results should be preserved when varying the size of the synthesized data.

The rest of this chapter is structured as follows. We introduce our model for knowledge graphs in section 4.2, and discuss the synthesis approach in section 4.3. In a set of experiments, we discuss the validity of our approach in section 4.4, and conclude with a discussion the results.

4.2 Knowledge Graph Model

We define a knowledge graph $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is the T-box and $\mathcal{A}$ is the A-box containing relations assertions $\mathcal{A}_R$ and type assertions $\mathcal{A}_C$. We define N_C as the set of concepts (types), N_R as the set of roles (object properties) and N_I as the set of individuals (entities which occur as subject or object in relations). The set of relation assertions is defined as $\mathcal{A}_R = \{p(s, o)|p \in N_R \wedge s, o \in N_I\}$ and the set of type assertion as $\mathcal{A}_C = \{C(s)|C \in N_C \wedge s \in N_I\}$.

In our proposed model, we learn the joint distribution of types over instances. To that end, we compute $P(T)$, which is the probability of an individual having a set of types T. We define the set of types $\tau(s)$ of a given individual s as $\tau(s) = \{C|C(s) \in \mathcal{A}_\mathcal{C}\}$ and the set of individuals of given set of types T as $E_T = \{s|\tau(s) = T\}$. This is important because most knowledge graphs allow instances to have multiple types, and by modeling the distribution of instances over sets of types we can capture the dependencies between types, which is relevant for the problem described in this chapter. It is important to note that, e.g., the instance `Arnold_Schwarzenegger` with set of types $\{\texttt{Actor}, \texttt{Politician}, \texttt{BodyBuilder}\}$ is not considered to belong to $\{\texttt{Actor}, \texttt{Politician}\}$ when computing the distributions. With that, we make sure that $\sum_{T \in \mathcal{P}(N_C)} P(T) = 1$, where $\mathcal{P}(N_C)$ is the powerset of types containing all possible combinations of types

$$P(T) = \frac{|\{s|\tau(s) = T\}|}{|N_I|} \tag{4.1}$$

We also model the joint distribution of relations and the type set of their subject (T_s) and object (T_o), which we call $P(r, T_s, T_o)$. This distribution allows us to model how different types are related, and capture domain and range restrictions of relations in a fine grained way. We can model not only that the relation `playsFor` has domain `Athlete` and range `SportsTeam`, but also how athletes are distributed over more specific types (e.g., `FootballPlayer`, `BasketballPlayer`, etc.) and how teams are distributed over subclasses of `SportsTeam` (e.g., `FootballTeam`, `BasketballTeam`, etc.). Most importantly, we can model that `FootballPlayer playsFor FootballTeam` and `BasketballPlayer playsFor BasketballTeam`.

We model the joint distribution $P(r, T_s, T_o)$ with the chain rule (4.3). We decompose it into the distribution of relations over facts $P(r)$, conditional distributions of subject type set given relation $P(T_s|r)$ and a conditional distributions of object type set given subject type set and relation $P(T_o|r, T_s)$.

$$\begin{aligned} P(r, T_s, T_o) &= P(r)P(T_s|r)P(T_o|r, T_s) && (4.2) \\ P(r) &= \frac{|\{p(s, o) \in \mathcal{A}_R | p = r\}|}{|\mathcal{A}_R|} && (4.3) \\ P(T_s|r) &= \frac{|\{p(s, o) \in \mathcal{A}_R | p = r \wedge \tau(s) = T_s\}|}{|\{p(s, o) \in \mathcal{A}_R | p = r\}|} && (4.4) \\ P(T_o|r, T_s) &= \frac{|\{p(s, o) \in \mathcal{A}_R | p = r \wedge \tau(s) = T_s \wedge \tau(o) = T_o\}|}{|\{p(s, o) \in \mathcal{A}_R | p = r \wedge \tau(s) = T_s\}|} && (4.5) \end{aligned}$$

It is important to note that in case there are inconsistencies in the knowledge graph, such as domain/range violations or the assignment of inconsistent types, they are also captured in the distribution $P(r, T_s, T_o)$, and can be later replicated with their respective probabilities.

Besides the probability distributions of types and relations, individuals also follow a certain probability distribution, and not all relations have a uniform distribution w.r.t. their subjects and objects. In many cases, when selecting the individuals from E_{T_s} and E_{T_o}, there might be some bias which we should take into account. For instance, if we select $r = \texttt{livesIn}$, $T_s = \{\texttt{Person}\}$ and $T_o = \{\texttt{Country}\}$, we should not select the individual for `Country` based on a uniform distribution. The distribution should be biased towards more populous countries, e.g., the probability of selecting `China` should be much higher than that of `Vatican`. At the same time, for the $r = \texttt{capitalOf}$ with $T_o = \{\texttt{Country}\}$, the distribution of countries should be uniform since all the countries are equally likely to have a capital.

After selecting the relation r and type set of subject T_s and object T_o, we then need to select the subject and object individuals. Since in our synthesis process we first generate the individuals and their type assertions and then generate the relations assertions, there is a limited number of individuals belonging to a given type set T which we define as $n_T = |E_T|$.

Following those considerations, we compute the conditional distributions of subject and object individuals given a relation and type set of subject and object, which we call $P(e|r, T_s)$ and $P(e|r, T_o)$, respectively. To that end, we count the occurrences of subject individuals for all relations r and subject type set T_s, and occurrences of object individuals for all r and T_o. We then sort the individuals by frequency in descending order and fit a distribution model.

We need to select an instance from a finite set E_T, and we should be able to vary the size n_T in order to be able to scale the knowledge base up and down. Therefore we consider the use of uniform and exponential truncated distributions (c.f. Equations 4.6 and 4.7).

$$f(x,b) = \begin{cases} \frac{1}{b} & \text{, if } 0 \leq x < b \\ 0 & \text{, otherwise} \end{cases} \tag{4.6}$$

$$f(x,b) = \begin{cases} \frac{e^{-x}}{1-e^{-b}} & \text{, if } 0 \leq x < b \\ 0 & \text{, otherwise} \end{cases} \tag{4.7}$$

In truncated distributions, occurrences are limited to values which lie inside a given range. In the case of Equations 4.6 and 4.7, that interval is $0 \leq x < b$. It is important to use truncated functions, because when synthesizing relation assertions and selecting the individual for a given type, we can set $b = n_T$, and select an individual amongst the limited number of individuals that have the required type.

All distributions presented earlier in this section can effectively replicate some characteristics of a knowledge graph, such as in and out degree and density of the graph, however, they are not able to replicate more complex patterns involving paths in the graph. An example of such pattern in a knowledge graph containing data about families is that people who are married to the parent of a given child are also the parent of that child with some confidence. This pattern can be represented with the horn rule below.

$$\texttt{marriedTo(x,y)} \wedge \texttt{childOf(x,z)} \Rightarrow \texttt{childOf(y,z)} \quad [\mathit{conf} = 0.93]$$

Horn rules are basis of inductive logic programming (ILP) systems, such as ALEPH (132), WARMR (72), DL-Learner (100), and AMIE (66). There are also ILP extensions with probabilistic methods (161) and that can efficiently handle numerical attributes (125). We choose to use AMIE especially because of its better scalability in comparison to ALEPH and WARMR.

As most ILP systems, AMIE uses techniques to restrict the search space. AMIE mines only *closed* and *connected* rules. A rule is connected if all of its atoms are connected transitively to every other atom of the rule, and two atoms are connected if they share a variable or a constant. A rule is closed if every variable in the rule appears at least twice. Such rules do not predict merely the existence of a fact (e.g. `diedIn(x,y)` ⇒ `wasBornIn(x,z)`, which is connected rule, but not closed), but the concrete arguments for it (e.g. `diedIn(x,y)` ⇒ `wasBornIn(x,y)`).

We use the horn rules learned by AMIE in our KB model in order to represent more complex patterns and use their associated PCA (partial close-world assumption) confidence value in the synthesis. In our model, we are able to ensure various relation characteristics. The RDF Schema domain and range restrictions can be ensured by the joint distribution $P(r, T_s, T_o)$. The horn rules can model symmetric, transitive, equivalent, and inverse properties.

To cover even more complex schemas, we additionally learn functionality, inverse functionality and non-reflexiveness from the data. All relations which do

Algorithm 1 Knowledge base synthesis process

```
function GEN_KB($n_e, n_f, P(T), P(r), P(r, T_s, T_o), \mathcal{H}$)
    $\mathcal{A} \leftarrow \emptyset$                                        ▷ Create empty A-Box
    $E \leftarrow \{\}$                               ▷ Map of type sets and their entities
    for $i \leftarrow 1$ to $n_e$ do                                   ▷ synthesize entities
        $T_i \leftarrow$ randomly choose from $P(T)$
        $E[T_i] \leftarrow E[T_i] \cup \{e_i\}$
        for $C \in T_i$ do
            $\mathcal{A} \leftarrow \mathcal{A} \cup \{C(e_i)\}$
    $i \leftarrow 0$
    while $i < n_f$ do                                 ▷ synthesize relation assertions
        $r_i, T_{s_i}, T_{o_i} \leftarrow$ randomly choose from $P(r, T_s, T_o)$          ▷ use chain rule
        $s_i \leftarrow$ SELECT_ENTITY($E[T_{s_i}], P(e|r_i, T_{s_i})$)
        $o_i \leftarrow$ SELECT_ENTITY($E[T_{o_i}], P(e|r_i, T_{o_i})$)
        if VERIFY_TRIPLE($s_i, r_i, o_i$) then
            $\mathcal{A} \leftarrow \mathcal{A} \cup \{r_i(s_i, o_i)\}$
            CHECK_HORN_RULES($\mathcal{A}, (s_i, r_i, o_i), \mathcal{H}$)
            UPDATE_DISTRIBUTION($P(r, T_s, T_o)$)
            $i \leftarrow i + 1$
    return $\mathcal{A}$
```

not have any same individual as both subject and object of a triple are considered non-reflexive, all relations with object cardinality of 1 are considered functional, and with subject cardinality of 1 are considered inverse functional. Learning these characteristics from data allows us to detect relations which might not have been conceived as, or not defined as such in the schema, but which in the available data present the characteristics. For instance, a dataset with the `childOf` relation, which is not functional, might contain data about people which have exclusively one child, and with our approach we ensure this characteristic is replicated.

4.3 Synthesis Process

Algorithm 1 summarizes the process of synthesizing a knowledge graph. As input, it uses the probability distributions $P(T)$, $P(r, T_s, T_o)$, $P(e|r, T_s)$, and $P(e|r, T_o)$, a set of horn rules $\mathcal{H}$, as well as the desired number of individuals n_e and relation assertions n_f to be synthesized.

The function VERIFY_TRIPLE first verifies if the exact same triple is already present in the synthesized KG. Then it checks whether functionality, inverse functionality, and non-reflexiveness are satisfied. That is, it verifies if there is no assertion with the given subject already present in the KG for functional relations, no assertion with the given object for inverse functional relations, and the given subject and object are different individuals for non-reflexive relations.

The function CHECK_HORN_RULES ensures that the patterns learned with the horn rules are replicated in the synthesized data. It checks if a newly synthesized fact triggers any of the learned horn rules. If a rule is triggered, the rule will produce a new fact with a probability equal to that of its confidence. The new facts produced by rules also need to be checked against the horn rules again, which means that the CHECK_HORN_RULES function is called recursively until it does not produce any new facts.

The function UPDATE_DISTRIBUTION makes sure that the original distribution $P(r, T_s, T_o)$ is not distorted by the production of new facts from horn rules, which may not follow $P(r, T_s, T_o)$. Therefore, it is necessary to adjust the joint distribution in order to compensate this effects. We do that by simply keeping counts for the relations, subject and object type sets, and based on the number of facts to be synthesized and the distribution of already synthesized facts we can adjust $P(r, T_s, T_o)$.

Another detail not shown in Algorithm 1 is the use of a pool of subjects for functional and pool of objects for inverse functional relations. We do that in order to avoid generating facts which violate the functionality and inverse functionality restrictions. If no pools are considered, the probability of generating violating facts for a given relation increases linearly with the number of already existent facts. With the pools, all individuals of a given type are initially in the pool, and whenever an individual is picked to generate a new fact, this individual is removed from the pool and cannot be picked again, therefore preventing the violations.

In the synthesis process some characteristics can be easily changed. Noise can be introduced by smoothing the distribution $P(r, T_s, T_o)$, making the probability for invalid combinations of relations, subject and object types non-zero. The density of the knowledge graph can be altered by modifying the ratio n_f/n_e between number of facts and individuals. It is possible to change the scale of the synthetic knowledge graphs by simply multiplying the original number of individuals n_e and facts n_f by a constant. That is, assuming that the number of relations in the knowledge graph grows linearly with the number of individuals.

However, some knowledge graphs might have relations which are quadratic, e.g. `owl:differentFrom` that indicates individuals that are not the same. Therefore, for the quadratic relations of a knowledge base, we need to scale the number of relation assertions quadratically with the number of individuals. This kind of relations are rather rare, and they can be difficult to automatically detect. We use a simple heuristic based on thresholds for the average number of different objects per subject and different subjects per object. If both thresholds are reached, we assume the relation to be quadratic.

One important characteristic is that the synthesis process is based on pseudo random number generators (PRNG), therefore, the process is deterministic and identical datasets can be generated if the same seed is used. By using different seeds, it is also possible to generate different datasets from the same model and with similar characteristics, allowing us to test the stability of methods.

4.4 Experiments

The link prediction task consists of predicting the existence (or probability of correctness) of edges in the graph (i.e., relation assertions). This is important since existing knowledge graphs are often missing many facts, and some of the edges they contain are incorrect. Nickel et al. (137) present a review of multirelational models, many of which have been used for the link prediction task. We select five popular methods to be used in our experiments: Path Ranking algorithm (98), SDValidate (151), Holographic embeddings (HolE) (138), Translation embeddings (TransE) (15) and RESCAL (139). In our experiments, we evaluate the prediction of relation assertions only. All the measurements reported were obtained using 5-fold cross-validation. The test set consists of the 20% of positive positive triples selected in the cross-validation, plus negative examples. There are the same number of positive and negative examples in the test set, and the negative examples are generated by corrupting each of the positive triples following the method described by Bordes et al. (15).

Type prediction can be considered a subtask of link prediction where we are interested on prediction links for the relation `rdf:type`. There are several type prediction approaches which rely on external features (219; 5; 67; 71), however, we concentrate on methods which rely on features extracted from the knowledge graph. The methods used in the experiments are SDType (150) and SLCN (124), as well as the state-of-art multilabel classifiers MLC4.5 (37), MLP (223) and MLkNN (224) – multilabel versions of decision tree, multilayer perceptron and k-nearest neighbors – with ingoing and outgoing links used as features as described in (124).

As input knowledge graphs, we use Wikidata, DBpedia (2015-10), and NELL. We use the following smaller domain specific datasets: Thesoz[2], Semantic Bible[3] AIFB portal[4], Nobel Prize[5] and Mutagenesis. We also select four of the largest conference datasets from the Semantic Web dog food corpus[6], i.e., LREC2008, WWW2012, ISWC2013, and ESWC2015. Some relevant statistics about the datasets used in the experiments are shown in Table 4.1.

For every input KG, we synthesize replicas of three different sizes increased by factors of 10. For smaller datasets we also scale the replicas up. On the Semantic Web dog food datasets we synthesize replicas of sizes 10%, 100% and 1000%. For large datasets we scale the replicas down (DBpedia and Wikidata replicas are of sizes 0.01%, 0.1% and 1%, and the remaining datasets 1%, 10% and 100%).

We use the scikit-kge[7] implementation of HolE, TransE and RESCAL, and the

[2] http://www.gesis.org/fileadmin/upload/dienstleistung/tools_standards/thesoz_skos_turtle.zip

[3] http://www.semanticbible.com/

[4] http://www.aifb.kit.edu/web/Web_Science_und_Wissensmanagement/Portal

[5] http://www.nobelprize.org/nobel_organizations/nobelmedia/nobelprize_org/developer/manual-linkeddata/terms.html

[6] http://data.semanticweb.org/dumps/conferences/

[7] https://github.com/mnick/scikit-kge

Dataset	Entities	Types	Rels	Type ass.	Relation ass.	Density
Wikidata	19060716	474	482	40198183	18955236	$1.082 \cdot 10^{-10}$
DBpedia	4940352	1027	646	31521734	14747048	$9.353 \cdot 10^{-10}$
NELL	1475674	276	248	5565472	174621	$3.233 \cdot 10^{-10}$
AIFB	27100	63	82	59613	59349	$9.855 \cdot 10^{-7}$
Mutagenesis	14157	91	4	48111	26533	$3.310 \cdot 10^{-5}$
SemanticBible	789	71	31	2563	2482	$1.286 \cdot 10^{-4}$
Thesoz	48540	10	16	109960	275430	$7.306 \cdot 10^{-6}$
NobelPrize	10013	23	18	19506	30148	$1.671 \cdot 10^{-5}$
ESWC2015	1285	16	25	1285	4062	$9.840 \cdot 10^{-5}$
ISWC2013	2548	20	39	2545	9992	$3.946 \cdot 10^{-5}$
WWW2012	3836	22	43	3907	15406	$2.435 \cdot 10^{-5}$
LREC2008	3502	7	24	3502	16514	$5.611 \cdot 10^{-5}$

Table 4.1: Statistics about the datasets used in the experiments

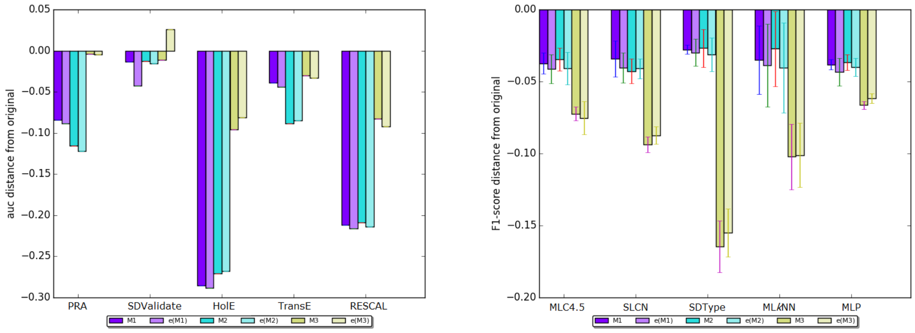
scikit-learn implementation of MLkNN, MLC4.5 and MLP. We implemented the remaining methods ourselves. The proposed synthesis process code is available to download.[8]

The evaluation measures used in the link experiments are the area under the precision-recall curve (PR AUC) and area under the ROC curve (ROC AUC). For the type prediction experiments we use micro-averaged F_1-score and accuracy. We compute the distance of these evaluation measures between the results on the original datasets, and their synthetic replicas. In order to compare the ranking of methods, we use the Spearman-ρ rank correlation coefficient. All the results reported in this chapter were obtained with 5-fold cross-validation.

In order to evaluate how the different parts of the proposed knowledge base model affect the results on link and type prediction tasks, we use 6 different models in our evaluation: M1, M2, M3, e(M1), e(M2) and e(M3):

- M1 is the simplest version, which considers only the distributions $P(T)$ and $P(r, T_s, T_o)$. The bias to selection of individuals is not considered, and individuals are always selected from an uniform distribution. No relation characteristics (apart from domain and range restrictions covered by $P(r, T_s, T_o)$) are considered.
- M2 is M1 plus functionality, inverse functionality and non-reflexiveness of relations.
- M3 is M2 plus the horn rules learned with AMIE.
- The models e(Mi) are the model Mi plus the biases to selection of individuals $P(e|r, T_s)$ and $P(e|r, T_o)$.

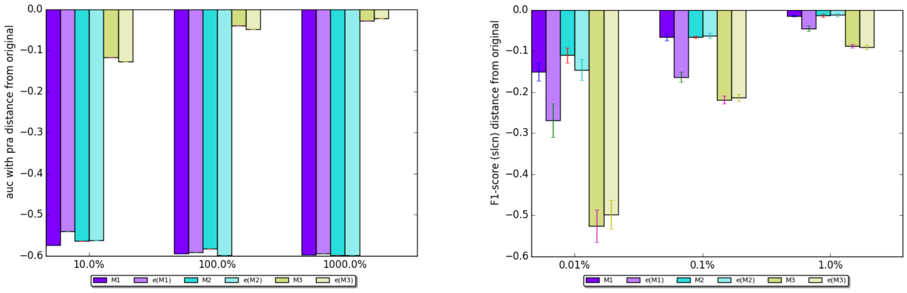
[8] https://github.com/aolimelo/kbgen



(a) Link prediction PR AUC distance on Nobel Prize dataset

(b) Type prediction F_1-score distance on Wikidata

Figure 4.1: Distances of performance measures to original datasets



(a) Link prediction on LREC2008 (PRA)

(b) Type prediction on DBpedia (SLCN)

Figure 4.2: Effect of scaling the replica sizes up and down

We use AMIE with its default parameter settings (i.e., no rules with constants, maximum rule length = 3, confidence computed with PCA, minimum support = 100 examples, minimum head coverage = 0.01).

We use PRA with maximum path length of 3 for all datasets. For HolE, TransE and RESCAL we learn embeddings with 20 dimensions and maximum of 100 epochs. While this may not be the optimal settings for most datasets, we consistently use the same settings throughout all of our experiments, since our aim is not to achieve optimal results, but to show that the benchmark synthesis works as desired.

Figure 4.1a shows an example of PR AUC distance on link prediction from for the Nobel Prize datasets between original and replica (100% size) with the 5 selected methods. It is clear that the use of horn rules significantly improves the results, as M3 and e(M3) performs better than the other methods, except from SDValidate, which relies on exclusively on distributions of relations and object types and does not exploit more complex path patterns.

Figure 4.1b shows an example of F_1-score distance on type prediction for Wikidata between original and replica (0.1% size). It is noticeable that horn rules do not improve the results, as M1, M2, e(M1) and e(M2) perform better than M3 and e(M3). This is explained by the fact that most of the evaluated type prediction methods rely solely on ingoing and outgoing links of entities. Moreover, as explained in Section 4.2, horn rules can disturb the original distribution $P(r, T_s, T_o)$, which is crucial for the replication of ingoing and outgoing links.

Tables 4.2 and 4.3 show a summary of the results obtained over all datasets for type prediction and link prediction, respectively. The values with subscript *all* report the average of the results over all different sizes of replicas, while those with subscript *large* report the averages over the largest size of replicas only. We do that because different models, especially M3 and e(M3), perform worse than others for smaller replica sizes, and we also want to know how the models perform when ruling out this effect.

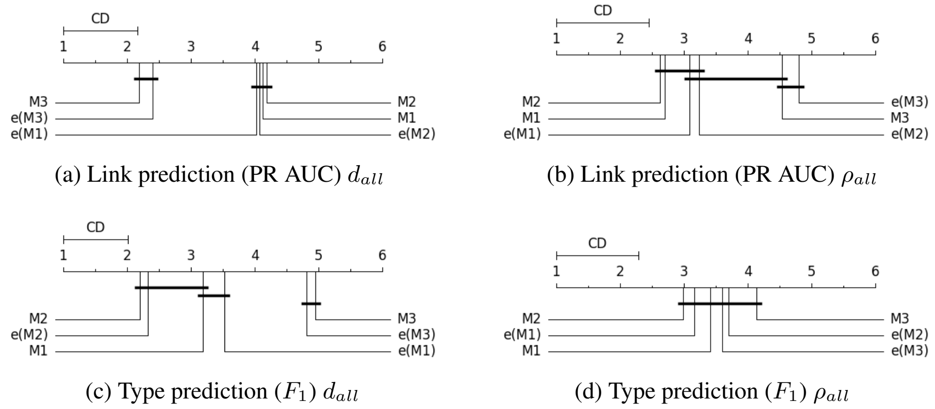
The results of Table 4.2 indicate that in terms of distance, M3 is the best method overall, however, when it comes to preserving the rankings, the results become more mixed. It is clear that introducing the horn rules does have a positive effect on the model, especially for the distances which are reduced to less than half of that of other models. In Table 4.3 we can see that M2 is the best overall in terms of distance for both PR and ROC AUC, while for the rankings, the use of horn rules again have a positive impact with M3 being the best method overall. The link prediction results were reported for all datasets apart from DBpedia and Wikidata. Because of the large size of these two datasets and the complexity of the approaches, the experiments did not finish in less than a week.

		M1	e(M1)	M2	e(M2)	M3	e(M3)
PR AUC	ρ_{all}	0.527	0.643	0.567	0.607	0.643	**0.653**
	ρ_{large}	0.640	0.740	0.590	0.580	0.730	**0.800**
	d_{all}	0.243	0.247	0.247	0.245	**0.112**	0.115
	d_{large}	0.215	0.228	0.216	0.219	**0.082**	0.089
ROC AUC	ρ_{all}	0.647	0.613	**0.657**	0.613	0.610	0.577
	ρ_{large}	0.650	0.610	0.640	0.630	**0.670**	0.620
	d_{all}	0.231	0.230	0.231	0.231	**0.109**	0.111
	d_{large}	0.211	0.215	0.208	0.211	**0.087**	0.095

Table 4.2: Summary of the link prediction results

		M1	e(M1)	M2	e(M2)	M3	e(M3)
F_1-score	ρ_{all}	0.208	0.221	0.195	0.259	**0.362**	0.265
	ρ_{large}	0.343	0.273	0.251	0.400	**0.456**	0.410
	d_{all}	0.086	0.098	**0.082**	0.083	0.131	0.130
	d_{large}	0.059	0.065	**0.055**	0.057	0.083	0.084
Accuracy	ρ_{all}	0.290	0.334	0.315	0.343	**0.406**	0.307
	ρ_{large}	0.470	0.420	0.357	0.498	**0.502**	0.433
	d_{all}	0.061	0.064	**0.057**	0.061	0.065	0.066
	d_{large}	0.056	0.060	**0.054**	0.057	0.061	0.062

Table 4.3: Summary of the type prediction results



(a) Link prediction (PR AUC) d_{all}

(b) Link prediction (PR AUC) ρ_{all}

(c) Type prediction (F_1) d_{all}

(d) Type prediction (F_1) ρ_{all}

Figure 4.3: Nemenyi Critical distance diagrams for link and type prediction

We also perform the Nemenyi test in order to find how significant the differences of the evaluated models is, both in terms of distance and ranking. Figure 4.3 shows the critical distance diagrams. For the distances d the models on the left side are the best performers, since lower distances are desired, while for Spearman's rank correlations ρ the models on the right side are the best performers, since higher correlations are desired.

In Figure 4.3a we can see that PR AUC distances on link prediction between the models with horn rules (M3 and e(M3)) and the others is very significant, while the differences in terms of Spearman-ρ from Figure 4.3b are closer to the critical distance (CD). We can also observe that the difference between M3 and e(M3) is not significant, indicating that the use of bias to selection of instances does not have a great impact. One possible explanation for that is the fact that, in order to simplify our model and abstract from specific instances, we assume that, for a given type set, the most frequent instances are always the same. That is, if we consider the type set {`Country`} as object of `livesIn` and `beatifiedPlace`, we assume that the most frequent country in both cases is the same individual, while in reality the most frequent country for `livesIn` would be China and for `beatifiedPlace` Italy. Since the computation of the bias can be very expensive, especially for larger datasets with high number of types and individuals, M3 would be a more reasonable choice than e(M3).

When analyzing Figure 4.3c, we notice that, in terms of F_1-score distance, the M2, e(M2) and M1 are not significantly different from each other, and the use of horn rules has a significant negative effect. The Spearman-ρ from Figure 4.3d values are very close to each other, without any significant difference between the evaluated models.

We illustrate the difference in runtime for the synthesis processes with different methods with Figure 4.4. The plot shows the number of facts generated over time for the ESWC2015 dataset. It is clear that M3 and e(M3) are significantly slower

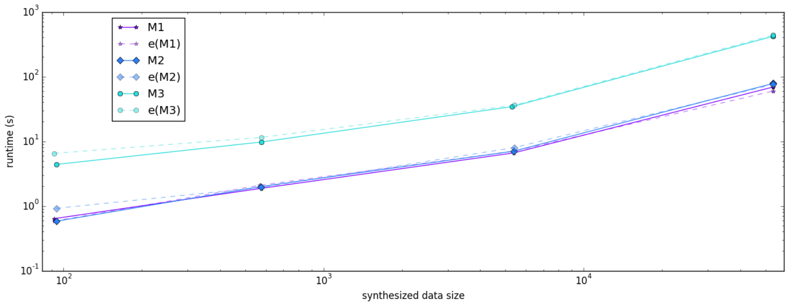


Figure 4.4: Synthesis process runtime over dataset size for the ESWC2015 model

than the others. It is also worth noting that these two models require horn rules, which need to be learned with AMIE further increasing the model learning time.

4.5 Conclusion

In this chapter, we have proposed a knowledge graph model and synthesis process which is able to capture essential characteristics of existing knowledge graphs, which allows us to create replicas of those graphs at different scales. Overall, the model M3 was the best performer, and the use of horn rules significantly improved the results. The use of a bias to selection of subject and object individuals did not show any significant improvement. In general, we recommend the use of M3, unless the objective is to replicate the results of type prediction on a single methods, without performing any comparisons. In Chapter 11 we summarize the conclusions of this and other works presented throughout this thesis and discuss possible future works.

Chapter 5

Type Prediction using Hierarchical Multilabel Classification

5.1 Introduction

Type information plays an important role in Semantic Web (SW) knowledge bases, with type assertion axioms, defined by the `rdf:type` relation, being one of the atomic building blocks of knowledge bases. Many datasets suffer from type assertion incompleteness. For example, for DBpedia(12), the upper bounds for completeness of DBpedia 3.8 types are estimated to be at most 63.7%, with at least 2.7 million missing type statements, while YAGO types in DBpedia 3.8 are estimated to be at most 53.3% complete(151). For example, `Arnold_Schwarzenegger` in DBpedia is assigned only the type `OfficeHolder` and none of the other suitable types, such as `Actor` and `BodyBuilder`.

A possible way to automatically infer type information on the Semantic Web is the use of reasoning, e.g., standard RDFS reasoning via entailment rules. However, reasoning methods are sensitive to noisy data, and since open knowledge bases created by crowdsourcing and/or heuristics are often noisy, logic-based reasoning approaches are likely to multiply errors. Statistical approaches, on the other hand, are more robust to noise, since they do not rely on the quality of the T-box axioms and are less influenced by single wrong A-box axioms. Therefore, they are considered to be more suitable for the type prediction task (150).

One example that illustrates this problem is that in DBpedia the Album entity `Abbey_Road` from the Beatles, is confused with the Musical Studio entity `Abbey_Road_Studios` in some triples. The fact `No_Reply_(song) recordedIn Abbey_Road` would lead to the inference that `Abbey_Road` is not only an Album but also a Populated Place, which is the range of the property `recordedIn`. In (150), we have shown that RDFS reasoning is prone to propagate noise, whereas heuristic inference is suitable to limit the influence of noise. Therefore, in this work, we pursue the use of heuristic inference methods.

Since most Semantic Web knowledge bases organize the possible types as hierarchies (defined in ontologies), we propose to model the type inference problem in noisy and incomplete knowledge bases as a *hierarchical multilabel classification* problem. It is *hierarchical* because we assume the types to be structured in a hierarchy, and it is *multilabel* because instances are allowed to have more than one type. For example, in a knowledge base with the type hierarchy depicted in Figure 5.1, the instance `Arnold_Schwarzenegger` should be typed as `OfficeHolder`, `Actor`, and `BodyBuilder`, as well as their generalizations `Artist`, `Athlete`, and `Person`, which can be inferred from type hierarchy.

As SW knowledge bases, especially cross-domain ones, can have a large number of types, the high dimensionality of the label space may challenge a multilabel classification algorithm in many ways. First, the number of training examples annotated with each type, in particular those in the lower levels of the hierarchy and in the long tail of an uneven distribution, will be significantly smaller than the total number of examples. This is similar to the class imbalance problem in single-label classification (164). Second, the computational cost of training a multilabel classifier may be strongly affected by the number of labels (195).

Due to the presence of ontologies and their type hierarchies on the Semantic Web, viewing type prediction as a hierarchical machine learning problem is the most natural translation of the type prediction problem to a machine learning problem. However, it has never been viewed like that – to the best of our knowledge, all machine learning based methods for type prediction in SW knowledge bases proposed so far flatten the problem to non-hierarchical classification (149). One possible reason that hierarchical multilabel classification has not been applied in the field may be scalability issues when applying those methods to large-scale SW knowledge bases.

In this chapter, we propose *SLCN* (for *Scalable Local Classifier Per Node*), a modification of the *local classifier per node* approach, which improves the scalability by performing local sampling, feature selection, and class balancing. We show that our approach outperforms the current state of the art approaches for type prediction in SW knowledge bases, and does so in a more scalable way than existing algorithms for hierarchical multi-label classification.

The rest of this chapter is structured as follows. First, we briefly introduce the foundations of hierarchical multilabel classification in Section 5.2, followed by a problem statement in Section 5.3. We outline the proposed approach in section 5.4, and report the outcome of experiments on various SW knowledge bases in Section 5.5. We conclude give an outlook of possible future works in Section 5.6.

5.2 Preliminaries

In this section, we lay out the foundations of hierarchical multilabel classification used in this work.

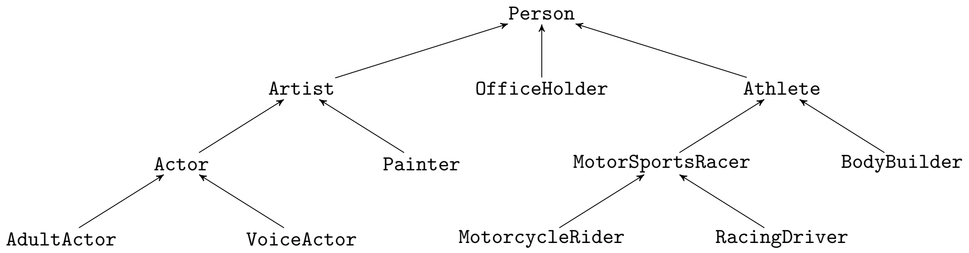


Figure 5.1: A subset of the DBpedia type hierarchy

5.2.1 Multilabel Classification Approaches

In the multi*label* classification problem, there are multiple classes and, contrary to the single-label multi*class* classification problem, instances are allowed to belong to more than one class. We define the set of classes as $C = \{c_1, ..., c_{|C|}\}$, and we represent the multilabel of an instance x with a binary vector $y = (y_1, ..., y_{|C|}) \in \{0, 1\}^{|C|}$.

Some of the existing multilabel classification approaches are standard binary classification algorithms which have been adapted to the multilabel task, without requiring problem transformations. This includes, e.g., *AdaboostMH*(175), *MLkNN*(224) and *BPMLL*(226). Other approaches, such as *Binary Relevance* (BR),*Classifier Chains*(164) (CC), *Label Powerset* (LP), and Random k-Labelsets (RAKeL)(196), transform the multilabel problem into a set of binary classification problems.

Binary Relevance (BR) is the simplest transformation approach, where a binary classifier is trained for each class assuming the classes are mutually independent. More complex transformation methods, such as *Classifier Chains*(164) (CC) and *Label Powerset* (LP), can model dependencies between the classes. There are also ensemble methods, such as Ensembles of Classifier Chains (ECC)(164) and Random k-Labelsets (RAKeL)(196), where several classifiers are trained on different subsamples and combined into a single model.

These approaches are agnostic with respect to a hierarchy relations among the labels, and hence, they do not necessarily guarantee the predicted classes to be consistent with the hierarchy.

5.2.2 Hierarchical Multilabel Classification Approaches

The hierarchical multilabel classification problem is similar to the multilabel classification problem, but the classes C are structured in a hierarchy G. The labels of an instance should be consistent with G, i.e., if an instance belongs to a non-root class then it must also belong to its ancestors (i.e., $c_i \sqsubseteq c_j \wedge y_i = 1 \rightarrow y_j = 1$). The class hierarchy can be of two types: a tree, which allows nodes to have a single parent only, and a directed acyclic graph (DAG) which allows nodes to have multiple parents.

As pointed out by Silla et al. (179), most of the current literature focuses on working with trees as it is a simpler problem. There are mainly two types of hierarchical multilabel classification approaches: local and global classifiers. The main difference is that the former breaks down the classification problem into smaller and simpler problems exploiting the class hierarchy, while the latter considers the problem as a whole, learning a single more complex model. In the next subsections we present these approaches in more details.

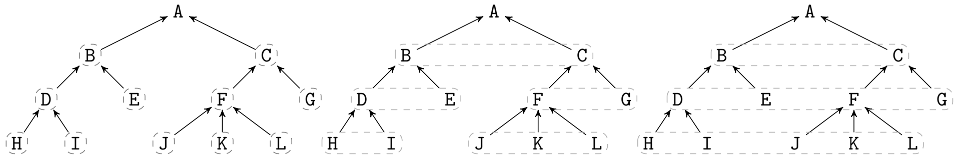
Local Classifier Approach

The hierarchy is taken into account by using a local information perspective to transform a multilabel classification problem into a set of simpler subproblems. This is a kind of transformation approach, since for every subproblem works the local classifier is trained on a different transformed dataset. According to Silla et al. (179), there are mainly three approaches of using local information: *local classifier per node*, *local classifier per parent node*, and *local classifier per level*. The local hierarchical classification algorithms share a similar top-down approach in their prediction phase, where the classifier first predicts its first-level (most generic) class of an instance, then it uses that predicted class to reduce the choices of classes to be predicted at the second level (the children of the classes predicted at the first level), and so on, recursively, until the most specific prediction is made.

Local Classifier Per Node (LCN): The local classifier per node approach consists of training one binary classifier for each node of the class hierarchy. Each local binary classifier predicts whether an instance belongs to the class associated with the node or not. There are two main ways to define the training set of the local binary classifiers, which are called *negative examples selection policies*. One is the *all* approach, which uses all instances to train all local classifiers, and *siblings*, which uses the instances belonging to a node's class and its siblings' classes to train the local classifiers. A comparison of different negative example selection approaches is made in (54) and (57). The results indicate that both approaches have roughly similar performances, however, *siblings* is more scalable than *all*.

Local Classifier Per Parent Node (LCPN): In this approach, a local multilabel classifier is learned for every non-leaf node in the hierarchy. The labels are the direct child nodes and the training instances are those which belong to the parent node class. If each multilabel problem is transformed into a set of binary problems with the binary relevance method, this is equivalent to local classifier per node. Depending on the choice of the local multilabel classifier, it is possible to model dependencies between sibling nodes.

Local Classifier Per Level (LCL): This is the type of classifier approach least used so far on the literature. The local classifier per level approach consists of training one multilabel classifier for each level of the class hierarchy. That means it is prone to class-membership inconsistency and therefore requires a post-processing step to prevent it. In the literature this approach was only mentioned as a possible approach by Freitas et al. (64), and used as a baseline comparison method in (36) and (43). Moreover, there is no publicly available implementation of this kind of approach.



(a) LCN (b) LCPN (c) LCL

Figure 5.2: Hierarchical multilabel classification local classifier approaches

Figure 5.2 illustrates the difference between the three local classifier approaches. The dashed closed curves indicate the set labels of each local classifier. In the case of LCN (5.2a), each of the eleven local binary classifiers predicts whether an instance belongs to its correspondent class or not. For LCPN (5.2b), there are five local multilabel classifiers, whose labels are sibling nodes. For LCL (5.2c), there are three local multilabel classifiers, whose labels are the nodes of each level of the hierarchy.

Global Classifier Approach

In contrast to local classifier approaches, the global classifier approach (also known as *big bang approach*), learns one single classification model built from the training set, taking into account the class hierarchy as a whole during a single run of the classification algorithm. When used during the prediction phase, each instance is classified by the induced model, a process that can assign classes at potentially every level of the hierarchy to the instance. Global classifier approaches lack the kind of modularity for local training of the classifier that is a core characteristic of the local classifier approaches.

An example of global classifier approach is MLC4.5 (37), which is a decision tree algorithm adapted to handle multilabel data. A single decision tree is created for the classifier, where each leaf node contains a vector with the class distributions. This method guarantees consistency with the hierarchy, as the probability of a class in the class distribution cannot be smaller than that of its children. Therefore, for any probability threshold, the generated prediction will be consistent with the hierarchy.

5.2.3 Evaluation Measures

Silla Jr. et al. (179) recommend the use of hierarchical loss (*h-loss*), and the hierarchical precision (*hP*), recall (*hR*), and F-measure (*hP*) to evaluate hierarchical multilabel classifiers. In this chapter, we also use the hamming loss (*hamm*), which is commonly used in (non-hierarchical) multilabel classification and serves as basis for the *h-loss*.

The *hP*, *hR*, and *hF* (86) are the micro-averaged measures of precision, recall and F-measure per class. By using the micro average, each class is weighted according to the label frequencies. Equations 5.1, 5.2 and 5.3 show the definition of these measures, where tp_i, fp_i and fn_i denote respectively the number of true positives, false positives and false negatives of the class c_i. Similarly to their binary class versions, *hP*, *hR* and *hF* values range is in the interval $[0, 1]$.

$$hP = \frac{\sum_{i=1}^{|C|} tp_i}{\sum_{i=1}^{|C|} (tp_i + fp_i)} \tag{5.1}$$

$$hR = \frac{\sum_{i=1}^{|C|} tp_i}{\sum_{i=1}^{|C|} (tp_i + fn_i)} \tag{5.2}$$

$$hF_\beta = \frac{(\beta^2 + 1) \cdot hP \cdot hR}{\beta^2 \cdot hP + hR} \tag{5.3}$$

Equation 5.4 shows the Hamming loss (*hamm*) for one instance. We denote the true label vector of an instance as y, and the predicted vector as $\hat{y}$, with $y_i = 1$ if the instance is of class c_i, $y_i = 0$ otherwise. Hamming loss reports how many times on average, a class label is incorrectly predicted, i.e., the number of false positives and false negatives, normalized over total number of classes and total number of examples.

$$l_h(\hat{y}, y) = \sum_{i=1}^{|C|} 1_{\hat{y}_i \neq y_i} \tag{5.4}$$

$$hl_H(\hat{y}, y) = \sum_{i=1}^{|C|} 1_{\hat{y}_i \neq y_i} \max_{\{j | c_i \sqsubseteq c_j\}} 1_{\hat{y}_j = y_j} \tag{5.5}$$

Equation 5.5 shows the hierarchical loss (*h-loss*) (30) for one instance, which extends hamming loss to account for any existing underlying hierarchical structure of the labels. The idea of hierarchical loss is based on the notion that, whenever a classifier makes a mistake at any node in a given hierarchy, no further loss should be counted for any mistake in the subtree rooted at that particular node ignoring any subtree which is rooted at a wrong prediction node.

5.3 Problem Definition

In a knowledge graph, not every instance may come with proper type information. Untyped instances and instances with incomplete set of types are a common problem in Semantic Web knowledge bases (149), therefore, we need methods which can automatically predict types of instances. Thus, the task of type inference is to assign types to untyped instances, as well as adding types to instances with incomplete type information. To that end, available information about the instance, such as its relation to other instances, is used to train an inference model.

At the same time, not all information in the RDF data may be correct; there can be various types of errors (220), including wrong relations between instances (151; 153), wrong interlinks between datasets (147), and wrong literal values (211; 62), among others. Thus, when training a classifier for predicting missing types, those errors will shine up as *noise* in the respective training set, and require a noise-tolerant learning approach. We assume that the type hierarchy is correct, and we restrict the hierarchical structure to trees for simplicity and because DAGS are not supported by multilabel classification libraries.

In particular, types in RDF knowledge bases come are organized in hierarchies. Hence, we model the type prediction task as a hierarchical multilabel classification problem, which we define according to the categorization proposed in (179). The classification problem is defined as $< T, MPL, PD >$, which means that the type of graph representing the class hierarchy is a tree (T), instances are allowed to have multiple paths of labels (MPL), and that instances are allowed to have partial depth (PD) labeling (i.e., non-mandatory leaf node prediction).

The partial depth labeling is important in our problem since in many cases the class hierarchy is incomplete, requiring an instance which cannot be typed with a leaf node to be assigned a more general type. In Fig. 5.1, Arnold_Schwarzenegger is neither an AdultActor nor a VoiceActor, i.e., none of the specializing classes of Actor is appropriate. Thus, the instance should be typed as an Actor, which is a non-leaf node. Supporting multipath labels is also relevant because many instances might have multiple labels which are not in the same path in the hierarchy. In the same example, Arnold_Schwarzenegger is labeled with OfficeHolder, BodyBuilder, Actor, and their generalizations, thus having three paths in the hierarchy.

Although our problem definition does not support DAGs, they can be transformed into trees by selecting (e.g., at random, or by leveraging a priori distributions) a single parent for nodes with multiple parents. This simplifies the hierarchy, but leads to an information loss, which could in theory result in a drop in the quality of the predictions.

The extraction of features for the classifier is also an important part of the problem addressed in this chapter. Different datasets might have domain specific features highly valuable for the type prediction. The extraction of features from knowledge bases is a problem which deserves an exclusive study. Therefore we focus on graph and latent features which can be extracted from any knowledge graph.

5.4 Approach

The problem of type prediction in RDF data requires highly scalable approaches which can handle a high number of labels, features, and instances inherent to many Semantic Web datasets. We propose a more scalable version of a local classifier per node approach *SLCN*, which uses local feature selection (c.f. Appendix A).

In our approach, we assume that the knowledge base has a type hierarchy which is materialized in the dataset, i.e., if an instance is assigned a given type, it must also be assigned all its superclasses. If the hierarchy is not materialized, we perform simple reasoning to infer the assertions of all superclasses absent in the dataset by exploiting the `subClassOf` relations.

5.4.1 Algorithm

SLCN is based on the local classifier per node (LCN) with top-down prediction approach and *siblings* negative examples selection policy. This means that we train one binary classifier for every class $c_i \in C$, and each of those classifiers is trained on a local transformed dataset with a binary class label (belongs to the type: $y_c = 1$, or not: $y_c = 0$). The top-down prediction approach means that when predicting the types of a given instance, we first classify the instance for the types in the highest level.

When considering a non-mandatory leaf-node prediction problem, the class-prediction top-down approach has to use a stopping criterion that allows an example to be classified just up to a non-leaf class node. We follow the approach proposed by Wu et al. (215), where for all the types which the instance is predicted to belong to, the local classifiers of its subtypes predict if the instance belongs to any of its children, and so forth. Whenever the instance is predicted not to belong to a given type, then it is assumed that it does not belong to any of its subtypes either, therefore there is no need to run the local classifiers of the children nodes.

Assuming that a hierarchical multilabel classifier is perfect and correctly predicts all classes and we want to, for example, predict the types of the instance `Arnold_Schwarzenegger`. The classifier would first predict it is a `Person`. Then it would predict that it also belongs to its three subtypes `Artist`, `OfficeHolder` and `Athlete`. Following the `Artist` branch, it would then predict that it belongs to `Actor` and does not belong to `Painter`, and finally that it does not belong to either `AdultActor` or `VoiceActor`. Following the `Athlete` branch, the classifier would predict it belongs to `BodyBuilder`, and does not belong to `MotorsportRacer`. The local classifiers for the subtypes `MortorcycleRider` and `RacingDriver` would not need to make any prediction since the instance does not belong to their supertype `MotorsportRacer`, and therefore, in order to be consistent with the hierarchy, cannot belong to any of its children.

The top-down approach ensures that the outcome of the multilabel classifier is consistent with the type hierarchy. However, it can cause the *blocking problem* (187), which may occur during the top-down process of classifying a test example. The classifier at a certain level in the class hierarchy predicts that the example in question does not have the class associated with that classifier. In this case the classification of the example will be blocked, i.e., the example will not be passed to the descendants of that classifier. Sun et at. (187) propose methods for addressing this this problem (e.g., such as threshold reduction, restricted voting, and extended multiplicative thresholds). However, the results show that, although the proposed methods can reduce the blocking problem, they also cause a degradation in precision. Therefore, in our approach we decide not to use any of these approaches.

As scalability is an important factor in the problem studied in this chapter, we choose to use the *siblings* negative examples policy, which reduces the sizes of local training datasets for classes in the lower levels of the hierarchy. The local training sets are created including the instances belonging to the target class as positive examples and the instances belonging to its sibling classes as negative examples. For instance, the transformed dataset with *siblings* for the type `BodyBuilder` would contain as negative examples the instances belonging to its sibling class `MotorsportRacer` and, because we allow partial-depth prediction, the instances which belong to its superclass `Athlete` but none of its children.

Typically, the number of labels in the lower levels of the hierarchy is higher, and the lower the level of the label node, the smaller the subset is. Assuming that the label hierarchy has a fanout b, and the instances have a single path only, the average transformed dataset size would be $|D| * (b * log_b(|C|))/|C|$ instead of $|D|$. The average size of the transformed datasets also increases with the number of different paths instances have. However, for simplicity and because the average number of different paths per instance is low in most real datasets (c.f. average number of paths per instance (ANP) in Table 5.1), we ignore this factor when calculating the average size.

In the proposed approach, local feature selection, sampling, and class balancing are performed for every local classifier. The intuition is that in each binary subproblem, where for the instances of a given class we predict whether they belong to a subclass, not all the features might be relevant. Especially in cross-domain datasets, such as DBpedia, YAGO, and Wikidata, the set of features required to predict, for instance, if an `Athlete` is a `MotorsportRacer` is completely different from those required to predict if an `Infrastructure` is an `Airport`. This allows the local classifier to handle a smaller set of locally relevant features instead of a larger set with features relevant to all classes.

Moreover, we choose to use the *filter* instead of the *wrapper* feature selection method, where we calculate the information gain of each feature and select the top-k most relevant features ranked by information gain. Since the idea of local feature selection is to reduce the training time of the local classifiers, it makes more sense to perform the feature selection if its complexity is lower than that of the classifier training. Hence, we decide to use a simple feature selection method, whose complexity grows linearly with the number of features.

The benefit of local feature selection cannot be achieved using global multilabel classification methods, since a single model is learned on a single set of selected features, which has to be relevant for all the classes. Selecting features globally might lead to preferring features relevant to the most frequent classes, and potentially leaving out features which are relevant to less frequent classes. In Appendix A we conduct experiments which show that local feature selection performs consistently better than global feature selection.

For the local sampling, we set a maximum local training sample size n. The idea is that if a local classifier has a number of instances smaller than the maximum training sample size, no sampling is performed, so that the training set does not lose any valuable instances. On the other hand, local classifiers with a high number of instances, such as those for the classes in higher level of the hierarchy, will be trained on a smaller sample of size n, reducing the time required for training the local classifier. When sampling the data, potential class imbalance can be addressed individually for each class in its transformed dataset. For that, we define a bias to uniform class distribution $u \in [0, 1]$, where $u = 0$ means that the class distribution is left as it is, and $u = 1$ means that class weights are assigned values that result in an uniform class distribution. With that, the classifier settings can be defined by the triple $< k, n, u >$. In the experiments discussed in section 5.5.2, we evaluate the influence of each of these parameters on the performance of SLCN.

Algorithm 2 shows the pseudo code of SLCN indicating the main characteristics of the approach. The sampling and feature selection (SAMPLE_INSTANCES and SELECT_FEATURES) is performed for every class $c \in C$ and incorporated into the training of SLCN. In the algorithm X is the features matrix, where instances are represented by rows, and columns are features, Y is the labels matrix, also with instances represented as rows, a $|C|$ columns represent the types, h is a map containing all the nodes of the type hierarchy, *local_feats* is a map containing the sets of selected features for each class, and *local_clfs* a map of the local binary classifiers of each class.

The function PREDICT_RECURSION illustrates how the top-down prediction approach stopping criterion works, which we implement by keeping a set of indices i to the instances to which the local classifiers should be applied.

One limitation of SLCN is that it does not support disjointness between classes, since it assumes independence between sibling nodes. However, at the moment, most knowledge bases do not contain class disjointness axioms. Approaches which can model dependencies between classes, such as MLC4.5 and LCPN with ECC or LPW, should be able to handle such disjointness even if not explicitly defined in the ontology. When training the classifier, since we use the siblings negative examples selection policy, we implicitly use the closed world assumption in order to generate negative labels for the local classifiers. This can be a problem on datasets where the type assertions are highly incomplete (146).

Algorithm 2 SLCN pseudocode

```
$local\_clfs \leftarrow \emptyset$
$local\_feats \leftarrow \emptyset$

function TRAIN($X, Y, h$)
    for $c \in C$ do
        $n \leftarrow$ number of instances in $X$
        if HAS_PARENT(h[c]) then
            $i \leftarrow \{i|Y[i, h[c].parent] = 1\}$        ▷ Select instances that belong to parent
        else
            $i \leftarrow \{1, ..., n\}$
        $X_{local} \leftarrow X[i]$
        $Y_{local} \leftarrow Y[i, c]$
        $X_{local}, Y_{local} \leftarrow$ SAMPLE_INSTANCES($X_{local}, Y_{local}, n$)
        $local\_feats[c] \leftarrow$ SELECT_FEATURES($X_{local}, Y_{local}, k$)
        $X_{local} \leftarrow X_{local}[:, local\_feats[c]]$
        $local\_clfs[c] \leftarrow$ TRAIN_LOCAL_CLASSIFIER($X_{local}, Y_{local}$)

function PREDICT_RECURSION($X, Y, h, n, i$)
    $X_{local} \leftarrow X[:, local\_feats[n]]$
    $Y[i, n] \leftarrow$ PREDICT_LOCAL_CLASSIFIER($local\_clfs[n], X_{local}$)
    i $\leftarrow \{i|Y[:, n] = 1\}$
    for $c \in h[n].children$ do
        Y $\leftarrow$ PREDICT_RECURSION($X, Y, h, c, i$)
    return Y

function PREDICT($X, h$)
    $n \leftarrow$ number of instances in $X$
    $Y \leftarrow$ ZEROS($n, |C|$)        ▷ Initialize the matrix $Y$ with zeros (empty predictions)
    for $r \in$GET_ROOTS($h$) do
        $Y \leftarrow$ PREDICT_RECURSION($X, Y, h, r, \{1, ..., n\}$)
    return $Y$
```

5.4.2 Features

It is not possible to directly apply hierarchical multilabel classification methods on knowledge bases. In order to be able to work with classifiers, we need to represent every typed instance $x \in X$ in the knowledge base as a feature vector, and the types of the given instance as labels. We focus on general features which can be extracted from the relations between entities represented in a knowledge graph. Therefore, we do not investigate features extracted from external sources and text. According to Nickel et al. (137), two kinds of features can be obtained from knowledge graphs: latent features and graph features. The former consists of features which cannot be directly observed in the graph, often these are lower-dimensional representations (also known as embeddings) of entities. The latter consists of features that can be directly observed from the edges in the graph.

Latent features are dense lower dimensional representations of entities in an embedding space. The existent latent multirelational learning models, often used for the link prediction problem, generate this kind of entity representation. Since in these models every entity on the knowledge graph is represented by embeddings, we can use these embeddings as features and types as labels and train a multilabel classifier for type prediction. Assuming that in the latent feature representations, entities of same type are located close to each other in the embeddings space, these features should in principle be useful for the type prediction task.

Some of the state-of-the-art embedding models are TransE (15), TransR (111), RESCAL (139), multiway neural networks (mwNN) (51) and Holographic Embeddings (HolE) (138). In this chapter we choose to use Holographic Embeddings (HolE) (138), which learns compositional vector space representations of entire knowledge graphs. The proposed method is related to holographic models of associative memory in that it employs circular correlation to create compositional representations. HolE is efficient to compute, easy to train, and highly scalable, but at the same time it is highly expressive and can model complex relations. This achieved by using circular correlation as the compositional operator (c.f. Equation 5.6), which is illustrated by the 3-dimensional example for c in Equation 5.7. It is shown that holographic embeddings are able to outperform state-of-the-art methods, such as TransE and RESCAL, for link prediction in knowledge graphs and relational learning benchmark datasets.

Circular correlation, also known as compressed tensor product, enables relations to be modeled as d-dimensional vectors instead of $d \times d$ matrices like in RESCAL. With that the amount of memory for representing relations is reduced from $O(d^2)$ to $O(d)$ and the runtime for learning the model is reduced from $O(d^2)$ to $O(d \log d)$ (138).

$$c = a \star b = \sum_{0}^{d-1} a_i b_{(k+1) mod d} \tag{5.6}$$

$$\begin{aligned} c_0 &= a_0 b_0 + a_1 b_1 + a_2 b_2 \\ c_1 &= a_0 b_2 + a_1 b_0 + a_2 b_1 \\ c_2 &= a_0 b_1 + a_1 b_2 + a_2 b_0 \end{aligned} \tag{5.7}$$

An important characteristic of the circular correlation as an operator to model relations is that it is noncommutative, i.e., $a \star b \neq b \star a$, which enables it to model asymmetric relations. In HolE, the probability of a triple (s, p, o) is modeled as shown in Equation 5.8, where e_s and e_o are the embedding representations of the subject s and object o, and r_p is a vector that represents the relation p, and σ is the logistic function, that converts the triple score to a probability.

$$\sigma(r_p^\top (e_s \star e_o)) \tag{5.8}$$

In this chapter, we will call the set of entity embedding features E, where E has d dimensions. In general, d is small ($d \in \{5, 10, 25, 50, 150\}$), but the feature vectors are dense, i.e. they have few zeros.

Graph features can be directly extracted from the triples in the graph, and therefore are simpler to be obtained. In this work, we propose the extraction of *binary* features, following (152), which are not specific to a dataset at hand, but applicable on any general SW knowledge base. R_{out} is the set of outgoing relations, R_{in} is the set of ingoing relations, Q_{out} is the set of qualified relations, i.e., pairs of outgoing relations and object types, and Q_{in} is the set of ingoing qualified relations, i.e., pairs of ingoing relations and subject types. For convenience, we define the set of ingoing and outgoing relations as $R = R_{\text{in}} \cup R_{\text{out}}$ and the set of all qualified relations as $Q = Q_{\text{in}} \cup Q_{\text{out}}$.

The feature sets for the classification problem are extracted with the following SPARQL queries, where the keyword `a` is used as a shorthand notation for `rdf:type`:

R_{out} : select distinct ?x ?p where {?x ?p ?z, ?x a ?c}
R_{in} : select distinct ?x ?p where {?z ?p ?x, ?x a ?c}
Q_{out} : select distinct ?x ?p ?t where {?x ?p ?z, ?z a ?t, ?x a ?c}
Q_{in} : select distinct ?x ?p ?t where {?z ?p ?x, ?z a ?t, ?x a ?c}

In theory, on a knowledge base containing relations $p_i \in P$, and $c_i \in C$ types, R_{out} and R_{in} can have up to $|P|$ dimensions, and Q_{out} and Q_{in} up to $|P| \times |C|$ dimensions. In practice, however, R_{in} is often smaller than R_{out} because, for example, data properties, which have literals as objects, appear always as outgoing relations of entities and never as ingoing. Also, because of domains and range restrictions of relations, many combinations of outgoing relation and object type, as well as ingoing relation and subject type never occur, therefore the sizes of Q_{out} and Q_{in} often are significantly smaller than $|P| \times |C|$. It is important to note that, in contrast to the dense entity embeddings feature set E, the feature vectors Q and R are in practice highly sparse with few non-zero entries. These characteristics can be observed in Table 5.1, which give the size of these feature sets on the datasets used in our experiments.

In our experiments, we define the set of features used in a type prediction task as F, which may consist of any combination of the features sets R_{out}, R_{in}, Q_{out}, Q_{in} or E. While in SLCN it is possible to include dataset specific features, such as DBpedia categories or text features extracted from Wikipedia abstracts, in this work, we concentrate on general features which can be extracted from *any* SW knowledge base. It is worth mentioning that, in contrast to SDType (150) and other existing methods, which usually rely on a certain kind of features, the proposed hierarchical multilabel classification approaches can handle any kind of features which could be extracted from knowledge bases, and is thus more versatile.

In the future we plan to perform experiments with different kinds of features and propositionalization strategies (168), and evaluate how they affect the predictive performance. However, since in this chapter we focus on the prediction methods and not the feature extraction, we restrict ourselves to the features described previously.

In summary, the contributions of this work are the formulation of the type prediction as a hierarchical multilabel classification problem and the proposal of SLCN, a scalable hierarchical multilabel classifier based on the local classifier per node approach which exploits local feature selection and sampling in order to improve scalability and facilitate the use of such approach on large cross-domain LOD datasets. Moreover, we investigate how the use of different feature sets affect the performance of SLCN and other methods. We consider embedding features obtained with HolE, as well as ingoing and outgoing relations and qualified relations.

5.5 Experiments

The experiments are divided into five main parts. First, we evaluate the performance of different local classifiers for SLCN, and different parameter values for $< k, n, u >$. Second, we compare the influence of graph and latent features, showing that the latter only lead to a marginal performance improvement. Subsequently we compare SLCN to SDType and different state-of-the-art multilabel classifiers, analyzing performance and scalability with respect to the number of instances, features, and labels. We do not compare the proposed approach to RDFS reasoning, because it has already been shown to outperform reasoning in the case of real-world SW knowledge bases (150). Finally, in the last part, we make a comparison on different large-scale RDF datasets.

For our experiments, we use MULAN 1.5, which is an open-source Java library for learning from multilabel datasets based on WEKA (197). It includes a variety of state-of-the-art multilabel classification algorithms, and offers multilabel feature selection and evaluation. Apart from SDType, we compare SLCN to the local approaches HMC, HOMER, and the global approach MLC4.5 (37). HMC is an implementation of the LCPN approach. HOMER (195) is similar to HMC, but it uses balanced clustering to generate a hierarchy for flat labels, where the non-leaf nodes are meta-labels identifying label clusters. MLC4.5 and SDType were re-implemented in the MULAN framework. The performance of SDType is very sensitive to the chosen confidence threshold, and the optimal threshold may vary with the used dataset. Therefore, for our experiments, we added an extra step to the training phase of SDType in order to find the confidence threshold which maximizes the hF measure. Apart from that, all methods were used with their standard settings in MULAN.

5.5.1 Datasets

In our experiments, we use four different large scale cross-domain datasets: DBpedia, DBpedia with YAGO types, NELL, and Wikidata[1]. We also use AIFB and Mutagenesis datasets, which are two smaller and simpler domain-specific datasets.

[1] The datasets used are available for download at
http://dws.informatik.uni-mannheim.de/en/research/hmctp

They are especially needed for the latent features experiments, since the computation of entity embeddings for the large scale datasets can be expensive. Because MULAN can only handle trees, not arbitrary DAGs, we convert all DAG type hierarchies to trees by retaining only the subsumption relation of the most frequent parent node. Table 5.1 shows some statistics about the different datasets, including number of instances, percentage of instances with partial-depth (PD) and multipath (MPL) labels, average number of paths per instance (ANP),number of labels ($|C|$), and size of the different feature sets ($|R_{\text{out}}|$, $|R_{\text{in}}|$, $|Q_{\text{out}}|$, $|Q_{\text{in}}|$). In the next paragraphs we briefly discuss relevant characteristic of each dataset used.

Dataset	Instances	MPL	ANP	PD	$\|C\|$	$\|R_{\text{out}}\|$	$\|R_{\text{in}}\|$	$\|Q_{\text{out}}\|$	$\|Q_{\text{in}}\|$
DBpedia	4 218 125	0.02%	1.02	32.6%	476	1390	659	30 423	10 427
DBp(YAGO)	2 886 305	81.4%	3.16	86.2%	454	1308	638	61 595	45 484
NELL	29 317	38.9%	1.10	32.2%	264	248	248	2721	3056
Wikidata	19 254 100	63.4%	1.64	18.4%	474	1324	474	53 175	119 207
AIFB	27 100	48.3%	1.48	10.8%	58	81	81	287	538
Mutagenesis	14 157	0%	1.00	0%	87	4	4	118	14

Table 5.1: Statistics about the datasets used

DBpedia: We use DBpedia 2014[2] with mapping-based properties. There are two main issues with existing DBpedia type assignments. The first is that DBpedia only contains single path labels, although it is clear that several instances, such as Arnold_Schwarzenegger, should belong to multiple paths. This happens because of its extraction framework, which maps infoboxes to types and assigns an instance to the type of the first infobox of its Wikipedia page. That makes it an exception amongst other main RDF datasets which, as Table 5.2 illustrates, have a significant portion of its instances with multipath labels. The second problem is that the correct type can be trivially predicted from outgoing properties, as reported in (151), which happens because the DBpedia outgoing properties and types are generated in one step from the same original information. Therefore, in our experiments, we use only the feature sets R_{in} and Q_{in} for DBpedia. DBpedia 2014 has a class hierarchy which is not a tree only because of the class Library, which is a subclass of EducationalInstitution and Building. All the other classes have a single parent class. In order to convert it to a tree, we choose EducationalInstitution to be the only superclass, following the tree depicted by the DBpedia Ontology browser.[3] Since DBpedia types were originally materialized with the DAG hierarchy, after the transformation to a tree, all the 816 instances of Library (0.02% of the total) appear to have two paths in the tree.

DBpedia with YAGO types: Unlike DBpedia, YAGO[4] extracts its type hi-

[2] http://wiki.dbpedia.org/Downloads2014

[3] http://mappings.dbpedia.org/server/ontology/classes/

[4] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/

erarchy from Wikipedia categories. Because of that, it has a staggering 384 174 different types and a complex DAG type hierarchy. Moreover, YAGO has a very limited number of relations ($|R_{\text{out}}| = 37$) with most relations on the people and location domain), which results into a small number of features which are biased to specific classes. With the number of labels much greater than the number of features, the YAGO dataset as it is, is not well suited for the type prediction problem. Most of the DBpedia instances are linked to the YAGO types. Therefore it makes sense to combine both datasets by using DBpedia features and YAGO types as labels. With that, the problem of DBpedia's exclusively single-path labels, which are extracted together with the outgoing properties, can be ruled out. The problem of the high number of YAGO labels can be solved by simply choosing the top-k most frequent types. In our experiments, arbitrarily select the 474 most frequent YAGO labels.

Wikidata: Similarly to what was done to the YAGO types, in Wikidata, which also has a large original $|C| = 29099$, we arbitrarily select the 454 most frequent types. In contrary to the other knowledge bases used in the experiments, Wikidata does not rely on information extraction methods to generate its RDF graph. Wikidata is part of the Wikimedia community and its pages contain structured data, which can be exported to RDF format (56). Its type hierarchy is a DAG and, in order to transform it into a tree, for all types with multiple parents we keep the `rdfs:subClassOf` relation with the parent with greatest number of instances and delete the rest.

NELL: We use the NELL dataset (version 08m.690), which has originally 1 168 998 instances. However, the NELL's graph is highly sparse with only around 2% of instances being both typed and having at least one ingoing or outgoing relation. Therefore for the datasets used in our type prediction experiments we remove the other 98% of instances and use only the remaining 29317.

AIFB: The AIFB dataset[5] describes the AIFB research institute in terms of its staff, research group, and publications. The data is an export of the AIFB website and contains around 270 thousand triples. The type hierarchy is originally a wide and shallow tree with average fanout 14.25 and average depth 2.04.

Mutagenesis: The MUTAG dataset is distributed as an example dataset for the DL-Learner toolkit[6]. It contains information about 340 complex molecules that are potentially carcinogenic, which is given by the isMutagenic property. The molecules can be classified as "mutagenic" or "not mutagenic", and the main entity types atoms bonds and compounds which define the molecules. The type hierarchy is also originally a tree with average fanout 7.17 and average depth 2.49.

[5] http://www.aifb.kit.edu/web/Web_Science_und_Wissensmanagement/Portal

[6] http://dl-learner.org

5.5.2 SLCN Base Classifier and Parameter Settings

We conduct a first experiment to evaluate the performance of different types of local classifiers on our approach. Four different popular binary classifiers available in WEKA are evaluated. Table 5.2 reports the results of the comparison, which was performed on a random sample of the DBpedia data with YAGO types containing 28 863 instances (1% of the total) and features $F = R$. The results indicate that J48 (an implementation of the C4.5 decision tree algorithm) and LibSVM perform equally well in terms of prediction quality, with J48 being about eight times faster than SVM. Thus, we use J48 as a base classifier in the subsequent experiments.

classifier	rt(ms)	*h-loss*	*hamm*	hP	hR	hF
J48	111 711	**2.5060±0.2044**	**0.0069±0.0005**	**0.5136±0.0373**	**0.4961±0.0249**	**0.5029±0.0110**
NaiveBayes	**56 692**	2.8531±0.1119	0.0076±0.0002	0.4519±0.0125	0.4161±0.0178	0.4330±0.0110
AdaboostM1	104 238	2.6207±0.1384	0.0072±0.0003	0.4831±0.0215	0.4311±0.0172	0.4548±0.0028
LibSVM	880 441	**2.5080±0.2367**	**0.0069±0.0006**	**0.5165±0.0522**	0.4705±0.0313	0.4891±0.0101

Table 5.2: Comparison of different local classifiers on SLCN

In the experiments, we evaluate how the three parameters k, n and u (i.e., the number of features, the local training sample sizes, and the bias to uniform class distribution) affect the performance of SLCN. The evaluation is performed on the same sample described before, using J48 as local classifier and the default setting $k = 100$, $n = 500$. We then vary k and n measuring the runtime as well as hP, hR, hF, *h-loss* and *hamm*.

The plots in Figure 5.3 show hF and *runtime* for different parameter values. It is notable that for both the number of features k and maximum train set size n, the hF curves flatten after a certain point, while the runtime curves continue to grow. The optimal values for n and k depend on characteristics of the data, and may vary from dataset to dataset.

As the local classification problems can be rather skewed, we have also performed experiments with different sampling biases towards a more uniform class distribution in the local sampling. Since SLCN is based on LCN with *siblings* negative example selection, the classes are not as imbalanced in the local training sets as they are in the whole dataset. Moreover, we select the most frequent classes from Wikidata and YAGO, which excludes the smallest classes, and hence avoids the most skewed local classification problems. Therefore, the sampling bias to uniform class distribution does not significantly affect the performance of SLCN, i.e., we stick to stratified sampling in our experiments.

5.5.3 Graph Features vs. Latent Features

In this section, we perform a comparison between latent features and graph features for type prediction. In these experiments we use HolE features, which were learned on the whole datasets excluding the `rdf:type` relations. HolE provides

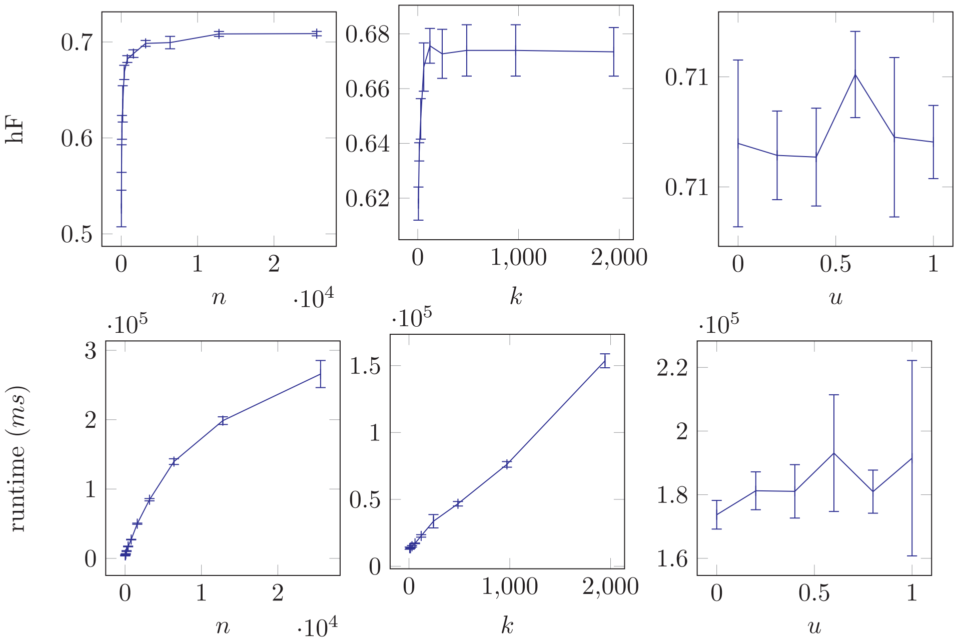


Figure 5.3: Evaluation of the impact of the parameters n and k on hF and runtime.

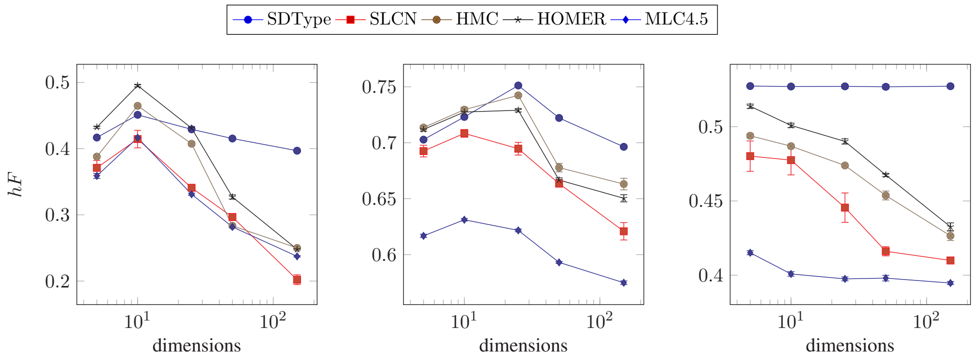
state-of-the-art performance with an expressive, yet simple and scalable model, which enables us to learn the model on larger datasets (138). We use the HolE implementation from the python library Scikit-KGE[7]. We learn holographic embeddings with the parameter settings recommended by the authors, and we vary the number of dimensions in order to find the optimal value on the smaller datasets AIFB and Mutagenesis.

Firstly, for the smaller datasets AIFB and Mutagenesis as well as NELL we learn HolE embeddings with number of dimensions $d \in \{5, 10, 25, 50, 150\}$, and evaluate how good the learned embeddings are as features for the type prediction task. Figure 5.4 shows how the hF is affected by the dimensionality of the embeddings, for the Mutagenesis dataset 10 was the optimal value, for AIFB it was 25, and for NELL it was 5. Since both datasets are fairly simple, we can observe that for the higher number of dimensions, the model overfits. This, however, should not be a problem for the other datasets, which are much more complex and contain more relations.

We evaluate the performance of hierarchical multilabel classifiers for type prediction with five different feature sets: qualified ingoing and outgoing relations (Q), ingoing and outgoing relations (R), HolE embeddings only (E), combination of R and HolE embeddings ($R \cup E$), and combination of Q and HolE embeddings ($Q \cup E$).

The objective is to evaluate the relevance of entity embeddings to the type

[7] `https://github.com/mnick/scikit-kge`



(a) Mutagenesis (b) AIFB (c) NELL

Figure 5.4: Type prediction results for different number of HolE embeddings dimensions

prediction task, and how they compare to other traditional graph features, as well as to examine if any improvement can be obtained by combining these two kinds of features. Since SDType cannot handle the real valued numerical features from E, for this type prediction method we discretize the numerical attributes into 25 bins using the equal frequencies approach, resulting in $25d$ binary features.

Dataset	Method	Q	R	E	$R \cup E$	$Q \cup E$
AIFB	HMC	**0.9870±0.0008**	0.9495±0.0005	0.7424±0.0010	0.9437±0.0019	**0.9867±0.0007**
	MLC4.5	**0.9880±0.0011**	0.9497±0.0007	0.6359±0.0028	0.9150±0.0017	0.9765±0.0016
	SLCN	**0.9764±0.0043**	0.9380±0.0032	0.7213±0.0057	0.9223±0.0023	**0.9764±0.0038**
	HOMER	**0.9872±0.0007**	0.9493±0.0003	0.7291±0.0021	0.9366±0.0016	**0.9868±0.0007**
	SDType	**0.8032±0.0040**	0.7373±0.0040	0.7512±0.0014	0.7706±0.0006	0.7844±0.0007
Mutagenesis	HMC	**0.9343±0.0020**	0.7607±0.0034	0.4647±0.0041	0.7605±0.0032	**0.9344±0.0022**
	MLC4.5	**0.9050±0.0026**	0.6796±0.0024	0.4165±0.0057	0.5998±0.0026	0.8897±0.0033
	SLCN	**0.9262±0.0053**	0.7622±0.0078	0.4145±0.0319	0.7582±0.0083	**0.9262±0.0053**
	HOMER	**0.9339±0.0020**	0.7778±0.0043	0.4953±0.0038	0.7777±0.0042	**0.9340±0.0022**
	SDType	**0.7372±0.0016**	0.7261±0.0056	0.4511±0.0030	0.4608±0.0031	0.5265±0.0023
NELL	HMC	0.8695±0.0019	**0.9165±0.0041**	0.4869±0.0023	**0.9186±0.0017**	0.8695±0.0019
	MLC4.5	0.9119±0.0016	**0.9548±0.0014**	0.4008±0.0038	0.8682±0.0019	0.7780±0.0040
	SLCN	0.7608±0.0041	**0.8956±0.0042**	0.4794±0.0065	0.8842±0.0025	0.7365±0.0076
	HOMER	0.8678±0.0155	**0.9251±0.0036**	0.5009±0.0030	**0.9250±0.0009**	0.8674±0.0155
	SDType	0.8922±0.0025	**0.9025±0.0026**	0.5271±0.0019	0.5588±0.0035	0.6093±0.0026

Table 5.3: Comparison of hF for type prediction on different feature sets

The results from Table 5.3 indicate that the best set of features is the qualified relations Q, which over all classifiers significantly improves the performance over the set of ingoing and outgoing relations R. The exception for that is NELL, where Q performs worse than R. This happens because the knowledge graph is highly incomplete, where several entities have no types or no ingoing or outgoing properties. For the classification dataset we select only the entities which have at least one type and at least one ingoing or outgoing relation, however, when extracting the features Q some of the objects of outgoing and subjects of ingoing relations are not typed entities, therefore we cannot use them as features in Q.

The latent features E, when used alone perform significantly worse than all other feature sets, indicating that this kind of features is not very relevant for the type prediction task. We also combine the embeddings E with R, in order to evaluate if the embeddings can add relevant information and improve performance. However, the experiments indicate that in some cases it does not significantly affect the hF measure, while in others it actually acts as noise, reducing the quality of the predictions.

Although the HolE embeddings have been shown to be useful in the link prediction problem, in the type prediction problem they do not seem to be of significant relevance when comparing to the graph features for the classifiers considered in our experiments. This can be attributed to the fact that these entity embeddings are learned with the objective of modeling links between entities, and not to separate them by types. Admittedly, the datasets used in these experiments are rather small and time constraints did not allow us to explore further approaches to exploit the embeddings. Further work applying deep learning methods and performing experiments on larger datasets would be better evaluate the relevance of embeddings in type prediction.

5.5.4 Scalability Experiments

In this section, we compare the scalability of the methods in terms of the number of instances, number of features, and number of labels of a dataset. The experiments were conducted on the same sample of DBpedia with YAGO types described in the previous section. To vary the number of instances, we randomly sample instances as training set and progressively increase the sample size, for the number of features we select features with highest information gain first, and for the number of labels we select the most frequent labels first.

Figure 5.5 shows the runtime and hF of each method for different number of instances, number of features and number of labels. SDType is the most scalable of the compared methods, however, its hF was significantly lower than all the other compared methods. The runtime of SLCN is close to that of SDType, improving the runtime in comparison to the other hierarchical multilabel classifiers, and improving hF in comparison to SDType. MLC4.5 has the best overall hF, however, in terms of runtime, it does not scale as well as SDType and SLCN. It is particularly noteworthy that the runtime of SDType and SLCN is significantly more scalable than the other approaches in terms of number of instances, features, and labels.

Although in the plots SDType and SLCN seems not to change its runtime with the number of instances, features and labels, their runtime is of course also affected. However, in comparison with the other methods, the increase in runtime is much smaller and cannot be visualized in the plot. The runtime of SLCN would grow

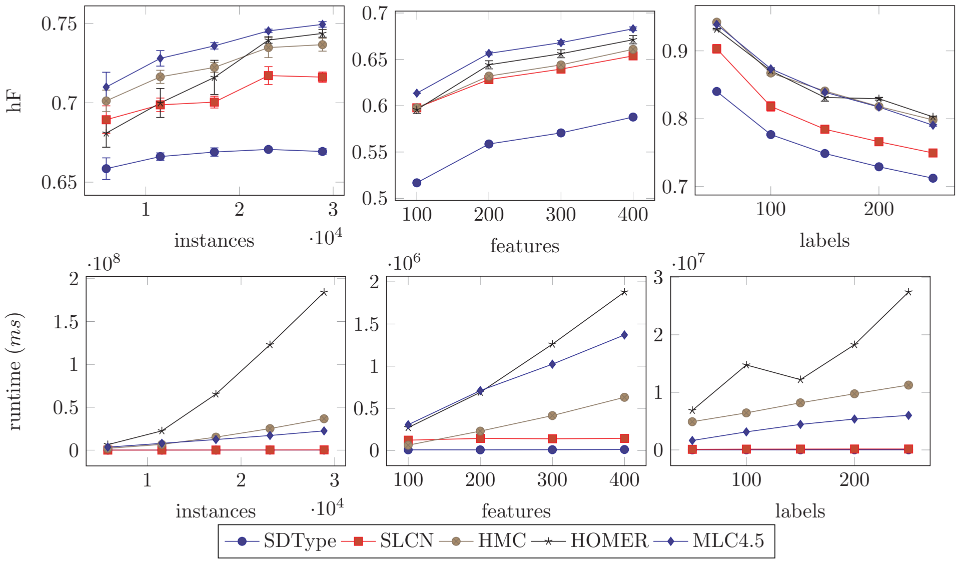


Figure 5.5: Scalability in terms of number of instances, features and labels

similarly to its local classifier for number features smaller than k and instances smaller than n. For larger values, the number of instances and features of the local datasets the would remain constant, and the increase in runtime would be determined by the instance sampling and feature selection methods used.

5.5.5 Large-Scale Experiments on SW Datasets

In this section, we perform large-scale experiments on whole RDF datasets. Table 5.4 shows the results of 5-fold cross validation on the RDF datasets presented earlier. Because of time limitation, we do not report the results for classifiers which require more than a week for training. HMC, HOMER and MLC4.5 were able to finish only on NELL, therefore for the other datasets in Table 5.4 we report the results only for SDType and SLCN, which were able to finish for all datasets, showing the effectiveness of the proposed approach in improving scalability.

On the NELL dataset, the HMC, HOMER and MLC4.5 perform better than SDType and SLCN. However, the runtime of the first three methods are notably longer than the others. When comparing SLCN against SDType, the former performs consistently better with respect to all evaluation measures, but longer runtime. Note that the results of SDType differ from those reported in (150) because the latter includes `owl:Thing` and classes in other ontologies, such as FOAF and schema.org, in the evaluation, while we exclude them. On all the other datasets, which are significantly larger than NELL (c.f. Table 5.1), SLCN is the best overall performer as HMC, HOMER and MLC4.5 were not able to finish in less than one week.

The use of qualified relation features (Q_{out} and Q_{in}) substantially increases the

Dataset	Method	hF	$h\text{-}loss$	$hamm$	rt(ms)
DBpedia $F = R_{\text{in}}$	SDType	0.7647±0.0021	0.7729±0.008	0.0027±0.0000	16 080 553
	SLCN	**0.8470±0.0009**	**0.4632±0.0051**	**0.0016±0.0000**	**7 024 255**
DBpedia $F = R_{\text{in}} \cup Q_{\text{in}}$	SDType	0.7702±0.0002	0.7501±0.0014	0.0026±0.0000	54 511 659
	SLCN	**0.8462±0.0005**	**0.4610±0.0048**	**0.0016±0.0000**	**10 154 987**
DBp(YAGO) $F = R_{\text{out}} \cup R_{\text{in}}$	SDType	0.6663±0.0001	2.6724±0.0019	0.0159±0.0000	6 744 282
	SLCN	**0.7029±0.0066**	**2.0965±0.0891**	**0.0133±0.0007**	**7 635 499**
DBp(YAGO) $F = R_{\text{out}} \cup R_{\text{in}} \cup Q_{\text{out}} \cup Q_{\text{in}}$	SDType	0.6705±0.0002	2.6477±0.0019	0.0157±0.0000	213 904 335
	SLCN	**0.7022±0.0064**	**2.1057±0.0903**	**0.0134±0.0007**	**48 374 257**
Wikidata $F = R_{\text{out}} \cup R_{\text{in}}$	SDType	0.7529±0.0001	0.5749±0.0002	0.0017±0.0000	208 957 224
	SLCN	**0.8116±0.0106**	**0.3748±0.0091**	**0.0014±0.0000**	**44 807 901**
Wikidata $F = R_{\text{out}} \cup R_{\text{in}} \cup Q_{\text{out}} \cup Q_{\text{in}}$	SDType	0.7759±0.0001	0.5189±0.0003	0.0016±0.0000	272 206 437
	SLCN	**0.8680±0.0028**	**0.2712±0.0060**	**0.0011±0.0000**	**64 413 619**
NELL $F = R_{\text{out}} \cup R_{\text{in}}$	SDType	0.9025±0.0026	0.4842±0.0129	0.0041±0.0001	**1 917 946**
	SLCN	0.8956±0.0042	0.5851±0.0148	0.0045±0.0002	2 547 871
	HMC	0.9165±0.0041	0.5014±0.0132	0.0036±0.0002	6 336 452
	HOMER	0.9251±0.0036	0.4799±0.0136	0.0032±0.0001	10 957 195
	MLC4.5	**0.9548±0.0014**	**0.3305±0.0091**	**0.0020±0.0001**	16 991 440
NELL $F = Q_{\text{out}} \cup Q_{\text{in}}$	SDType	0.8922±0.0025	**0.4447±0.0091**	0.0045±0.0001	**9 289 373**
	SLCN	0.7608±0.0041	1.0809±0.0175	0.0101±0.0002	18 252 489
	HMC	0.8695±0.0019	0.6859±0.0767	0.0055±0.0001	102 815 535
	HOMER	0.8678±0.0155	0.6866±0.1168	0.0056±0.0008	156 444 313
	MLC4.5	**0.9119±0.0016**	0.4840±0.0044	**0.0037±0.0001**	166 477 106

Table 5.4: Evaluation of different classification methods on large cross-domain SW datasets

dimensionality of the feature space, as it can observed in Table 5.1, and therefore the runtime is also increased. SDType is able to improve its results when considering the greater set of features for DBpedia and DBpedia with YAGO types, but SLCN actually yields slightly worse results. This may be because of a possibly higher level of dependency between the features in Q_{out} and Q_{in}. Since the filter feature selection method does not take dependencies between features into account, the selected feature set could contain several redundant features.

The results in Table 5.5 can illustrate the importance of the high scalability of SLCN. Training SLCN on AIFB with $F = Q$ is faster than training HOMER or HMC on $F = R$, and the prediction quality is significantly higher. On Mutagenesis, the training time is a bit slower, but again, the prediction quality is significantly higher. That shows that, when time and computing resources are limited, using SLCN allows you to work on a higher number of features in comparison to less scalable methods, and ultimately achieve better results.

Dataset	Method	hF	$h\text{-}loss$	$hamm$	rt(ms)
AIFB $F = R$	SDType	0.7373±0.0040	0.8844±0.0062	0.0164±0.0001	**80 892**
	SLCN	0.9380±0.0032	0.2296±0.0064	0.0046±0.0002	88 997
	HMC	**0.9495±0.0005**	**0.1975±0.0044**	**0.0038±0.0000**	442 235
	HOMER	**0.9493±0.0003**	0.1983±0.0034	**0.0038±0.0000**	387 434
	MLC4.5	0.9497±0.0007	0.1988±0.0059	**0.0038±0.0001**	389 573
AIFB $F = Q$	SDType	0.8032±0.0040	0.6902±0.0065	0.0128±0.0001	**207 402**
	SLCN	0.9764±0.0043	0.0675±0.0080	0.0018±0.0003	399 530
	HMC	**0.9870±0.0008**	**0.0360±0.0028**	**0.0010±0.0001**	1 609 020
	HOMER	**0.9872±0.0007**	0.0356±0.0027	**0.0010±0.0001**	3 739 919
	MLC4.5	0.9880±0.0011	0.0337±0.0028	**0.0009±0.0001**	1 484 945
Mutagenesis $F = R$	SDType	0.7261±0.0056	1.2770±0.0324	0.0164±0.0005	**56 511**
	SLCN	0.7621±0.0077	0.8650±0.0064	0.0118±0.0002	1679
	HMC	**0.7607±0.0034**	**0.8541±0.0060**	**0.0118±0.0002**	3505
	HOMER	**0.7778±0.0043**	0.8805±0.0122	**0.0121±0.0003**	29 643
	MLC4.5	0.6726±0.0023	1.3274±0.0076	**0.0177±0.0001**	55 289
Mutagenesis $F = Q$	SDType	0.7372±0.0016	0.9357±0.0041	0.0135±0.0001	**71 213**
	SLCN	0.9262±0.0053	0.2835±0.0121	0.0040±0.0003	10 304
	HMC	**0.9343±0.0020**	**0.2540±0.0032**	**0.0036±0.0001**	29 171
	HOMER	**0.9339±0.0020**	0.2557±0.0033	**0.0036±0.0001**	109 943
	MLC4.5	0.8991±0.0037	0.3728±0.0135	**0.0055±0.0002**	67 920

Table 5.5: Evaluation of different classification methods on smaller SW datasets

5.6 Conclusion

In this chapter, we have modeled the type prediction problem in Semantic Web knowledge bases as a hierarchical multilabel classification problem. We propose SLCN, and compare it to both popular hierarchical multilabel classifiers and the state-of-the-art type prediction approach SDType (which is currently one of the strongest and best scalable algorithms for the task at hand). The experiments indicate that the local feature selection and local sampling can significantly improve scalability without sacrificing the quality of the prediction. The results also show that SLCN can perform better than SDType, while scaling better than the other multilabel classifiers evaluated in this chapter. With enough computing power available, a state-of-the-art hierarchical multilabel classifier, such MLC4.5, is the best choice, while SLCN is a good trade-off when scalability is a major issue. The use of entity embeddings as features in our proposed type prediction approach and the results indicate that, under the experimental settings used in this chapter, the simple graph-features yield significantly better results.

Chapter 6

Detection of Relation Assertion Errors

6.1 Introduction

Many of the knowledge graphs published as Linked Open Data have been created from semi-structured or unstructured sources. The magnitude of many of these knowledge graphs, e.g.: DBpedia, NELL, Wikidata, YAGO, does not allow for manual curation, and, instead, require the use of heuristics. Such heuristics, however, do not guarantee that the resulting graphs are free from errors. Wikipedia, which serves as source for DBpedia and YAGO, is estimated to have 2.8% of its statements wrong (209), which add up to the error caused by the extraction heuristics. Therefore, automatic approaches to automatically detect wrong statements are an important tool for the improvement of knowledge graph quality.

Incompleteness is another major problem of most knowledge graphs. Automatic knowledge graph completion has been widely researched (137), with a variety of methods proposed, including embedding models. Although such methods can also be trivially employed for error detection, their performance has not yet been extensively evaluated on the task.

Many existing large-scale error detection methods rely exclusively on the types of subject and object of a relation (48; 151; 153), and try to spot violations of the underlying ontology and/or typical usage patterns. While types can be a valuable feature, some knowledge graphs lack this kind of information, have only incomplete type information, or have types which are not very informative. Moreover, some errors might contain wrong instances of correct types. For example, if someone adds the fact `playedFor`(`Ronaldo`, `Manchester_United`), which would be wrong because `Ronaldo` refers to Ronaldo Nazário instead of Cristiano Ronaldo, such an approach would not be able to detect the error.

In knowledge graph completion, paths in the graph have been proven to be valuable features (98; 69). For instance, in order to predict whether a person a lives in a place b (`livesIn`(a,b)), one important path feature is whether the person has a spouse who lives in b (`spouse`(a,X) $\rightarrow$ `livesIn`(X,b)), or whether the person has some child who was born in b (`childOf`(X,a) $\rightarrow$ `bornIn`(X,b)). Generalizing it for any pair of entities in a given relation, we can simply consider the previous example as path features `spouse` $\rightarrow$ `livesIn` and `childOf`$^{-1}$ $\rightarrow$ `bornIn`, with binary values indicating if the entities pair can be connected through each of the paths. For error detection, these features can complement the type features. However, searching for interesting paths for all the relations in a knowledge graph can be a challenging task, especially in datasets with many relations.

In this chapter, we propose a hybrid approach called *PaTyBRED* (Paths and Types with Binary Relevance for Error Detection), a method for the detection of relation assertion errors in knowledge graphs, which incorporates type and path features into local relation classifiers. Furthermore, we propose heuristic measures for the exploration of the paths search space. We perform an extensive comparison of our approach with state-of-the-art error detection and knowledge completion methods, and we conduct a manual evaluation of our approach on DBpedia and NELL.

6.2 Problem Definition

The problem addressed in this chapter is the detection of erroneous relation assertions in knowledge graphs. A dataset containing errors is given, and the facts should be ranked by their likelihood of being wrong.

It is important to note that we consider only features which can be extracted from the links between entities (`owl:ObjectProperty` relation assertions) and types (`rdf:type` assertions). To make the approach as versatile and applicable to as many knowledge graphs as possible, we do not use any other information, such as textual or numerical literals, or external knowledge sources. The problem can be defined as relation assertions error detection on internal features according to (149).

6.3 Approach

Our proposed approach is inspired by the Path Ranking Algorithm (PRA) (98) and SDValidate (151). It consists of a binary classifier for every relation which predicts the existence of a given pair of subject and object in the target relation. The set of classifiers can be thought of as a single multilabel classifier with binary relevance (i.e., each relation that can hold between a pair of instances is a label), where one binary classifier is learned for each class separately, and local feature selection (122) (c.f. Appendix A), with different classifiers being able to work on different sets of specialized features.

We use two kinds of features. The first one are the types of subject and objects. This kind of information has been successfully used for error detection in

SDValidate (151). By analyzing the types of subject and object in one given relation, one can easily spot a very common kind of error without relying on the domain and range restrictions, which are often inexistent or too general. For example, in DBpedia the triple `recordedIn(I'm_a_Loser, Abbey_Road)` is wrong. `I'm_a_Loser` is a song by The Beatles from the album `Abbey_Road` and the relation `recordedIn` has domain `MusicalWork` and range `PopulatedPlace`. A song being recorded in an Album is a clearly wrong fact. At the same time, if the object were `Abbey_Road_Studio` of the type `Recording_Studio`, which is not a subclass of `PopulatedPlace`, the fact would also be wrong according to a method relying solely on types. If there are many facts where songs are recorded in recording studios, statistical methods such as SDValidate would be able to identify that such a pattern is common, and therefore unlikely to be wrong, despite the violation of range restriction, while a song recorded in album is uncommon, therefore likely to be an error. Hence, statistical approaches such as SDValidate respect the actual usage of the ontology, rather than its axiomatic design. Recent works have been proposed that pinpoint such mismatches automatically (148). Moreover, type assertions might be absent or too general, resulting in no relevant information.

The main problem with this kind of approach is that it solely relies on type features. That means such approaches do not work on knowledge graphs with no type assertions, and may have poor performance on datasets with a shallow type hierarchy with non informative types or with incomplete type assertions. Moreover, by solely using type features, it is impossible to detect wrong facts with wrong entities of correct types, for instance, when a person instance is confused with another of same or similar name.

Alternatively we can use path features similar to those of PRA. However, solely relying on path features also has its problems. One of them is that correct facts may be labeled as error because of incompleteness. For instance, if river instances have the properties `country` (i.e., the countries a river passes through, typically multi-valued), and `mouthCountry` (i.e., the country where the river's mouth is, typically single-valued), then the feature `country` will be relevant for the relation `mouthCountry` since the confidence of the rule `mouthCountry`$(X,Y) \Rightarrow$ `country`(X,Y) is close to 1. However, some rivers do not have any assertions for `country` because of incompleteness, thus their correct `mouthCountry` assertion is predicted to be wrong. That can lead to propagation of incompleteness.

Another problem is that since `country` is a more relevant feature to predict the relation `mouthCountry` than vice versa, since the latter is far less common than the former. Hence, if an error occurs in the assertion of `country` for a river, it might happen that a correct `mouthCountry` assertion ends up being more likely to be detected as an error than the wrong `country` assertion. In order to make our approach more robust, we combine both type and path features.

Finding the relevant paths for each relation can be a challenging task. Since several paths may be relevant to different relations, we compute all possible paths up to a given length, and for every relation's local classifier we perform local feature selection. The number of possible paths grows exponentially with the number of relations, therefore an exhaustive search can easily become unfeasible. It is then crucial to have heuristics to efficiently navigate the search space. In the following subsection we propose and discuss such heuristic measures.

6.3.1 Extracted Features

Our method includes the following parameters that define the path selection: maximum path length, maximum number of paths per length, and path selection heuristics. Following the approach described in (98), we use the domain and range restrictions of relations for pruning uninteresting paths, and we do not allow a relation to be immediately followed by its inverse. If the number of possible paths of a certain length exceeds the maximum number of paths per length, we apply our path selection heuristics to prune the least interesting paths and comply with the specified paths upper limit.

We define a knowledge graph $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is the T-box and $\mathcal{A}$ is the A-box containing relations assertions $\mathcal{A}_R$ and type assertions $\mathcal{A}_C$. We define N_C as the set of concepts (types), N_R as the set of relations and N_I as the set of individuals (entities which occur as subject or object in relations). The set of relation assertions is defined as $\mathcal{A}_R = \{r(s,o)|r \in N_R \wedge s, o \in N_I\}$ and the set of type assertion as $\mathcal{A}_C = \{C(s)|C \in N_C \wedge s \in N_I\}$.

We define a path P as a sequence of relations $r_1 \rightarrow ... \rightarrow r_i \rightarrow ... \rightarrow r_n$. The sequence of relations is connected by a chain of variables, with $P(s,o)$ meaning s and o can be connected by a path $P(s,o) \iff r_1(s,x_1) \wedge ... \wedge r_i(x_{i-1},x_i) \wedge ... \wedge r_n(x_{n-1},o)$. The inverse of a relation r is denoted as r^{-1} where $r^{-1}(s,o) = r(o,s)$ can also be part of paths. A path of length one $P = (r)$ is equivalent to the relation itself $P(s,o) \equiv r(s,o)$. The length of a path is denoted as $|P|$. We define the set of subjects of P as $s_P = \{s|P(s,o)\}$ and set of objects as $o_P = \{o|P(s,o)\}$.

Relations and paths can be represented as adjacency matrices of size $|N_I| \times |N_I|$.The adjacency matrix of P can be computed by the dot product of its relations. However, computing the dot product of adjacency matrices can be an expensive operation, especially in large-scale knowledge graphs with millions of entities and high number of relations. Therefore, we need heuristic measures to explore the search space and compute the dot product only for the most relevant paths.

Let A and B be adjacency matrices – which can refer to a single relation or a path – which we want to concatenate in order to form a new path $A \cdot B$. We want a heuristic measure which can estimate the relevance of the path $A \cdot B$ without having to perform a potentially expensive matrix multiplication to compute its adjacency matrix. Since the paths computed are to be used by all relations, the proposed heuristic measures should not be computed with respect to a target relation, but only consider the matrices A and B.

Paths with empty adjacency matrices ($|A \cdot B| = 0$) are useless and should be pruned. A simple way to safely prune them is to calculate $o_A \cap s_B$. The set of objects o_A contains the columns of A which have non-zero elements, and the set of subjects s_B contains the rows of B which have non-zero elements. If the intersection is empty, then we know that $|A \cdot B| = 0$. Note that $|s_B| \leq |B|$ and $|o_A| \leq |A|$, and the intersection is cheaper to compute than dot product, therefore the runtime for computing $o_A \cap s_B$ is shorter.

For our proposed heuristic measures, we assume that paths with denser adjacency matrices are more likely to be more relevant features. Since the size of the intersection $o_A \cap s_B$ can be a good indicator of the number of nonzero elements in $A \cdot B$, we use it to define three measures for estimating the relevance of a path $A \times B$: We employ that characteristic into three proposed relevance measures $inter$, $m1$ and $m2$ (c.f Equations 6.1, 6.2 and 6.3).

$$inter(A, B) = |o_A \cap s_B| \tag{6.1}$$

$$m_1(A, B) = \frac{|o_A \cap s_B|}{|s_A \cap o_B| + 1} \tag{6.2}$$

$$m_2(A, B) = |o_A \cap s_B| \times |s_A \cup o_B| \tag{6.3}$$

By early pruning irrelevant paths, time is saved not only by computing fewer adjacency matrices, but also the number of features to be considered is reduced (fewer columns in the features table to be populated and less features to have the relevance computed).

Once the relevant paths have been selected, we compute their adjacency matrices and use them to populate the features used to train the relation classifiers. One of the problems of computing the whole adjacency matrix of paths is that some can be very dense and require a lot of memory. For example, the path `birthPlace` $\rightarrow$ `locatedIn`$^{-1}$ on DBpedia, which represents everything which is located in a place where someone was born in. Its adjacency matrix contains around 100 million non-zero elements and consumes more than 1GB of memory. As it is unlikely that all the entries in the matrix will be used, it would be desirable to handle such cases in a more efficient manner in order to restrict the memory consumption and speed up the paths adjacency matrices computation process.

It is worth pointing that the `rdf:type` relation is not considered in the paths. They are treated separately and are used to generate the type features, which consist of the set of asserted and subsumed types of an instance (we materialize the subsumed types into the assertions and ignore the subsumption relations). Integrating types into the paths can be problematic. Firstly it would significantly increase the search space. Secondly, a path which begins with the `rdf:type`, can only continue with `rdf:type`$^{-1}$ because types can only be objects in this relation (if we do not consider OWL class axioms in paths), and as mentioned earlier, we do not allow a relation to be immediately followed by its inverse.

6.3.2 Learning the Model

Once the paths have been selected, and their adjacency matrices have been computed, we can use them together with types as features to predict the existence of an entity pair (s, o) in a relation. The first step is to build a training dataset containing all extracted features for each relation r. We use as positive examples the entity pairs $D_{\text{pos}} = \{(s, o)|r(s, o)\}$, i.e. all the non-zero cells in the relation's adjacency matrix. Following (15), we generate negative instances $D_{\text{neg}} = \{\gamma(s, o)|(s, o) \in D_{\text{pos}} \wedge \gamma(s, o) \notin D_{\text{pos}}\}$ for supervised training by corrupting entity pairs with γ, which substitute the subject or the object for a random entity instance and ensuring the new pair is not positive. In a preliminary experiment, we compared this approach with that of (98), which is more expensive, and no significant difference in performance was observed.

As labels we use information from r indicating the existence of (s, o) in the relation. We extract path features from $\mathcal{A}_R$ and type features from $\mathcal{A}_C$. The path features are boolean values indicating whether a path connects s to o $(P(s, o)|\forall P \in \mathcal{P} - (r))$. The type features consist of the types of s and o (including subsumed types), i.e. $\{C|C(s)\}$ and $\{C|C(o)\}$. Other possible path feature is the existence of a path starting or ending in s and p $(P(s, X),\ P(X, s),\ P(o, X),\ P(X, o))$ as proposed in SFE (69), however the authors found out that this kind of feature does not improve performance. Our experimental results confirmed their results, therefore we do not consider this kind of feature in our approach.

Before we learn the local classifiers, we evaluate the relevance of the features. Since different features might be relevant for different relations, we perform feature selection separately for every relation. This allows the relation classifiers to work on a small set of locally relevant features, and, at the same time, removes irrelevant features which might act as noise and reduce the classifier's performance (122). We use the filter method, which simply select the top-k most relevant features, with χ^2 as relevance measure.

When comparing PaTyBRED with PRA and SFE, our approach has the following advantages. We try different popular classifiers to learn the relations, and we found that logistic regression, which is used in PRA and SFE, is not the best performer. We introduce a local feature selection step prior to training the relation classifiers. We propose heuristic measures to explore the paths search space. Moreover, negative evidence features, i.e. paths which connect negative but no positive entity pairs of a relation, are also considered. Since our approach is supervised and includes negative examples in the training data, this kind of features are extremely important to identify wrong facts.

6.4 Experiments

In our experiments, we evaluate the impact of different parameter settings in our approach, and compare it with SDValidate and state-of-the-art knowledge graph completion methods. We use ProjE[1] as well as the TransE and HolE implementations of scikit-kge[2]. The implementation of PaTyBRED is available on Github[3]. We do not directly compare our method with SFE, but we evaluate our approach with path features only (PaBRED), which perform at least as well as SFE.

The reported results from the embedding methods were obtained by not considering the type assertions. We tried adding the type assertions as an extra relation, however, this did not improve the results. The embedding methods suffer from the problem that the distribution of scores over different relations is not uniform. Often some relations have average triple scores lower than others, and this can result in a bias when detecting errors.

In order to reduce this problem, we run isolation forest to detect score outliers of each relation separately, and we use the outlier confidence values instead of the triple scores to rank the facts. Since unusually high scores are also outliers and we are interested only in the outliers of low scores, we do not consider as outlier any fact with score greater than the relation's average.

6.4.1 Datasets

In our experiments we use a variety of knowledge graphs, some of which are clean, and others noisy. In the first part of our experiments we automatically evaluate the performance of the error detection algorithms. In order to make the evaluation automatic, we use a variety of datasets to which we add synthesized wrong facts. We generate the erroneous facts by corrupting the subject or object of true facts, i.e., replacing the original entity with a randomly selected which results in a fact which does not exist in the original data. For our generation process, we add 1% of noise, and we generate two kinds of errors. In the first, we corrupt the triple by selecting any of the entities from the knowledge graph (independent of type), and in the second, we select only triples which have the same types as the original entity. That means the errors of the second kind are, in principle, more difficult to be detected than those of the first kind, since the new entity is more likely to have characteristics similar to those of the original one.

The datasets used are the following: As input knowledge graphs, we use DBpedia (2015-10), and NELL (08m-690). We use the following smaller domain specific datasets: Semantic Bible[4] AIFB portal[5] and Nobel Prize[6]. We also select four of the largest conference datasets from the Semantic Web dog food corpus[7], i.e., LREC2008, WWW2012, ISWC2013, and ESWC2015. WN18 and FB15k (Word-

[1] https://github.com/nddsg/ProjE

[2] https://github.com/mnick/scikit-kge

[3] https://github.com/aolimelo/kged

[4] http://www.semanticbible.com/

[5] http://www.aifb.kit.edu/web/Web_Science_und_Wissensmanagement/Portal

[6] http://www.nobelprize.org/nobel_organizations/nobelmedia/nobelprize_org/developer/manual-linkeddata/terms.html

[7] http://data.semanticweb.org/dumps/conferences/

Net 1.8 and a subset of Freebase with 15 000 entities), which have been widely used on link prediction experiments, are also used.

The Semantic Web dog food datasets are known to be correct and *locally* complete, i.e. no errors or missing relations *between contained entities*, therefore, the generated errors can be used as gold standard. We could not find any evaluation the of quality of AIFB, Semantic Bible or Nobel Prize. Since we cannot guarantee the quality of the data, the synthesized errors can be considered a silver standard. Because of incompleteness, some of the generated errors might actually be correct facts, meaning there can be false positives in the silver standard, and because of noise, there can also be false negatives in the silver standard.

The number of false positives is likely to be low even for highly incomplete datasets, since in general, the number of missing facts is significantly smaller than the number of possible facts ($|N_R||N_I|^2 - |\mathcal{A}_R|$) from which the generated wrong facts are drawn.

In the second part of the experiments we use DBpedia and NELL as large-scale real-world use cases. These datasets are known to be noisy and incomplete, with type assertion completeness estimated to be at most 63.7% on DBpedia (151). We do not synthesize any erroneous facts, and rank all the facts by their confidence values. Since we do not know the noisy facts or even the number of errors which exist in DBpedia, we manually evaluate the top-100 results.

6.4.2 Evaluation Measures

In our defined problem we use ranking measures to evaluate the performance of the error detection algorithms, since we compute scores for every triple in the graph and generate a ranking. Similar to link prediction papers we use the mean rank (μR), mean reciprocal rank (MRR), as well as their filtered variations fμR and fMRR (c.f. Equations 6.4 and 6.5), which filters out correctly higher ranked predictions.

$$fMRR = \frac{1}{|E|}\sum_{i=1}^{|E|}\frac{1}{rank_i - i + 1} \tag{6.4}$$

$$f\mu R = \frac{1}{|E|}\sum_{i=1}^{|E|} rank_i - i + 1 \tag{6.5}$$

We define E as the set of erroneous facts ordered by their rank in ascending order. Subtracting $i - 1$ from the rank ensures that better ranked true positives are filtered out. For instance, if $E = (1, 2, 3, 5, 8)$ its filtered sequence of ranks would be $(1, 1, 1, 2, 5)$.

		sembib	eswc	iswc	www	lrec	nobel	aifb	wn18	fb15k
$fMRR$	PaTyBRED $^{\text{LR}}_{10}$	0.800	0.835	0.811	0.212	0.754	0.690	0.014	0.584	0.618
	PaTyBRED $^{\text{RF}}_{10}$	0.840	0.927	0.933	0.559	0.747	0.680	0.120	0.860	0.770
	PaTyBRED $^{\text{SVM}}_{10}$	0.838	0.906	0.980	0.414	0.844	0.673	0.070	0.820	0.713
	PaTyBRED $^{\text{LR}}_{25}$	0.745	0.907	0.862	0.707	0.786	0.788	0.068	0.584	0.524
	PaTyBRED $^{\text{RF}}_{25}$	0.881	0.928	0.964	0.795	0.653	0.782	0.213	0.795	0.545
	PaTyBRED $^{\text{SVM}}_{25}$	0.848	0.860	0.980	0.537	0.822	0.788	0.045	0.570	0.765
$f\mu R$	PaTyBRED $^{\text{LR}}_{10}$	0.008	0.020	0.006	0.0023	0.011	0.076	0.041	0.00352	0.015
	PaTyBRED $^{\text{RF}}_{10}$	0.009	0.009	0.010	0.0003	0.006	0.080	0.031	0.00003	0.018
	PaTyBRED $^{\text{SVM}}_{10}$	0.011	0.012	0.008	0.0007	0.004	0.103	0.041	0.00003	0.014
	PaTyBRED $^{\text{LR}}_{25}$	0.005	0.022	0.003	0.0012	0.011	0.051	0.035	0.00349	0.014
	PaTyBRED $^{\text{RF}}_{25}$	0.003	0.028	0.010	0.0001	0.006	0.051	0.028	0.00004	0.020
	PaTyBRED $^{\text{SVM}}_{25}$	0.007	0.015	0.006	0.0003	0.005	0.063	0.028	0.00006	0.014

Table 6.1: Comparison of local classifiers and number of selected features on generated errors of kind 1

6.4.3 Parameter Settings

First, we evaluate how the different PaTyBRED parameters affect its performance. The evaluated parameters are the maximum path length (mpl), the maximum number of paths per length ($mppl$), the path selection heuristic measure ($pshm$), the number of locally selected features (k), and the local classifier (clf).

As far as the maximum path length (mpl) is concerned, the best results were achieved with $mpl = 2$, that is direct links and triangular patterns. Equivalent, inverse, and subproperty relations, as well as other kinds of associations can be exploited with direct links, while more complex associations with composed relations can be exploited with the triangular patterns. In none of the datasets used in our experiments, a $mpl > 2$ achieved better results. It seems that paths longer than two do not bring any information gain, while it significantly increase the search space and slows runtime.

In our experiments, we evaluate three different classifiers (clf): random forests (RF) (19), support vector machines (SVM) (42) and logistic regression (LR). We also try two different number of selected features k, i.e., $k = 10$ and $k = 25$. These numbers are low because we observed that only a small number of path and type features are relevant to the local relation classifiers. Table 6.1 show how the different settings of PaTyBRED $^{\text{SVM}}_{25}$ on various datasets. The results show that RF and SVM achieved the best results, while LR – which is used in PRA and SFE – lagged behind.

The heuristic measures used for selecting relevant adjacency matrices are those proposed in Section 6.3.1, i.e., $inter$, $m1$ and $m2$. As a baseline, we use the $random$ selection of paths. In order to better evaluate the quality of the paths selected we exclude the type features and consider exclusively the selected paths. We compared the heuristic measures on all the datasets presented in Section 6.4.1,

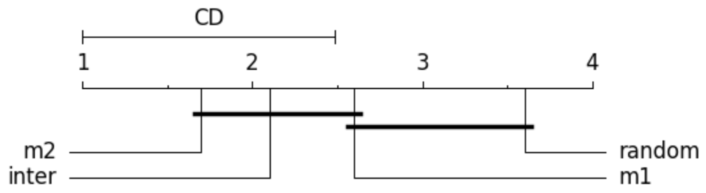


Figure 6.1: Critical distance diagram comparing path selection heuristics

		sembib	eswc	iswc	www	lrec	nobel	aifb	wn18	fb15k
$fMRR$	PaTyBRED	0.848	0.928	0.980	0.795	0.844	0.788	0.213	0.860	0.770
	TyBRED	0.463	0.782	0.315	0.744	0.693	0.758	0.205	—	—
	PaBRED	0.800	0.831	0.980	0.503	0.778	0.200	0.173	0.860	0.770
	SDValidate	0.265	0.140	0.218	0.109	0.307	0.464	0.022	—	—
	ProjE	0.102	0.175	0.047	0.098	0.138	0.187	0.048	0.004	0.014
	HolE	0.011	0.018	0.025	0.018	0.065	0.026	0.001	0.002	0.006
	TransE	0.058	0.001	0.000	0.001	0.039	0.051	0.005	0.001	0.000
$f\mu R$	PaTyBRED	0.003	0.009	0.003	0.0001	0.004	0.051	0.028	0.00003	0.014
	TyBRED	0.121	0.083	0.102	0.0740	0.113	0.084	0.085	—	—
	PaBRED	0.009	0.010	0.005	0.0008	0.004	0.227	0.056	0.00003	0.014
	SDValidate	0.355	0.397	0.326	0.3768	0.339	0.286	0.293	—	—
	ProjE	0.149	0.197	0.201	0.1796	0.179	0.177	0.252	0.18714	0.125
	HolE	0.204	0.258	0.108	0.1170	0.108	0.213	0.235	0.17304	0.083
	TransE	0.226	0.302	0.280	0.2381	0.163	0.320	0.329	0.26174	0.190

Table 6.2: Comparison of FMRR on generated errors of kind 1

ranked the measures and averaged them. In order to find out the significance of the results we perform Nemenyi Test with $\alpha = 0.05$. Since the number of datasets is rather small, the difference between $inter$ and $m2$ is not significant, however, they are significantly better than the random approach (c.f. Figure 6.1).

6.4.4 Comparison

Tables 6.2 and 6.3 report a comparison between PaTyBRED and the other state-of-the-art models. Table 6.3 refers to the datasets with errors with wrong entities of correct types and Table 6.2 refers to errors with wrong entities of any types. Table 6.3 does not contain results for WN18 and FB15k because the original datasets do not contain entity types, which prevents errors of kind 2 to be generated. For the same reason the results of SDValidate and TyBRED in Table 6.2 are not reported for WN18 and FB15k. We report values for $fMRR$ and $f\mu R$ ($f\mu R$ values divided by the total number of facts in the KB in order to make the values more comparable).

It is noticeable that the results for AIFB are significantly worse than other

		sembib	eswc	iswc	www	lrec	nobel	aifb
$fMRR$	PaTyBRED	0.482	0.553	0.941	0.609	0.532	0.022	0.272
	TyBRED	0.001	0.001	0.001	0.001	0.000	0.000	0.000
	PaBRED	0.579	0.567	0.941	0.625	0.486	0.250	0.205
	SDValidate	0.001	0.001	0.001	0.000	0.000	0.000	0.000
	ProjE	0.064	0.026	0.015	0.026	0.007	0.067	0.018
	HolE	0.022	0.015	0.043	0.049	0.059	0.053	0.004
	TransE	0.092	0.004	0.012	0.000	0.012	0.001	0.003
$f\mu R$	PaTyBRED	0.082	0.124	0.023	0.035	0.027	0.250	0.080
	TyBRED	0.597	0.503	0.512	0.495	0.551	0.526	0.496
	PaBRED	0.086	0.099	0.017	0.023	0.011	0.212	0.065
	SDValidate	0.570	0.457	0.467	0.506	0.495	0.495	0.475
	ProjE	0.215	0.362	0.223	0.245	0.254	0.274	0.269
	HolE	0.240	0.324	0.192	0.190	0.192	0.294	0.246
	TransE	0.247	0.308	0.239	0.337	0.148	0.413	0.339

Table 6.3: Comparison of FMRR on generated errors of kind 2

datasets. One of the reasons is the fact that it has no inverse relations, which can be extremely helpful on the error detection. Another reason is the fact that in AIFB the `author` is defined by 27 `author_n` relations, with n indicating the position in the authors list. That means it is necessary to not only model the author relation, but also all the n^{th}-author relations.

PaTyBRED, TyBRED and PaBRED were run with 6 different configuration: $clf \in \{\text{LR,RF,SVM}\}$ and $k \in \{10, 25\}$. For each dataset the results of the best performing configuration are reported. The maximum number of paths per length is set to 1000 and $m2$ is used as heuristic measure when the number of possible paths exceeds 1000, and the maximum path length is set to 2. The values reported for the embeddings methods were the best amongst number dimensions $d \in \{5, 15, 50, 100, 200\}$ and with the outlier detection, as explained earlier.

It is worth mentioning that the outlier detection helped improve the performance of embeddings' $f\mu R$ performance on average on 15%. The best results for the embedding methods were obtained with $d = 15$ or $d = 50$ depending on the dataset. The results reported for the knowledge graph completion in the original paper for ProjE on FB15k were with $d = 200$. On error detection with the same dataset the best performance was with $d = 50$, cutting the $f\mu R$ in half. Additionally, $d = 5$ and $d15$ also had better performance than $d = 200$. This indicates that when using embeddings for error detection, the dimensionality should be lower than for KGC. Since the dataset contains wrong triples, which shouldn't be fit by the model, overfitting can severely affect the performance (more than underfitting).

Our proposed method outperforms all the other methods, with the embedding methods having a surprisingly low performance. PaTyBRED performs best when combining types and paths, with TyBRED (with types only) and PaBRED (with

	sembib	eswc	iswc	www	lrec	nobel	aifb	nell	dbpedia
Paths	0.432	0.412	0.415	0.358	0.479	0.222	0.182	0.032	0.060
Types	0.568	0.588	0.585	0.642	0.521	0.778	0.818	0.968	0.940

Table 6.4: Proportion of path and type features selected

paths only) being generally worse. To further understand the importance of combining path type features, we analyze what kind of features are selected on the local classifiers and report the proportion of types and paths. Table 6.4 shows the average proportion of selected features over all relation classifiers with $k = 10$. Overall more type features are selected, but both kinds of features are relevant on the evaluated datasets. WN18 and FB15k are absent because they do not have type assertions, and therefore have only path features.

Table 6.3, where the erroneous facts contain wrong instances of correct types, shows how the performance of methods which rely on types exclusively (SDValidate and TyBRED) is similar to that of random ranking with $f\mu R$ around 0.5. It also shows how detecting errors of kind 2 is more difficult than those of kind 1, and it reveals the importance of using path features for detecting facts with wrong instances of correct types. We can also observe that PaBRED has performance similar to PaTyBRED and even better on some datasets for kind 2 errors, since type features are useless to detect those errors, and not considering type features ensures that these cannot potentially replace more useful path features. The only exceptions are on LREC and AIFBportal, where PaTyBRED has better $fMRR$ than PaBRED. However, on the same datasets PaBRED performs better in terms of $f\mu R$, meaning that it has better average rank but less highly ranked instances.

6.4.5 Manual Evaluation

In this section we perform a manual evaluation of PaTyBRED on two large-scale noisy datasets: DBpedia and NELL. We have a deeper look at the top-100 results and classify the triples as correct, wrong and *other errors*, i.e., correct triples with related errors, e.g. wrong or missing types of subject or object.

The results are shown in Figure 6.2 with PaTyBRED $^{\text{RF}}_{10}$ and PaTyBRED $^{\text{RF}}_{25}$ on DBpedia (dbp10, dbp25) and NELL (nell10, nell25). PaTyBRED seems to perform better on DBpedia with less local features (10) and more on NELL (25). Most of the other error cases occurred because of type assertion incompleteness, with the subject or object often having no types at all. Deleting these triples would lead to propagation of incompleteness. These cases could be automatically detected, and some of them fixed if the type completion methods (124; 150) are combined with error detection. The quality of predicted types can be asserted by the improvement of the scores of triples containing the entities with predicted types.

Some of the errors come from wrong links in Wikipedia pages, which originate from confusions between entities of similar names. One example of such problem is the fact `formerTeam`(`Alan_Ricard`, `Buffalo_Bill`), where the correct entity

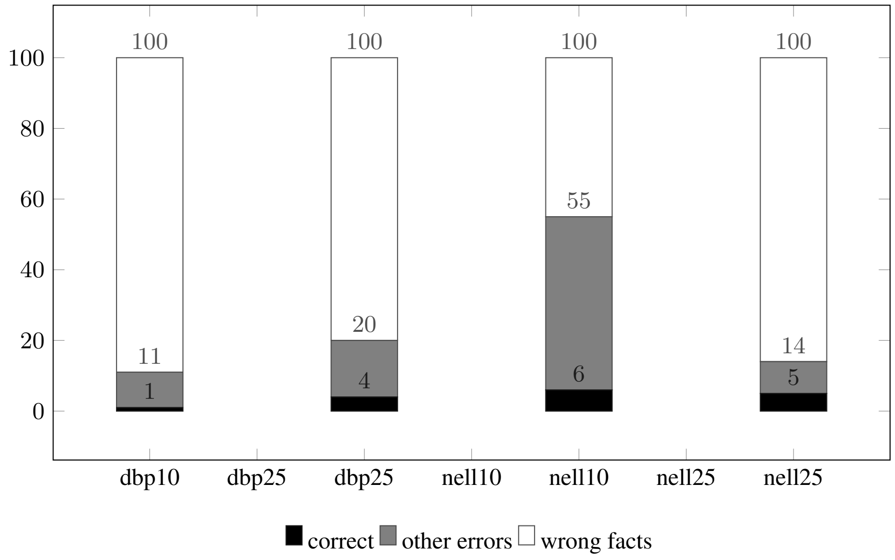


Figure 6.2: Manual evaluation on DBpedia and NELL

should be the NFL team `Buffalo_Bills` instead of the character `Buffalo_Bill`. In Chapter 7 we propose an approach which makes use of disambiguation links (in DBpedia the `dbo:wikiPageDisambiguates` relation) and string distance to correct these errors. By replacing subject or object with their respective candidates and computing the triple scores, we can substitute the wrong triple with the best scoring candidate (a similar idea has been used for correcting links in Wikipedia (206)). In the manual evaluation, five of the DBpedia errors could potentially be fixed with such an approach.

Entities in DBpedia are described in much more detail than in NELL (166). Around 20% of NELL's instances are untyped, while in DBpedia only 1% of them have no types other than `owl:Thing`. Furthermore, in NELL, reasoning is already used in the construction process for error detection, which means that very obvious errors and violations of the underlying ontology are already removed. This may explain why NELL performs better with more locally selected features, as opposed to DBpedia. By increasing the number of features the number of correct facts with untyped subject or object in the top-100 was reduced from 48 to 9, and the number of actual errors increased from 45 to 86.

Amongst the five correct facts from DBpedia which were wrongly predicted to be errors, two were from the relation `seeAlso`. That is understandable since the relation has very wide semantics, and any pair of vaguely related entities can be correct. Modelling such a complex relation can be a difficult task. Another error detected was `location`(`Alan_Turing_Institute`, `British_Library`), which is a correct fact, but the unique case of an organization which is located in a library. The last case is with the `foundedBy` relation, with two cases of newspapers found by political parties, not persons.

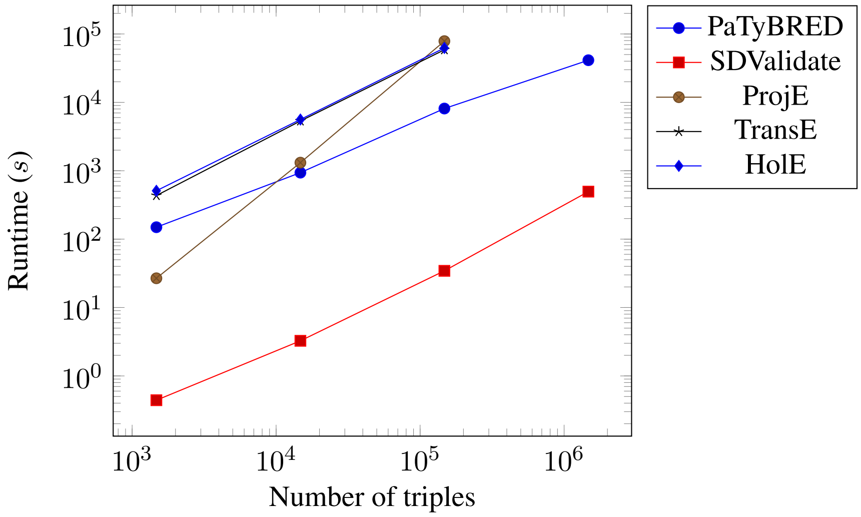


Figure 6.3: Runtime comparison of the evaluated methods

We also make a scalability comparison of the evaluate methods. The results are shown in Figure 6.3. We can observe that SDValidate has by far the lowest runtimes, since it is a model simpler than the others. Amongst the embedding methods, ProjE which directly optimizes the rankings in the link prediction task, has the steepest runtime growth. HolE and TransE have similar scalability being more scalable than ProjE. PaTyBRED, due to the aggressive local feature selection and sampling, has the least steep of the curves. This indicates the appropriateness of PaTyBRED for handling large datasets.

The scalability test is performed on synthesized replica of DBpedia with the M3 model (c.f Chapter 4) of sizes {0.01%, 0.1%, 1% and 10%} of the original size, that means the number of triples varies from around 1.5k to 1.5M triples.

6.5 Conclusion

We have shown that although the error detection problem is similar to knowledge completion, methods which perform well in knowledge completion might not necessarily be appropriate for error detection. We propose PaTyBRED, a robust supervised error detection method which relies on type and path features, and compare it with state-of-the-art error detection and knowledge graph completion methods. We demonstrate the importance of combining those path and type features together, and we also perform a manual evaluation of our approach on DBpedia and NELL.

Chapter 7

Correction of Confusions Between Entities

7.1 Introduction

Knowledge graphs are known to be both often incomplete and incorrect. Several link prediction and error detection methods have been proposed, however, few of them explicitly focus on error correction or address the problem of choosing which absent facts should be added to the knowledge graph.

The problem is that the number of possible relation assertions grows quadratically with the number of instances $n_c = n_i^2 n_r - n_f$, where n_i is the number of instances, n_r the number of relations and n_f the number of existing facts in the graph. For large datasets such as DBpedia, Wikidata and YAGO, computing the confidence score of all these facts is challenging. While pruning possible facts which violate ontology constraints, especially domain and range restrictions of relations, can significantly reduce the search space, the problem is still very challenging. To illustrate the size of the search space, in DBpedia (2016-10) $n_c \approx 4.4 \times 10^{17}$ facts; when filtering those triples which violate the domain and range restriction the number is reduced to $n_c \approx 2.8 \times 10^{17}$, which is still too large to compute confidence for all those candidates.

A promising approach for enriching a KG with some of its missing facts is the correction of erroneous facts. Some of the wrong facts which exist in a KG can be corrected. The error originates from some problem in the knowledge acquisition process, or in the source data. A good example of the latter are errors in Wikipedia, which serves as the main source of DBpedia and YAGO. If any of the links in the infobox are wrong, the extraction process generates a wrong fact. NELL, on the other hand, has text as main source of information and in many cases the source text has correct information, which cannot be extracted correctly.

In both cases it is common that an instance is confused with another one of a similar name (i.e., label or IRI). For example, the fact `formerTeam( Alan_Ricard, Buffalo_Bill)` is an erroneous fact from DBpedia which originates from a typo in Wikipedia: when referring to the NFL team Buffalo_Bills, the *s* was missing, therefore, the NFL team was confused with the character Buffalo_Bill. In NELL the entity insect_raccoon exists because of problems when extracting the fact that raccoons prey on insects, and is confused with mammal_raccoon.

Some relation assertion error detection approaches, such as PaTyBRED (121) and SDValidate (151), rely on type information, and since erroneous type assertions are also a common problem, that might result in correct relation assertions with an instance of incorrect or incomplete types being wrongly identified as erroneous. Therefore, combining such methods with type prediction (124; 150) is beneficial to rule out cases where the error is detected rather due to a missing or incorrect type of the subject or object than due to an erroneous relation assertion.

Therefore, it is relevant to make a careful analysis of detected errors, identify the source of each error, and if possible correct them. In this chapter, we propose CoCKG (Correction of Confusions in Knowledge Graphs), an automatic correction approach which resolves relation assertion errors caused by instance confusion. The approach relies on error detection methods as well as type predictors to assess the confidence of the corrected facts. It uses approximate string matching and exploits both searching for entities with similar IRIs as well as Wikipedia disambiguation pages (if available) to find candidate instances for correcting the facts.

7.2 Proposed Approach

Our approach consists of first running an error detection algorithm (PaTyBRED in the case of our experiments), selecting the top-k facts most likely to be wrong. In the next step, the error is heuristically verified to be an actual relation assertion error and not caused by missing type assertions in the object or subject with a type predictor tp. In the final step, candidate entities are retrieved, and if any of the candidates significantly improves the likelihood of the triple being right, we replace it by that candidate. The function CORRECT_TRIPLE in Algorithm 3 gives an overview of how CoCKG works. The parameter $\mathcal{K}$ is the set of all triples in the knowledge graph, $\mathcal{T}_{err}$ is the set of triple and confidence pairs generated by the error detection model (ed), tp is the type predictor, mc is the minimum confidence threshold, and mcg the minimum confidence gain threshold, i.e. the ratio of the new and old triple scores. In the next subsections we discuss the other parts in more details.

7.2.1 Type Prediction

After selecting the k triples most likely to be wrong, we first check if their confidence is low because of missing or wrong instance types (subject or object). In order to do that, we run a type predictor tp on the subject and object instances. We use as tp a multilabel random forest classifier based on qualified links (i.e. ingo-

Algorithm 3 Knowledge base correction process

```
function CORRECT_TRIPLES($\mathcal{K}, \mathcal{T}_{err}, ed, tp, mc, mcg$)
    $T_{corr} \leftarrow \emptyset$
    for $t, score_t \in \mathcal{T}_{err}$ do
        $s, p, o \leftarrow t$
        $s_{tp} \leftarrow$ PREDICT_TYPES($tp, s$)
        $o_{tp} \leftarrow$ PREDICT_TYPES($tp, o$)
        if $\neg$(CONF_NT($ed, t, s, s_{tp}$) $\vee$ CONF_NT($ed, t, o, o_{tp}$)) then
            $s_{cand} \leftarrow$ GET_CANDIDATES($s$)
            $o_{cand} \leftarrow$ GET_CANDIDATES($o$)
            $T_{cand} \leftarrow \{(s_i, p, o) | s_i \in s_{cand}\} \cup \{(s, p, o_i) | o_i \in o_{cand}\}$
            $T_{cand} \leftarrow T_{cand} - \mathcal{K}$
            $c_{best}, max_{conf} \leftarrow nil, conf$
            for $c \in T_{cand}$ do
                if $s \in domain(p) \wedge o \in range(p)$ then
                    $score_c \leftarrow$ CONF($ed, c$)
                    if $score_c \geq mc \wedge score_c / score_t \geq mcg$ then
                        $c_{best}, max_{conf} \leftarrow c, score_c$
            if $c_{best} \neq nil$ then
                $\mathcal{T}_{corr} \leftarrow \mathcal{T}_{corr} \cup \{(c_{best}, t)\}$
    return $\mathcal{T}_{corr}$
```

ing links paired with subject type and outgoing links paired with object type), as described in (124). If the set of predicted types of the subject are different from the actual types, we change the type features used by ed and compute a new confidence for the triple (c.f. CONF_NT). If the new score satisfies mc and mcg, then we conclude that the error was in the subject type assertions. The same is done for the object, and if in neither case the confidence thresholds are satisfied, we proceed to the next part where we try to substitute the subject and object with their respective lists of candidates.

Combining the type prediction process with the error detection also has the advantage that the newly predicted types can be validated on triples containing the instance whose types were predicted. This can help support, or contradict the type predictor, possibly detecting types which are wrongly predicted by identifying triples where the score is lowered with the new types.

7.2.2 Retrieving Candidates

One simple way to find candidate entities to resolve entity confusions is to use the disambiguation links. However, disambiguation pages are only available for Wikipedia-based knowledge graphs, and furthermore are not available for all entities (e.g. `Ronaldo` has no disambiguation page). In some cases the disambiguation pages miss important entities (e.g. the page `Bluebird_(disambiguation)` misses the entity `Bluebird_(horse)`. Therefore, in order to correct the erroneous fact `grandisre(Miss_Potential,Bluebird)`), we need an additional source of candidates.

Since in our experiments we consider DBpedia and NELL, which have informative IRIs (in the case of DBpedia extracted from the correspondent Wikipedia's page), we search for candidate entities which have similar IRIs. Alternatively, it could also be done with entity labels. This would be useful in KGs which have uninformative IRIs (e.g. Wikidata and Freebase). For simplicity, in this chapter, we refer to the informative part of an IRI as the "name" of the entity.

Retrieving all the instances of similar names can be a complicated task. This kind of problem is known as approximate string matching, and it has been widely researched (134; 218). For our method we use an approximate string matching approach based on (127). First, we remove the IRI's prefix and work with the suffix as the entity's name. We then tokenize the names and construct a deletions dictionary with all tokens being added with all possible deletions up to a maximum edit distance d_{max} threshold. This dictionary contains strings as keys and lists with all tokens which can turn into the key string with up to d_{max} deletions as values. Only pairs of tokens which share a common deletion string can have an edit distance less or equal than d_{max}. We also have a tokens dictionary which has tokens as keys and lists of entities which contain a given token as values. With that, given a token and a d_{max} we can easily obtain all the entities which contain that a string approximately similar to that token up to the maximum edit distance.

When searching for entities similar to a given entity, we perform queries for every token of the entity's name and we require that all tokens are matched. That is, for a certain entity to be considered similar, it has to contain tokens similar to all the tokens of the queried entity. A retrieved entity may have more tokens than the queried entity, but not less. The idea is that in general, when entering an entities name manually (e.g., in Wikipedia), it is common to underspecify the entity, but highly unlikely to overspecify it. E.g., it is more likely that `Ronaldo` is wrongly used instead of `Cristiano_Ronaldo` than the other way around. Furthermore, it reduces the number of matched entities.

We also perform especial treatment on DBpedia and NELL entity names because of peculiarities in their IRI structures. In DBpedia it is common to have between parentheses information to help disambiguate entities, which we consider unnecessary since the entity types are used in the error detection method. In NELL the first token is always the type of the entity, therefore, for similar reasons, we ignore it.

7.2.3 Correcting Wrong facts

At this point, for each assertion identified as erroneous, we have our list of candidate instances for subject and object from the disambiguation links and approximate string matching. We then compute a custom similarity measure $s(e_1, e_2)$ between an entity e_1 and a candidate e_2. Each entity e_i consists of a set of its tokens. The measure we propose consists of two components. The first is the sum of Levenshtein (d_L) distance of all matched tokens, and the second considers the number of unmatched tokens to capture a difference in specificity. The set of approximately matched token pairs is represented by $\mu(e_1, e_2)$ and the constant c is the weight of the second component. This measure is used to sort the retrieved candidates, to prune them in case there are too many, and to break ties when deciding which of the top-scoring candidates should be chosen.

$$s(e_1, e_2) = \sum_{(t_1,t_2)\in\mu(e_1,e_2)} d_L(t_1, t_2) + c\frac{|e_1| - |\mu(e_1, e_2)|}{|e1|} \tag{7.1}$$

In case the relation has domain or range restrictions, we remove the candidates which violate these restrictions. Later, for each of the candidates, we generate triples by substituting the subject and object by each of the instances in its candidates lists (first substitute subject only, then object only). That is, the total number of candidate triples is the sum of the size of the subject and object candidates list. We do not create candidate triples by substituting both the subject and object at the same time because, although possible, we assume the simultaneous confusion of both instances to be highly unlikely.[1] This is also done in order to make the number of candidate triples linear instead of quadratic.

We then remove the candidate triples which are already existent in the KG. We compute the confidence of all candidate triples and select that with highest confidence, given that mc and mcg are satisfied. Our method then outputs a list of triple pairs containing the wrong triple detected and the corrected triple predicted.

7.3 Experiments

In our experiments we run CoCKG on DBpedia (2016-10) and NELL (08m-690), then we manually evaluate the triples corrected by our approach. We run PaTyBRED on both datasets and select top-1% facts most likely to be errors to be processed by our correction method. We classify each corrected fact in four different categories:

1. WC: wrong fact turned into correct
2. WW: wrong fact turned into another wrong fact
3. CW: correct fact turned into wrong fact
4. CC: correct fact turned into another correct fact

Our approach was run with $mc = 0.75, mcg = 2$ and entity similarity measure with $c = 1.5^2$. That resulted in 24,973 corrections on DBpedia and 616 correction on NELL. It also detected that 873 (569) errors were caused by wrong types in

[1] For that to happen in the case of DBpedia, a Wikipedia user would have to go to the wrong article page and insert a wrong link in the infobox.

[2] The parameter values were selected based on heuristics and may not be optimal

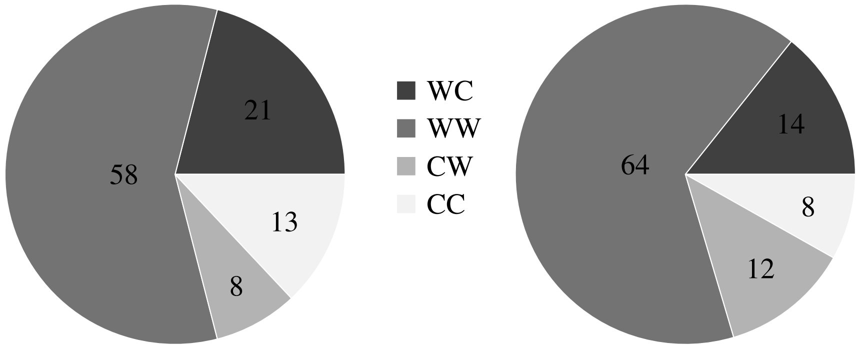


Figure 7.1: Manual evaluation on DBpedia and NELL respectively

DBpedia (NELL). Since manually evaluating all these corrections would be impossible, we randomly select 100 correction on each to perform the evaluation.

The results of our manual evaluation are shown in Figure 7.1. The proportion of facts successfully corrected (case 1) was rather low. While our approach can potentially improve the results by tweaking the parameters, and possibly using ensembles of different type predictors and error detectors, it currently cannot be used as a fully automatic approach. However, we believe that combining our approach with active learning is a promising direction which, with the help of specialists, could significantly improve results.

When evaluating some relations individually, we notice that some of them achieve good results. For instance, the relations `sire`, `damsire`, `grandsire` and `subsequentWork` reaching more than 90% of successful corrections (case 1). The results are good for these relations because horses are often named after other entities and artists often have albums named after themselves, which makes confusions easy to happen.

One of the problems of our approach is that since it relies on PaTyBRED, which cannot find many relevant path features on DBpedia and NELL (121), it is difficult to distinguish between candidate entities of same type. For example, in NELL, the entity `person_paul` as object of `book_writer` relation is always corrected with `writer_paul_feval`.

The decision to generate candidate triples by corrupting either the subject or object seemed to have worked well for DBpedia, where we could not find a triple where both subject and object were wrong. On the other hand, in NELL such case was observed a few times, e.g. `ismultipleof(musicinstrument_herd, musicinstrument_buffalo)` whose object was corrected to `mammal_buffalo` but the subject remained wrong.

Also, our assumption that confusions tend to use a more general IRI instead of a more specific, requiring all tokens of the queried to be matched, does not always hold. One example of confusion in DBpedia which contradicts this assumption is `language(Paadatha_Thenikkal , Tamil_cinema)`, whose corrected object would be `Tamil_language` and could not be retrieved by our approach. While this can be a problem, dropping this assumption also means that more candidates entities will be retrieved, increasing the number of unrelated candidates. This would possibly result in more occurrences of cases 2 and 3. Further experiments would have to be conducted in order to evaluate the effects of such change.

7.4 Conclusion

In this chapter we proposed CoCKG, an approach for correcting erroneous facts originated from entity confusions. The experiments show that CoCKG is capable of correcting wrong triples with confused instances, with estimated precision of 21% of the produced corrections in DBpedia and 14% in NELL. The low precision values obtained do not allow this process, as of now, to be used for fully automatic KG enrichment. Nevertheless, it works as a proof of concept and can be useful, e.g., as suggestions from which a user would ultimately decide whether to execute.

Part II

TBox Refinement

Chapter 8

Generation of SHACL Relation Constraints

8.1 Introduction

Since creating a large-scale knowledge graph by manual curation would be an endeavour which is hardly feasible, today's knowledge graphs are usually built using heuristic methods (166). Since those heuristics have to trade off coverage and precision, the resulting graphs are not free from errors. Hence, many works on automatic relation validation and error detection in knowledge graphs have been proposed (149).

In most cases, those approaches focus on individual relation assertions, and the outcome is a list of individual errors, which can be very long. Hence, it is difficult for experts to review and validate the results of those approaches. Furthermore, the approaches are only valid for one graph at hand and cannot be trivially reused if the graph changes. Hence, for constantly evolving graphs such as Wikidata or the Live edition of DBpedia, they are hardly applicable.

As ontologies are one of the pillars of the Semantic Web, and many knowledge graphs come with their own ontologies, they can also be used for validating individual assertions in a knowledge graph. However, designing a good ontology can be a challenging task, therefore there has been a lot of work on learning ontologies from data, using methods such as inductive logic programming (ILP) (24) or association rule mining (204) for automatically learning ontology axioms.

One of the main problems with these methods is the restricted expressiveness of the learned ontologies. Modern knowledge graphs are often complex, and constraints may require the use of axioms which cannot be expressed in OWL or cannot be learned by current state-of-the-art methods. Furthermore, the intended and the actual use of a property often diverge, leading to situations where a single ontology can hardly describe the different, often competing usages of a property.

One example for the latter case is the `president` in DBpedia. The relation is originally conceived to be used to define the person who presides an organization, hence in DBpedia's ontology it has the domain Organisation and range Person. However, the relation is often used to define the president which a member of the government served, e.g., president(Rex_Tillerson, Donald_Trump).

In order to allow both kinds of assertions, the domain of the relation should be more flexible accepting Organisation or Person. A possible solution using RDFS domain and range axioms is to use the most specific common parent of the two classes, that is Agent, however, that also allows subjects to be of the classes Deity, Family, which would be undesirable.

For the second use of the relation, path constraints can be useful for describing the relation. In the DBpedia both members of the government and presidents have successor relation assertions indicating the person who occupied their respective positions after them. We know that a member of the government should have the same president as its predecessor, or the successor of the president of its predecessor. The former case happens when a president has, e.g., different secretaries of state during its government, and the latter when the secretary of state is the first nominated by a new president. This can be represented with the disjunction of two graph path constraints:

$$\begin{aligned}\texttt{president}(a,b) \rightarrow \quad & (\texttt{successor}(c,a) \land \texttt{president}(c,b)) \lor \\ & (\texttt{successor}(c,a) \land \texttt{president}(c,d) \land \texttt{successor}(d,b))\end{aligned}$$

In this chapter, we propose an approach that can learn complex validation rules for knowledge graphs. For representing those validation rules, we propose to use SHACL[1] (Shapes Contraint Language), a versatile constraints language for validating RDF graphs recommended by the W3C. It is a highly expressive language which uses SPARQL queries to define constraints called shapes.

In order to learn the constraints we use PaTyBRED (121), which uses type and path features to detect relation assertion errors and learns a decision tree for each relation. These decision trees are then converted to SHACL constraints, which can be easily integrated into a knowledge graph and used to validate the data.

8.2 Generating SHACL constraints

In this section we present our approach for translating decision trees learned with PaTyBRED into SHACL relation constraints. It is important to note that we focus on the creation of constraints for relations between entities (owl:ObjectProperty). Constraints for owl:DataProperty relations containing, e.g. numerical, textual or geographical data, are out of the scope of this work.

Learning such constraints has an important advantage when comparing to is opaque relation assertion error detection methods, such as embeddings. The SHACL constraints are human-readable and can be directly evaluated and improved by specialists without requiring the manual evaluation of its output.

[1] https://www.w3.org/TR/shacl/

8.2.1 SHACL

Shapes Constraint Language (SHACL) is a language for validating RDF graphs against a set of conditions, which are provided as *shapes* expressed in the form of an RDF graph called *shapes graph*. The RDF graphs that are validated against a shapes graph are called *data graphs*. The SHACL specification is divided into SHACL Core and SHACL-SPARQL. SHACL Core consists of frequently needed features for the representation of shapes, constraints and targets. The SHACL Core language defines shapes about the focus node itself (node shapes) and shapes about the values of a particular property or path for the focus node(property shapes).

SHACL-SPARQL consists of all features of SHACL Core plus the advanced features of SPARQL-based constraints and an extension mechanism to declare new constraint components. Constraint can be written as SPARQL ASK or SELECT queries. These queries are interpreted against each shape focus node. If an ASK query does not evaluate to true for a given node, then the constraint is violated. Constraints described using a SELECT query must return an empty result set when conforming with the constraint and non-empty set when violated.

SHACL also supports three different constraint severity levels: Info, Warning and Violation. The different levels have no impact on the validation, but may be used by to categorize validation results. It is up to the user to define how the different severity levels are handled.

8.2.2 Generation Process

We generate SHACL constraints by first training decision trees for relation validation using PaTyBRED. Next, we consider the learned decision tree, more specifically, the subtree with the conditions for an example to be classified as erroneous. The subtree is then converted it into a logical expression, whose negation is used as a constraint for the relation. The idea is that we used as constraints the negation of the expression that defines the examples which are predicted by PaTyBRED to be highly erroneous. In the rest of this section we describe in details how the generation of the constraints is done.

Firstly we identify the nodes which contain only – or mostly – erroneous relation assertions. For a node not to be pruned it needs to satisfy minimum support and confidence thresholds, or be an ancestor of a node which satisfies the thresholds. If a non-leaf node satisfies both thresholds, all its ancestors can be pruned (to avoid redundancies). This pruned tree can then be directly converted into a logical expression which will translate the conditions into a single SHACL constraint. Each literal $L_{i,j}$ is a variable which may be negated or not. This can be directly translated to node conditions in the tree which are satisfied (right branch) or not (left branch).

Figure 8.1 shows an example of how the pruning process works. The c_i nodes represent conditions and the 2-dimensional vectors at the leaf nodes represent the classes distribution, where the first dimension indicates the number of negative

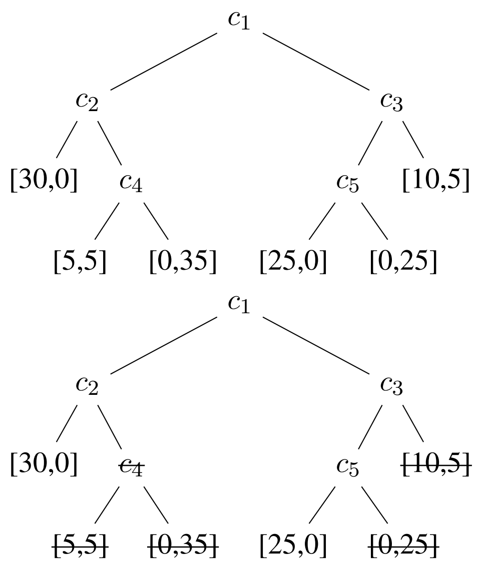


Figure 8.1: Example of decision tree pruning

examples (erroneous) and the second the number of correct examples. The pruned tree from Figure 8.1 can then be converted into the following expression:

$$\neg((\neg c_1 \land \neg c_2) \lor (c_1 \land \neg c_3 \land \neg c_5))$$

A confidence value of 1 means that only pure nodes containing exclusively negative examples can be selected. It also means that if the learned constraints are to be applied on the original data, no existing errors can be detected. In order to enable detection of preexisting errors, the confidence threshold of less than 1 is necessary. We can use different confidence thresholds to define different SHACL constraints with different severity levels. Constraints with lower confidence may be used as warnings, while higher confidence values close to 1 maybe used as violations.

Since PaTyBRED relies on path and type features, all conditions in the decision tree nodes will be of the following kinds: subject type, object type and path.

The decision tree's logical expression can be directly translated to SHACL Core using `sh:and`, `sh:or` and `sh:not`. A shape for a relation `:r` can be defined with `:rShape a sh:NodeShape`. We define the target nodes of the shape as subjects of the target relation with `sh:targetSubjectsOf`. Subject type features test if the subject of the relation assertion is of a certain class `:C`. This can be done in SHACL with `:rShape sh:class :C`. The object can be restricted to a type `:C` with the expression `:rShape sh:property [sh:path :r; sh:class :C]`.

The main problem with SHACL Core is when translating path features. In the decision trees we consider pairs of subject and object as examples, however SHACL validation is performed on a single node basis. Its vocabulary provides the components for property pair constraints sh:equals and sh:disjoint. The first requires that for all focus nodes the set of nodes reach by both properties (or property paths) should be identical, while the second requires that the sets are disjoint. The problem is that what we need to represent is the subsumption relation between a pairs of paths.

This can be illustrated with Example 1. If we want to validate the relation assertions of :playedFor we need to consider the subject-object pairs (:Anelka, :Chelsea) and (:Anelka, :Arsenal). Assuming every (s,o) pair is required to also be connected by the path :livedIn/:^locatedIn in order to be correct, then both assertions should be valid. However, since the set of objects reached from :Anelka with :playedFor are {:Chelsea,:Arsenal} and the objects reached with :livedIn/:^locatedIn are {:Chelsea,:Arsenal,:Westham}, an error on the focus node :Anelka would be detected if we use sh:equals to represent the path pattern.

Example 1:

```
:Anelka      :playedFor  :Chelsea .
:Anelka      :playedFor  :Arsenal .
:Anelka      :livedIn    :London .
:Chelsea     :locatedIn  :London .
:Arsenal     :locatedIn  :London .
:Westham     :locatedIn  :London .
```

Example 2:

```
:Anelka      :playedFor  :Chelsea .
:Anelka      :playedFor  :Arsenal_ARG .
:Anelka      :livedIn    :London .
:Chelsea     :locatedIn  :London .
:Arsenal_ARG :locatedIn  :Sarandi .
:Westham     :locatedIn  :London .
```

A similar problem happens if we try to use the negation of sh:disjoint. In Example 2 the subject-object pair (:Anelka, :Chelsea) is correct, whereas (:Anelka, :Arsenal_ARG) is incorrect, since the pair is not connected with the path :livedIn/:^locatedIn because :Anelka did not live in :Sarandi. If we validate the data using the negation of sh:disjoint, the sets of objects reached with the two paths are not disjoint because both have :Chelsea, therefore the validator would assume that for the focus node :Anelka there is no assertion error with relation :playedFor. This would only work if the relation :playedFor were functional. For that reason, we cannot correctly translate the PaTyBRED decision trees into SHACL Core.

In SHACL-SPARQL path features can be correctly translated in a more intuitive way, since it is possible to work directly with subject-object pairs. Moreover, it has the advantage of using a well-established and widely used language instead

Feature	SHACL-SPARQL	SHACL Core
$C(s)$	`{$this a :C}`	`_:b sh:class :C .`
$C(o)$	`{?o a :C}`	`_:b sh:property [sh:path :r; sh:class :C] .`
$p(s,o)$	`{$this :p ?o}`	N/A
$p(X,s)$	`{?X :p $this}`	`_:b sh:property [sh:path [sh:inversePath :p]] .`
$p(s,X)$	`{$this :p ?X}`	`_:b sh:property [sh:path :p] .`
$p(X,o)$	`{?X :p ?o}`	N/A
$p(o,X)$	`{?o :p ?X}`	N/A

Table 8.1: PaTyBRED features translation into SHACL

of requiring the learning of a whole new vocabulary. The template for a SHACL-SPARQL relation constraint is shown below. The SPARQL constraint is defined with the `sh:SPARQLConstraint` component. The variable `$this` indicate the focus node and `?o` its correspondent objects in the target relation.

```
:relSHACLShape a sh:NodeShape ;
sh:targetSubjectsOf :rel ;
sh:sparql [
 a sh:SPARQLConstraint ;
 sh:select """
  SELECT $this ?o
  WHERE {
   $this :rel ?o .
   FILTER(!(E))
  }
 """ ;
] .
```

The relation constraints expression is represented by `E`, which is negated because during validation the select query needs to return an empty set if `$this` satisfies the constraint. Table 8.1 shows how the PaTyBRED features can be converted into SHACL-SPARQL and Core. The path `:p` represents a property chain `:r1/.../:rn` in SHACL-SPARQL, with the ˆ character before a relation indicating the inverse of the relation.

For the earlier `president` relation example from DBpedia the expression `E` could be defined as shown below. Every variable in the logical formula is expressed as a different EXISTS clause. Negated literals can be represented by simply negating a single variable EXISTS clause. Alternatively, disjunctions and conjunctions can be represented in a single EXISTS clause using "UNION" and "." respectively, however expressing negations would be complicated.

```
EXISTS {?o a :Person} &&
(EXISTS {$this a :Organisation} ||
 (EXISTS {$this a :Person} &&
  (EXISTS { $this ^:successor/:president ?o} ||
   EXISTS { $this ^:successor/:president/:successor ?o}
  )
  )
)
```

It is important to note that the number of variables and the length of the expression will depend on the number of features selected defined by PaTyBRED. It also depends on the decision tree settings, such as the maximum depth, maximum number of leaf nodes, minimum samples on leaf and on split.

8.3 Experiments

To evaluate the learning of relation constraints, we compare the constraints learned with our approach with domain and range restriction axioms learned with statistical schema induction (SSI) (204). We conduct experiments on two large-scale knowledge graphs, i.e., DBpedia and YAGO.

SSI uses association rule mining to induce domain and range restrictions from the data. In order to learn such restrictions, it generates transaction tables where transactions correspond to relation assertions and items correspond to relation and subject types, for domain learning, or relation and object types, for range learning. Then rules of the forms $\exists r.\top \sqsubseteq C$ and $\exists r^{-1}.\top \sqsubseteq C$ (i.e., domain and range axioms respectively) are learned with association rule mining.

We run both methods with minimum confidence of 0.95 and minimum support of 50 instances. For SSI, we use the most specific domain and range axioms that satisfy the minimum confidence and support thresholds. Every constraint and axiom preserves its original confidence value, and for every fact violating the constraints we assign the confidence of its original axiom.

We rank the detected errors by the scores, and select the top-10000 (top-10k) errors with each method (less than 1% of the total amount of relation assertions). Since many triples are in the top-10k of both methods, we manually evaluate only those triples which are selected by one method and not the other.

We decided to evaluate the compared approaches based on their ability to detect existing errors. Evaluating the quality of the generated constraints by themselves, without considering their ability to detect errors, would be subjective. Since both methods induce the constraints from the ABox and the detection of errors is their main application, we think it is fair to evaluate the approaches by how accurately they can detect errors in an incorrect dataset like DBpedia.

The learned SHACL constraints are translated from PaTyBRED decision trees learned with $mpl = 2$, $mppl = 5000$, $k = 10$ and $nneg = 1$. Out of 646 `owl:ObjectProperty` relations from DBpedia 2015-10 considered, we learned 440 SHACL constraints. Out of those 122 were simple domain and range restrictions, 224 were combinations of subject and object types and 94 had path features (from which 43 had length 2). The relevance of triangular path features in DBpedia is rather small, contributing to only 6% of the features selected (c.f. Table 8.1).

Figure 8.2 shows the results of our manual evaluation on DBpedia[2]. Since there is some overlap in the top-10k triples detected with each method (380 triples in DBpedia and 5963 in YAGO), we also present the results of the evaluation on the differences between the two methods in Figure 8.3. We call SHACL-SSI the set of triples selected by the SHACL constraints and not by SSI, and SSI-SHACL the set of those selected by SSI and not SHACL. We then select random samples of 100 errors from SHACL-SSI and SSI-SHACL and manually evaluate them.

In the manual evaluation we classify the triples detected as errors into four categories.

- WT-CC: wrong triple with correct types
- WT-WC: wrong triple with wrong types
- CT-WC: correct triple with wrong types
- CT-CC: correct triple with correct types

We consider a fact to have wrong type (WC), if either the subject or the object in the triple has wrong or missing triple assertions. That includes instances which are untyped, has too general types, or has wrong type assertions. A relation assertion is considered correct (CT) if the pair of subject and object entities is correct, independent of their types.

The results from Figure 6.2 show that the SHACL constraints are better at detecting wrong triples, with a higher number of wrong triples with correct types (WT-CC), which are more difficult to detect. Also, the number of correct triples with wrong types (CT-WC) is reduced, showing that the more flexible SHACL constraints are better at modeling noisy and incomplete relations. We suppose that on datasets where path features are more relevant, our learned SPARQL constraints would have a greater advantage when compared to SSI, since the latter only exploits subject and object types.

We illustrate the results obtained with our method showing two examples of SHACL constraints learned on DBpedia learned for the relations `parent` and `kingdom`, as well as the relation `isCitizenOf` learned on YAGO. The `:parentShape` constraint uses exclusively path features, and it exploits the fact that generally people have children with their spouses and that it is the inverse of the child relations. In the DBpedia ontology child and parent are not the inverse of each other, with the two relations having different number of assertions. By considering the two path features, the constraint is more flexible requiring that neither paths connect subject and object for a relation assertion to violate the constraint. Such flexibility is particularly important on incomplete datasets, such as DBpedia.

[2] The manual annotations can be accessed in `http://data.dws.informatik.uni-mannheim.de/hmctp/shacl-eval/`

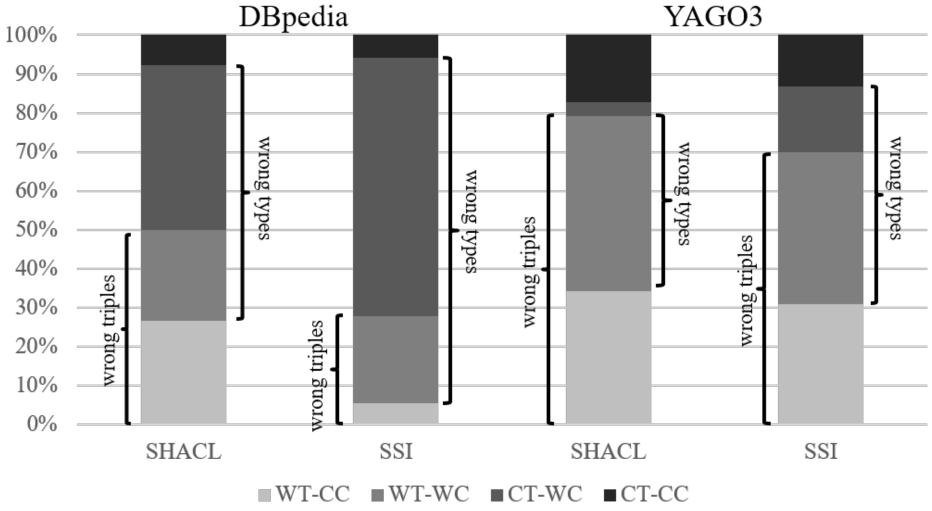


Figure 8.2: Manual evaluation on DBpedia and YAGO

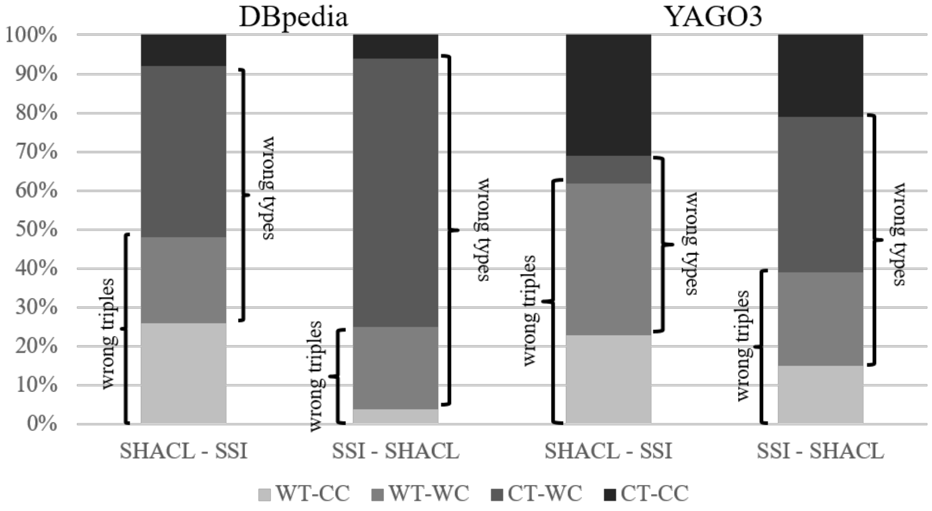


Figure 8.3: Manual evaluation of the differences between SHACL and SSI on DBpedia and YAGO

```
:parentShape a sh:NodeShape ;
sh:targetSubjectsOf :parent ;
sh:sparql [
 a sh:SPARQLConstraint ;
 sh:select """
  SELECT $this ?o WHERE {
   $this :parent ?o .
   FILTER(!EXISTS {$this :parent/:spouse ?o} &&
          !EXISTS {$this ^:child ?o} )}
 """ ;
] .

:kingdomShape a sh:NodeShape ;
sh:targetSubjectsOf :kingdom ;
sh:sparql [
 a sh:SPARQLConstraint ;
 sh:select """
  SELECT $this ?o WHERE {
   $this :kingdom ?o .
   FILTER(!EXISTS {$this :family/:kingdom ?o} &&
          !EXISTS {$this :phylum/:kingdom ?o} &&
          !EXISTS {$this :genus/:kingdom ?o} )}
""" ;
] .

:isCitizenOfShape a sh:NodeShape ;
sh:targetSubjectsOf :isCitizenOf ;
sh:sparql [
 a sh:SPARQLConstraint;
 sh:select """
  SELECT $this ?o WHERE {
   $this :isCitizenOf ?o .
   FILTER(
    !EXISTS {?o a :Country} ||
    (!EXISTS {$this :wasBornIn/:isLocatedIn ?o} &&
     !EXISTS {$this :graduatedFrom/:isLocatedIn ?o}
   ))}
""" ;
] .
```

The `:isCitizenOfShape` constraint learned on YAGO3 requires the object to be of type `:Country` and the subject to be born in a place located in or graduated from an institution located in the same country. The constraint is not entirely correct, since people who were not born in or did not graduate in a given country can still be citizen of that country. However, it reveals interesting patterns in the data. Moreover, by varying the minimum confidence threshold one can obtain more aggressive constraints, such as the one shown above, or more conservative ones which do not require the paths conditions to be fulfilled.

The `:kingdomShape` exploits the fact that for every level of the life taxonomy below kingdom (from species to phylum), most instances have assertions of the kingdom relation. The constraint requires that for every pair of subject-object at least one of following three paths should exist: `:family/:kingdom`,

`:phylum/:kingdom` and `:genus/:kingdom`. The problem is that while this holds for the majority of the `:kingdom` assertions, those which have a phylum as subject cannot have one of the three aforementioned paths because phylum is the level directly under kingdom. Statistical methods – including our approach – identifies such case as outlier, since the proportion of subjects which are phyla is very small. This happens because there are orders of magnitude more species, genera, families, orders and classes than phyla.

This case illustrate the importance of having readable constraints, which can be understood and improved by specialists. The constraint could be easily fixed by adding the path `^:phylum/:kingdom` to the expression, which would include the cases where the subject is a phylum into the definition.

One of the limitations of our approach is the cost of considering paths of length $mpl > 2$ on datasets with many relations. In order to enable PaTyBRED to be used on large-scale datasets, such as DBpedia and NELL, conservative values for mpl and $mppl$ need to be selected. This reduces the number of paths whose adjacency matrix needs to be computed and the number of features considered in the relations' training data. This improves the scalability, however, it also means that relevant paths can be possibly left out.

Another limitation is that in its current implementation, PaTyBRED generates negative examples by substituting the subject or object by a randomly selected entity. Since the distribution of instances over classes on most KGs is highly skewed, with some classes being much more likely to be sampled than others. That means the generation of potentially relevant negative examples with instances of infrequent classes is unlikely, which may make it difficult to learn constraints with such infrequent classes.

In order to compensate for this effect, we would need to introduce a bias to selection of entities on the generation of negative examples. A possible solution is to make it more likely to generate instances of the same or sibling classes, making it more likely to select entities of classes that are more closely related to the class of the original entity. That is an interesting problem, however it requires extensive research in order to verify its effectiveness on mitigating the issue.

8.4 Conclusion

In this chapter we propose a method for learning SHACL-SPARQL constraints for relations which is based on the relation assertion error detection method PaTyBRED. We compare the learned SHACL constraints with RDFS domain and range restriction learned with statistical schema induction. We performed a manual comparison of the two approaches on DBpedia, and we show that our SHACL constraints are better at detecting wrong relation assertions while being more robust when handling noise and incompleteness of subject and object type assertions.

Chapter 9

Inductive Lexical Learning of Class Expressions

9.1 Introduction

There has recently been an increase in the number and size of RDF knowledge bases, in particular in the context of the Linked Open Data initiative. However, there is still a lack of knowledge bases that use expressive ontologies and instance data structured according to those ontologies. Many datasets focus on instance data and give less attention to the ontological layer. One of the reasons for this is the effort required to build up an ontology. To address this problem, a multitude of approaches have been devised using a plethora of methods (108). In particular, there have been two main branches of research: On the one hand, lexical ontology learning approaches aim at constructing ontologies from textual input (116) and, on the other hand, logical learning approaches use existing RDF data as input to construct ontologies (79; 25). In this work, we present the first algorithm, we are aware of, which combines lexical and logical ontology learning. This constitutes the first step on a larger research agenda aiming to improve ontology learning algorithms to a state in which they achieve sufficient precision and recall to be employed in practice. Previous studies have shown that current algorithms have not yet achieved this goal (c.f. Lehmann et al. (102)) and ontology learning remains an extremely challenging problem.

Using a short example, we briefly want to illustrate how schemata improvements can enable more powerful reasoning, consistency checking, and improved querying possibilities. In particular, in this article we are concerned with learning $\mathcal{EL}$ description logic concepts for definitions.

Example 1. *The following definition in description logic syntax was learned by our approach for the class* `Astronaut`[1] *in DBpedia (106).*

[1] We omit the namespace `http://dbpedia.org/ontology/` for readability

$$\begin{aligned} \mathtt{Astronaut} \equiv \mathtt{Person} \sqcap \exists \mathtt{mission}.\mathtt{SpaceMission} \\ \sqcap \exists \mathtt{timeInSpace}.\mathtt{minute} \end{aligned}$$

The definition states that a person who was on a space mission and spent time in space is an astronaut and vice versa. Adding this definition to an ontology can have the following benefits: 1.) It can be used to detect inconsistencies and quality problems. For instance, when using the Pellet Constraint Validator[2] on a knowledge base with the above axiom, it would report astronauts without an associated space mission as violation.[3] 2.) Additional implicit information can be inferred, e.g., in the above example each person, who was on a space mission and spent time in space can be inferred to belong to the class Astronaut, which means that an explicit assignment to that class is no longer necessary. 3.) It can serve as documentation for the purpose and correct usage of schema elements. For instance, in the above example it can be argued that someone is an astronaut if he is trained for a space mission, whereas the definition requires to actually take part in such a mission. The definition clarifies the intended usage. Overall, we make the following contributions:

- first approach to combining logical and lexical ontology learning
- analysis of statistical relevance measures for learning class expressions
- a manual evaluation on a realistic large scale data set

The adapted algorithm is called *ELTL* ($\mathcal{EL}$ Tree Learner) and part of the open-source framework DL-Learner[4] (101) for concept learning in description logics (DLs). The remainder of the chapter is structured as follows: Section 9.2 covers preliminaries such as a definition of the learning problem in logical ontology learning and a description of the base algorithm we use. Subsequently, in Section 9.3, we describe how statistical relevance measures applied on textual resources can be integrated into the logical learning framework. Section 9.4 describes experiments and insights obtained from them and we conclude in Section 9.5.

9.2 Preliminaries

For an introduction to OWL and description logics, we refer to (8). In this section, we focus on giving an overview of the base learning algorithm we draw on. The task we investigate resembles *Inductive Logic Programming* (140) using a description logic knowledge base as background knowledge and $\mathcal{EL}$ concepts as target language. In the ontology learning problem we consider, we learn a definition of a class A, which has (inferred or asserted) instances in the considered ontology.

[2] http://clarkparsia.com/pellet/icv/

[3] Under OWL semantics, this is not a violation, due to the Open World Assumption, unless we can infer from other knowledge that the person cannot have taken part in a mission

[4] http://dl-learner.org

To define the class learning problem, we need the notion of a *retrieval* reasoner operation $R_\mathcal{K}(C)$, which returns the set of all instances of C in a knowledge base $\mathcal{K}$.

Definition 1 (class learning problem)**.** *Let an existing named class A in a knowledge base $\mathcal{K}$ be given. Analogous to standard information retrieval, the F-Score of an $\mathcal{EL}$ concept C is computed based on precision on recall where the precision is defined as $\frac{|R_\mathcal{K}(C) \cap R_\mathcal{K}(A)|}{|R_\mathcal{K}(C)|}$ and recall as $\frac{|R_\mathcal{K}(C) \cap R_\mathcal{K}(A)|}{|R_\mathcal{K}(A)|}$. The goal of the* class learning problem *is to maximize F-Score wrt. A.*

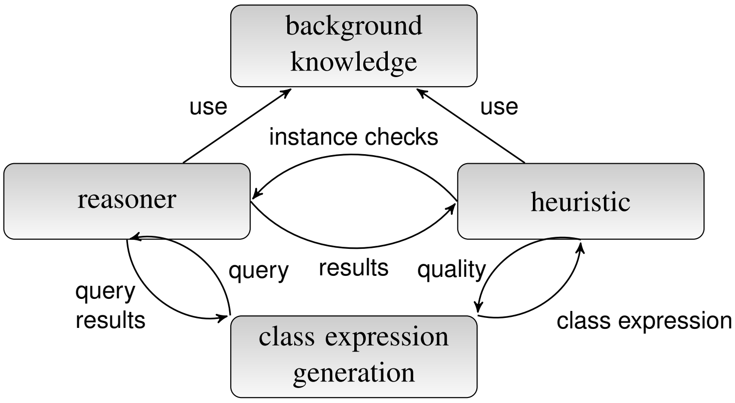


Figure 9.1: Outline of the general learning approach in ELTL: Class expressions taking the available background knowledge into account are generated and evaluated in a heuristic with respect to the target learning problem. Figure adapted from (77).

Figure 9.1 gives a brief overview of our base algorithm *ELTL* ($\mathcal{EL}$ Tree Learner), which follows the common "generate and test" approach in ILP. This means that learning is seen as a search process and several class expressions are generated and tested against a background knowledge base. Each of those class expressions is evaluated using a heuristic, which we will analyze later in more detail.

Definition 2 (refinement operator)**.** *A* quasi-ordering *is a reflexive and transitive relation. In a quasi-ordered space $(S, \preceq)$ a* downward (upward) refinement operator *ρ is a mapping from S to 2^S, such that for any $C \in S$ we have that $C' \in \rho(C)$ implies $C' \preceq C$ ($C \preceq C'$). C' is called a* specialization *(*generalization*) of C.*

Refinement operators can be used for searching in the space of expressions. As ordering we can use subsumption ($\sqsubseteq$), which is a quasi-ordering relation. If an expression C subsumes an expression D ($D \sqsubseteq C$), then C will cover all examples which are covered by D. This makes subsumption a suitable order for searching in expressions as it allows to prune parts of the search space without losing possible solutions.

The approach we used is a top-down algorithm based on refinement operators as illustrated in Figure 9.2. This means that the first class expression which will

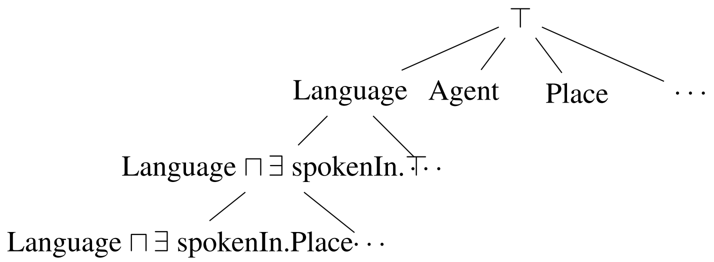


Figure 9.2: Illustration of a search tree in ELTL.

be tested is the most general expression ($\top$), which is then mapped to a set of more specific expressions by means of a downward refinement operator. Naturally, the refinement operator can be applied to the obtained expressions again, thereby spanning a *search tree*. The search tree can be pruned when an expression does not cover sufficiently many instances of the class A we want to describe.

The heart of such a learning strategy is to define a suitable refinement operator and an appropriate search heuristics for deciding which nodes in the search tree should be expanded. The refinement operator in the ELTL algorithm is defined and evaluated in (103) and has several beneficial theoretical properties not further detailed here.

9.3 Approach

The learning algorithms in the DL-Learner framework are designed by combining a refinement operator with a search heuristic. While the operator itself is *ideal* with respect to a set of beneficial theoretical properties as shown in (103), we are investigating and improving the learning algorithm by incorporating a more intelligent search heuristic.

Learning concepts in Description Logics is a search process. The refinement operator is used for building the search tree, while a heuristic decides which nodes to expand. This decision can be done based on different criteria like the number of positive and/or negative examples covered by the class expression or the length of the concept represented by the node. While the existing heuristics basically rely on scores based on metrics according to coverage of the examples obtained via logical inference, we have observed in previous experiments (102; 25) that best coverage does not always result in the most intuitive class expressions, and the search process can be improved by taking into account information contained in textual resources. Therefore, we extract statistical information out of given text, which might give some insights on the relevance of other ontology entities for the class we want to describe. This idea is substantiated by the assumption that words which are more related to each other tend to co-occur more often in texts as also expressed in the famous statement to “know a word by the company it keeps” (61) as frequently quoted throughout the linguistics community. This relevance score can then be combined with the other metrics in the heuristic and, thus, influence the navigation in the search tree, which in the end can result in better class descriptions.

Apart from the novel idea of including those relevance measures, one of our main goals was to evaluate which measure is suitable. In order to measure the relevance of entities for the definition of a given entity, we use popular co-occurrence based association measures (32). The measures employed in this chapter are Jaccard, Dice, Semi-conditional Information (SCI), Pointwise Mutual Information (PMI), Log Likelihood Ratio (LLR), Chi-Square (χ^2), T-test, and Significant PMI (sPMI) (45). The first two measures have the advantage of being simple to compute and their values always fall into the $[0,1]$ interval. The latter five measures incorporate some notion of statistical significance, by considering the ratio of observed ($f(x,y)$) and expected frequency assuming independence $Ef(x,y)$ of an entity pair x, y. PMI takes into account only the occurrence probabilities, ignoring the absolute frequency, which results in a tendency to yield high score values for low frequency pairs. sPMI solves this problem by incorporating corpus level significance, which considers the probability of observing a given deviation between $f(x,y)$ and its expected value $Ef(x,y)$. SCI multiplies PMI by the conditional probability $p(y|x)$, which tends to favor highly frequent pairs, therefore compensating the PMI's problem. LLR and χ^2 are the only ones which have the null addition property (32), which means that the measure is affected by the addition of data containing neither x not y.

Apart from influencing the search process in the learning algorithm, the textual evidence included by relevance measures can also influence the final ranking of class expressions. Actually, we have two possibilities of how the learning process can benefit from information contained in texts related to the knowledge base:

External Text Corpus The first option relies on an external text corpus $\mathcal{C}$. We treat each document d in the corpus as a separate context and get information about the occurrence of the class c we want to describe, the occurrences of each other schema entity (a class or property) e_i, as well as the joint occurrences of c with each e_i. The retrieved information is then processed by the chosen relevance measures. The computation of the relevance score for each e_i is done in advance before the learning algorithm starts, because we have to normalize the values (we're doing a min-max normalisation), especially when we're using more than one relevance measure. In the current approach, we check the occurrence of an entity in a text by just taking a human-readable name for the entity[5] and check if it occurs syntactically in the text, i.e., we do not perform any kind of disambiguation which is planned to be integrated in a future version.

Local Textual Information The second option uses textual resources which are contained in the underlying knowledge base, e.g., the individuals could be accompanied by textual descriptions summarizing important facts about them like the birth place of a person or that an astronaut participated in a particular space mission. This information can be used to get the relevance of the schema entities e_i by checking for the occurrences of its labels in the descriptions of instances of

[5] Usually we use `rdfs:label`, but it's of course possible to use any other property.

the class we want to describe.

9.4 Evaluation

9.4.1 Experimental Setup

We performed our experiments on the English DBpedia data set in version 3.9 accessible via a local mirror. The DBpedia data set was extracted from Wikipedia and at its core consists of resources corresponding to Wikipedia articles and facts extracted from article pages. DBpedia provides a lightweight OWL ontology, which defines the different classes and properties used throughout the data set. The DBpedia ontology contained a total of 529 classes of which 423 are leaf classes, i.e., classes not having any non-trivial subclasses. Furthermore, the ontology contained 927 object and 1,406 datatype properties. The ontology is also used for extracting the data from Wikipedia using mappings between infobox templates and ontology classes as described in (106). The extraction process based on this mapping results in the so-called mapping-based data set which we use in our experiments. Overall, our experiments data set contained about 63.5 million triples describing 3,243,481 instances.

On this data set, we used DL-Learner (specifically ELTL) enhanced with the relevance metrics as described above to learn class expressions for the classes contained in the DBpedia ontology. As a corpus for the relevance metrics the abstracts of all Wikipedia articles which described concepts modeled by an ontology class were crawled and then provided for retrieval using a SOLR[6] instance. This corresponds to the “External Text Corpus” scenario described in the Section 9.3. Though we also implemented the second scenario, we opted to only evaluate the first one which is more generally usable because it does not rely on greater amounts of textual descriptions in the data set itself.

We generated class expressions for all leaf classes of the DBpedia ontology which had at least 3 instances (288 out of 423). We then computed a sample set of at most 100 positive examples (instances belonging to the class) and 200 negative examples. For the negative examples, 100 instances belonging to the sibling classes and 100 instances of super classes which were not contained in the class to describe were randomly chosen. After applying DL-Learner, we performed a manual evaluation to find the combination of relatedness measures which resembles the human perception of intuitiveness of a class expression best. To do this, we randomly chose 100 of the classes with at least 10 alternative class expressions generated. For each class with more than 50 class expressions, we picked the top-50 expressions ranked by F-score. We handed the generated lists to four human annotators (two researchers not involved in the research presented in this submission from the Universities of Mannheim and Leipzig, respectively) along with the instruction to mark the class expressions which they consider most suitable for

[6] `https://lucene.apache.org/solr/`, version 4.1.0

#	class expression	F-score	PMI	sPMI
1	Person $\sqcap \exists$ selection.$\top$	0.977	0.662	0.529
3	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ birthPlace.PopulatedPlace	0.960	0.797	0.549
4	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ birthPlace.Place	0.960	0.716	0.518
5	Person $\sqcap \exists$ mission.SpaceMission	0.950	0.493	0.664
8	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ nationality.Country	0.947	0.707	0.498
12	Person $\sqcap \exists$ nationality.Country	0.937	0.697	0.489
13	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ occupation.PersonFunction	0.937	0.672	0.487
15	Person $\sqcap \exists$ timeInSpace.minute	0.933	0.771	0.571
17	Person $\sqcap \exists$ mission.SpaceMission $\sqcap \exists$ timeInSpace.minute	0.933	0.620	0.643
19	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ mission.SpaceMission	0.933	0.584	0.603
21	Person $\sqcap \exists$ mission.SpaceMission $\sqcap \exists$ birthPlace.Place	0.933	0.615	0.599
22	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ nationality.Country $\sqcap \exists$ birthPlace.Place	0.933	0.733	0.499
29	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ birthDate.date	0.923	0.553	0.466
30	Person $\sqcap \exists$ mission.SpaceMission $\sqcap \exists$ occupation.PersonFunction	0.923	0.571	0.568
31	Person $\sqcap \exists$ mission.SpaceMission $\sqcap \exists$ nationality.Country	0.923	0.605	0.579
41	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ birthPlace.PopulatedPlace $\sqcap \exists$ mission.SpaceMission	0.920	0.703	0.596
48	Person $\sqcap \exists$ selection.$\top \sqcap \exists$ nationality.Country $\sqcap \exists$ occupation.PersonFunction	0.917	0.701	0.477

Table 9.1: Excerpt of the 50 class expressions that have been evaluated for the class `Astronaut`. The first column denotes the rank of the DL-Learner output without taking statistical measures into account. Only the class expressions have been shown in random order to the evaluators.

being added to the DBpedia ontology as definitions of the class. Additionally, we explicitly highlighted the possibility of marking multiple class expressions in cases where they are equally suitable or no expression if there was no expression close to an acceptable definition. The evaluation process took two hours per annotator on average. An example of evaluated class expressions for the class `Astronaut` is shown in Table 9.1.

In the second part of the evaluation, we applied several classification approaches to the F-score and relevance measures values to find a combination which is suited to reproduce the human assessment of intuitiveness. For this purpose, we employed the implementations provided by the Weka toolkit (74) in version 3.6.6.

9.4.2 Results

First, we computed the inter-rater agreement using the Fleiss' Kappa (63) statistical measure to get a score of how much homogeneity, or consensus, there is in the ratings given by judges. We evaluated the agreement on two different levels in terms of granularity. On the class level we expect to have an agreement if the evaluators selected at least one class expression to be useful as definition for the corresponding class. Here we got a Fleiss' Kappa value of 0.51 which can be interpreted as "moderate agreement" (96). On the more fine-grained class expression level, we assume to have an agreement if the same class expression was selected as an appropriate class definition. The Fleiss' Kappa value was approximately 0.28 which can be seen as a "fair agreement". For the 288 classes processed by DL-Learner on average 51 class expressions have been generated. The average length of the expressions, which is defined in a straightforward way, namely as the sum of

algorithm	accuracy	F-score	AUC
C4.5	77.5%	77.1%	79.6%
SVM	73.3%	74.6%	73.3%
Logistic Regression	72.8%	73.3%	79.9%
Conjunctive Rule	69.5%	67.9%	72.5%
Naive Bayes	64.1%	54.3%	75.6%
ELTL Baseline	59.4%	61.7%	63.8%

(a) Results of 10-fold cross-validation for different classifiers.

feature added	accuracy
T-test	61.3%
+ F-score	73.5%
+ LLR	77.3%
+ Jaccard	77.0%
+ PMI	78.0%
+ χ^2	78.1%
+ SCI	78.5%

(b) Accuracy gain for features using C4.5.

Table 9.2: Results of relevance measure analysis.

the numbers of concept, role, quantifier, and connective symbols occurring in the expression was ≈ 10.
Our experiments address the following research questions:

1. Which relevance measures are particular suitable, and how should they be combined?

2. Can a combination of statistical relevance measures improve the results of logical ontology learning?

In order to answer the first question, we cast this task as a supervised machine learning problem itself in which F-score and the presented relevance measures are features of a learned definition. A definition is then considered to be a positive example if an evaluator selected it in our experiment and negative otherwise. Since this leads to a skewed distribution with more negative examples, we applied random subsampling on the negative examples. This results in an equal distribution of 302 positive and negative examples. In a first step, we used these to obtain a suitable classifier. We ran different types of classifiers (see Table 9.2a), i.e., support vector machines, decision trees, rules and probabilistic classifiers, as implemented in the Weka toolkit[7], with their default settings and used 10-fold cross-validation. As a baseline, we used an optimal threshold for the F-Score, which was determined by the Weka threshold selector meta classifier. An interesting insight is that the inclusion of relevance measures indeed significantly improves the standard approach of computing F-Score on the underlying RDF data, which allows us to positively answer the first research question.

The C4.5 decision tree algorithm performed best, so we used it as a base for feature analysis. This analysis was performed by using standard wrappers for feature subset selection (89). In this case, we could exhaustively run all combinations of features in C4.5 via 10-fold cross-validation and optimizing for predictive accuracy. The best performing feature subset is {F-score, PMI, χ^2, Jaccard, LLR, SCI,

[7] `http://www.cs.waikato.ac.nz/ml/weka/`

T-test}. We used this subset and iteratively removed the feature which caused the least loss in predictive accuracy. This allows us to observe the increase in accuracy obtained by adding features as shown in Table 9.2b:

The first three features led to significant improvements in the ability to detect promising definitions whereas the other features showed only small contributions (even negative in one case). We also analyzed the weights of normalized features in the SVM classifier:

$$\begin{array}{llll}
3.6477 \times \text{F-score} & -2.2027 \times \text{PMI} & -0.1476 \times \chi^2 & \\
+0.7601 \times \text{Dice} & +0.8325 \times \text{Jaccard} & -0.4517 \times \text{LLR} & \\
+0.2963 \times \text{SCI} & +3.7772 \times \text{sPMI} & +1.1387 \times \text{T-test} & -4.5916
\end{array}$$

It is notable here that PMI indeed has a negative weight. We believe this is due to high PMI values for low frequency entity pairs, so the negative weight along with the high positive weight of sPMI essentially acts as a noise filter. χ^2 and LLR also have negative weights, which might be related to their null addition property. We also noted that some metrics have very low values close to zero in the majority of cases and are essentially only used a tie breaker in the SVM classifier whereas the C4.5 decision tree can make better use of those values.

9.4.3 Discussion

During the manual annotation of the created class expressions, the annotators did not find suitable definitions for the classes in a number of cases. Based on the comments provided by the annotators and manual inspection of the affected classes, we were able to find patterns helping to categorize the problems. In the following, we describe the categories and give examples for each.

Limited Ontology Vocabulary This problem arises due to relying on the classes and properties defined in the DBpedia ontology. In these cases, the ontology does not provide any entities which could be used to describe the class both accurately and exhaustively. For example, the expressions generated for describing the class `Bodybuilder` contained properties as `height` together with the class `Athlete`. Obviously, this is a correct description of a bodybuilder but it also matches all other athletes since the "restrictions" are actually properties of the parent class. However, DL-Learner was unable to choose a better definition since the ontology did not contain single properties or combinations of these specifically able to describe a body builder. This type of problems could only be solved by manually adding more specific properties or classes.

Limited Usage of Vocabulary A related problem arose when the ontology contained an entity usable to describe a class fully which was not created by DL-Learner. For instance, when describing the concept `CanadianFootballLeague`, DL-Learner created definitions like `SportsLeague ⊓ ∃team.SportsTeam` that describes a sports league but is not specific enough to exclude sport leagues other than Canadian football league. Replacing `SportsTeam` in the definition by CanadianFootballTeam would lead to a flawless definition but is not proposed by DL-Learner. This is because the positive examples do not contain a significant number of assignments of teams to a Canadian football league that are also asserted to be Canadian football teams.[8] Again, this problem is hardly solvable when learning expressions but only at the data level.

Superfluous Restrictions Some class expressions were also not chosen by the annotators because they contained superfluous restrictions. This is most often the case when defining subclasses of Person like Writer and including restrictions on, e.g., birthPlace. Clearly, this is not a restricting property for writers all being persons. Thus, some of these definitions were not chosen by the annotators who tried to choose the definitions as compact as possible. Most probably, this problem is also caused by the missing vocabulary to describe some classes suitably. We could try to prevent the generation of such definitions by considering the domain and ranges of properties and filtering restrictions if properties are defined for super classes of the currently considered class. However, then we would depend more strongly on the correctness of the schema whose quality showed to be doubtful in many cases throughout our experiments.

Another interesting example is the definition of an Architect, which uses the property significantBuilding though from the word meaning this would not be a definition covering all architects but only the more renowned ones which not only have regular buildings but also significant ones. DBpedia, as it is derived from Wikipedia, contains only few data on architects which are not famous for their buildings. A different but less general problem was discovered for classes belonging to the biological taxonomy in DBpedia. Here, some generated wrong definitions pointed to flaws in the usage of biology-specific properties like kingdom.

In summary, we discovered a combination of measures for generating more intuitive class expressions. From the inclusion of textual information, we were able to complement the purely logical information employed by DL-Learner with additional knowledge about how related specific properties are evaluated by humans. Most problems detected during the manual annotation can be traced back to missing or underspecified input data.

9.5 Conclusion

In this chapter, we presented first steps towards combining the previously distinct logical and lexical ontology learning areas. By extending a formerly pure logic based approach with statistical methods which can be used on text corpora, we were able to foster the generation of more intuitive class expressions. An extensive manual evaluation with four human annotators showed that the integration of relevance measures can significantly improve results. Nevertheless, we also discovered and analyzed several problems for which we were partially able to trace them back to data quality issues.

[8] Only 2 of 10 teams are assigned to be a Canadian football team.

Chapter 10

Learning Rules With Numerical Attributes

10.1 Introduction

Discovering patterns by learning rules from knowledge or data bases enables us to obtain concise descriptions of a domain, to predict new facts and to detect anomalies. Especially by considering numerical values such as birth dates or measurements, which are common in many data sets, we are often able to unveil interesting patterns (e.g. spatial or seasonal). We illustrate the benefits and challenges of learning rules with numerical attributes by an example from the U.S. Census[1] dataset, where we would like to discover rules describing the marital status of a person with a minimum confidence threshold of 0.7. The rules shown in Table 10.1 do not satisfy the given threshold, but we can significantly increase their confidence values if we restrict the numerical *age* variable Y to specific intervals.

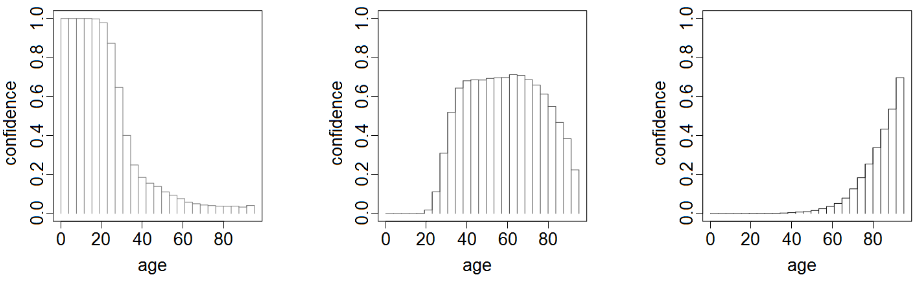
id	rule	conf
r_1	*maritalStatus(X,single) :- age(X,Y)*	0.40
r_2	*maritalStatus(X,married) :- age(X,Y)*	0.46
r_3	*maritalStatus(X,widowed) :- age(X,Y)*	0.06

Table 10.1: Example of rules without numerical intervals

In particular, by observing how the confidence values are distributed over the *age* variable Y (cf. Figure 10.1), we notice that for the numerical intervals $[0, 30]$, $[30, 80]$ and $[90, \infty)$ the confidences are much higher. In other words, the figure shows that younger people are likely to be single (10.1a), middle-aged married (10.1b), and elderly widowed (10.1c).

Given this insight, we can refine the rules r_1, r_2 and r_3 by restricting Y to the aforementioned intervals, and obtain rules which satisfy the confidence threshold

[1] U.S. Census (2000): `http://www.rdfabout.com/demo/census/`



(a) r_1 (b) r_2 (c) r_3

Figure 10.1: Confidence distribution over age Y

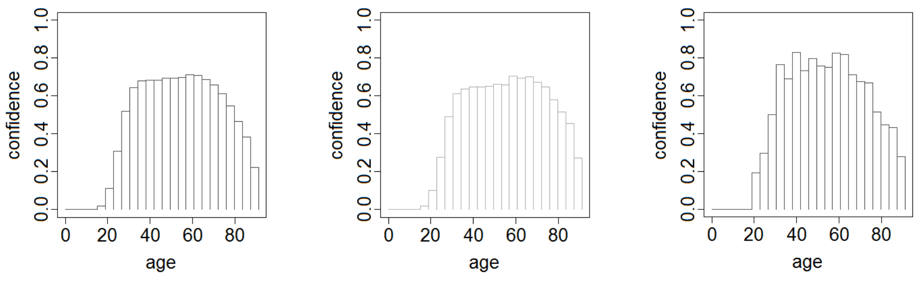
(cf. Table 10.2).

id	rule	conf
r_4	*maritalStatus(X,single) :- age(X,Y),Y* $\in [0,30]$	0.91
r_5	*maritalStatus(X,married) :- age(X,Y),Y* $\in [30,80]$	0.72
r_6	*maritalStatus(X,widowed) :- age(X,Y),Y* $\in [90,\infty)$	0.71

Table 10.2: Example of rules refined with numerical intervals

However, note that if we had attempted to learn a concept which is uncorrelated to the attribute *age*, such as *quarterOfBirth*, a costly search for interesting *age* intervals would have been useless. This is because for every quarter of the year the confidence distribution over Y is practically uniform, as seen in the Figure 10.3 example for the first quarter case (*quarterOfBirth(X,q1) :- age(X,Y)*). Hence, there is no interval for Y that yields significant confidence gain when compared to the overall Y domain. Generally, it is desirable to avoid the computation of intervals for uninteresting combinations of numerical and categorical attributes, since the queries to obtain the support and confidence distributions are expensive and the search space can easily become too large. So, *before* running these expensive queries for a rule with unrestricted numerical attributes variables, we would like to estimate the likelihood that the rule has a confidence distribution with interesting intervals.

This interestingness estimation can also be used for choosing better categorical refinements. For example, we could further extend rule r_2 (cf. Table 10.1) by also considering the U.S federal state a person lives in. We observe that the confidence distribution of rule r_2 refined with *Florida* (Figure 10.2b) is very similar to the confidence distribution of the more general rule r_2 (Figure 10.2a), whereas for *South Dakota* (Figure 10.2c) the distribution is less uniform and yields higher confidence values. Therefore, we are more likely to discover interesting rules if we choose *South Dakota* instead of *Florida* as a refinement of the base rule.



(a) USA (b) Florida (c) South Dakota

Figure 10.2: Confidence distribution of rule r_2 over the *age* attribute for the overall USA population (a) as well as refined by the states Florida (b) and South Dakota (c)

In this chapter, we propose an efficient approach to mining datalog rules with numerical and categorical attributes which employs information theory techniques to speed up the ILP algorithm. We estimate the likelihood of any given rule to have an interesting refinement in the form of an interval restriction on the numerical attribute. This way of measuring a rule's (potential) interestingness enables us to efficiently explore the search space, rank the rules, and prune uninteresting rules (e.g. *quarterOfBirth(X,q1)* :- *age(X,Y)*). Our approach comprises a preprocessing phase for computing the correlations between numerical and categorical attributes, as well as an extension to the ILP refinement step. We demonstrate the feasibility of our approach by means of experiments with three large-scale datasets (U.S. Census data, Freebase and DBpedia).

10.2 Problem Definition

Firstly we need to introduce the notions of *base rules* and *refined rules*:

Definition 1. *A Base Rule is a rule whose body contains a free numerical attribute variable.*

Definition 2. *A Refined Rule is a base rule with the numerical attribute variable restricted to some interval.*

For example, the rules r_1, r_2 and r_3 are examples of base rules, while r_4, r_5 and r_6 are their refined counterparts. Note that we only consider base rules with a single numerical attribute. Our goal is to learn refined rules which yield significant confidence gain over their corresponding base rules.

Definition 3. *Confidence Gain is the ratio of the confidence of a refined rule to its base rule ($conf(r_{ref})/conf(r_{base})$).*

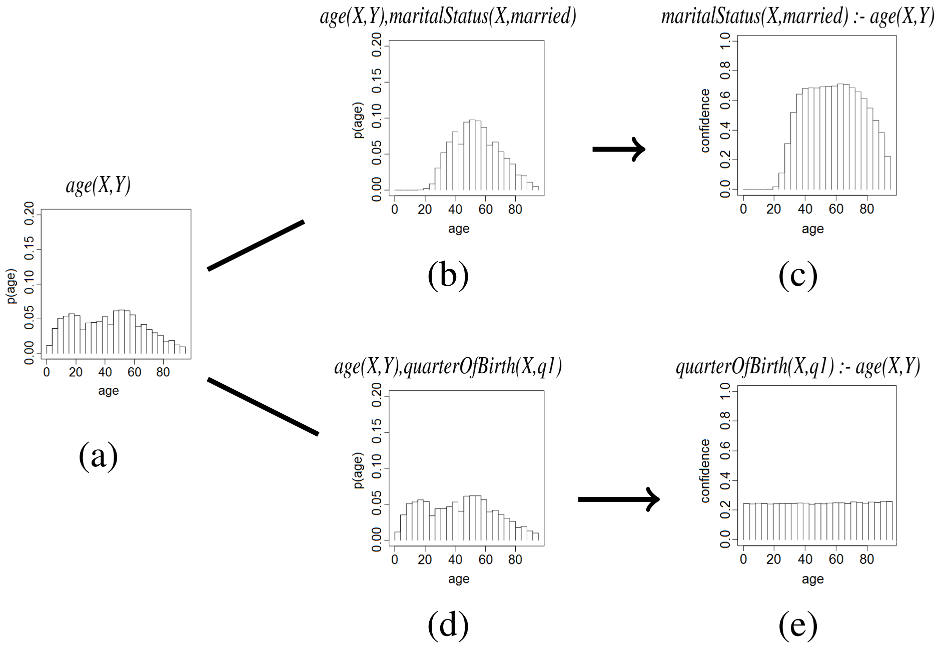
The confidence of a rule h *:-* b (with h being the head literal and b being the body clause of the rule) is defined as $conf(h\text{:-}b) = supp(\{h, b\})/supp(b)$. Its support is given by the number n^+ of covered positive examples, i.e., $supp(h\text{:-}b) = n^+(\{h, b\})$.

We consider a base rule *interesting* if it has a refined rule that satisfies the minimum support, confidence and confidence gain thresholds. As we want to efficiently explore the datalog base rules search space in ILP, our goal is to predict the interestingness of a base rule before computing its confidence distribution.

10.3 Interestingness Measure

We are interested in base rules with non-uniform confidence distributions of low entropy, which can result in interesting refined rules with high confidence gain. This kind of confidence distribution is the result of divergent support distributions of a rule's body and positive examples.

In Figure 10.3 we compare the rules *quarterOfBirth(X,q1) :- age(X,Y)* and *maritalStatus(X,married) :- age(X,Y)* which share the same body. As we see in Figures 10.3a and 10.3d, the first rule's body and positives distribution are practically the same resulting in an uniform confidence distribution (10.3e), while for the latter rule, its divergent positives distribution (10.3b) leads to a more interesting confidence distribution (10.3c).



(a)

(b)

(c)

(d)

(e)

Figure 10.3: Example of a body support distribution (10.3a) with different positives distributions (10.3b) and (10.3d)

The interestingness of a rule can be measured by computing the divergence of the $\{body\}$ and $\{body, head\}$ support distributions. Hence, we discretize the domain of Y into k disjoint buckets $b_1, \ldots, b_k$, in order to obtain the support histogram of a clause c, which is defined as:

$$h(c) =< h_1(c), \ldots, h_k(c) >,$$
$$\text{where } h_i(c) = supp(c|Y \in b_i) \text{ and } |h(c)|_1 = supp(c)$$

from which we obtain the probability density distribution of support $f(c) = h(c)/supp(c)$, where $|f(c)|_1 = 1$.

Based on this we define the interestingness measure as $I_Y(l|c)$, where l is the literal to be added to the clause c, and Y is the free numerical attribute variable we want to find an interval for. To compute I_Y we use a divergence measure D compare the distributions of c and$\{c, l\}$ over Y :

$$I_Y(l|c) = D(f(c)||f(\{c, l\})) \tag{10.1}$$

Because of sampling error, this divergence measure tends to yield higher values for clauses with lower support, and thus favors them over higher support clauses. In order to compensate this effect, and because we are also interested in rules with high support, we propose a hybrid interestingness measure combining divergence and support:

$$I_Y(l|c) = supp(\{c, l\}) * D(f(c)||f(\{c, l\})) \tag{10.2}$$

In our implementation, we use Kullback-Leibler (94) as the divergence measure D, and we apply Laplace smoothing to all distributions prior to the divergence value calculation in order to remove the zeros.

10.4 Correlation Lattice

In a preprocessing step, we create a correlation lattice for each numerical property. The purpose of the lattice is to model the correlations between a given numerical property and different categories, which in our implementation are defined by categorical literals.

A correlation lattice is structurally similar to an itemset lattice, introduced by (1) . It describes the correlations between a numerical property and multiple categorical attributes. The target numerical property with a join variable X and a free numerical variable Y is used as the root literal, and the categorical literals correspond to the items in the itemset lattice. All of the categorical literals are joined by the join variable X, forming in each node n a unique clause c_n. Figure 10.4 shows an example of lattice for the attribute *age(X,Y)*.

Additionally, every node n has a corresponding histogram $h(c_n)$ with c_n support distribution over Y. Therewith, all the edges between a parent node and a child node have an associated interestingness value of adding the child's new literal to the parent's clause. Since we need to compare the support distributions, the bucket

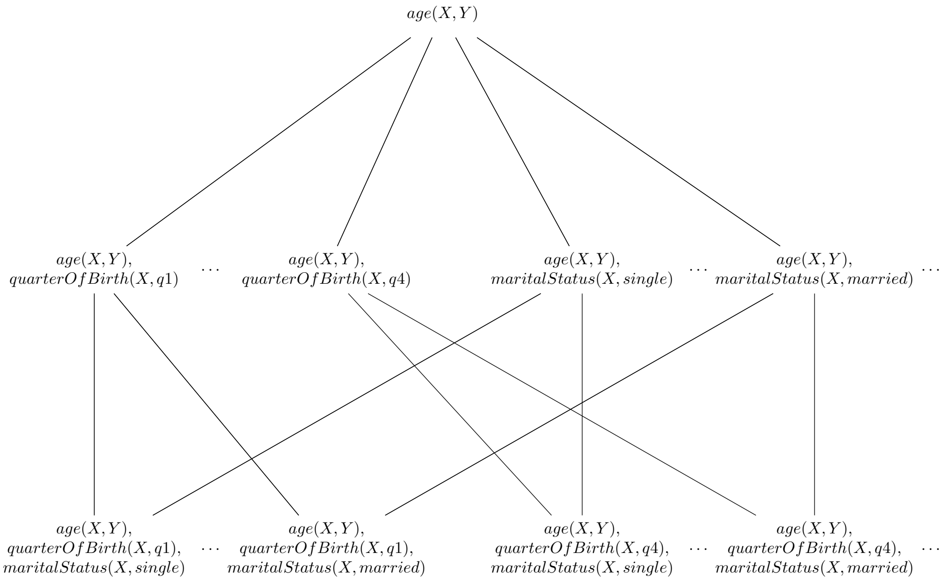

$age(X,Y)$
$age(X,Y)$, $quarterOfBirth(X,q1)$
$\cdots$
$age(X,Y)$, $quarterOfBirth(X,q4)$
$age(X,Y)$, $maritalStatus(X,single)$
$\cdots$
$age(X,Y)$, $maritalStatus(X,married)$
$\cdots$
$age(X,Y)$, $quarterOfBirth(X,q1)$, $maritalStatus(X,single)$
$\cdots$
$age(X,Y)$, $quarterOfBirth(X,q1)$, $maritalStatus(X,married)$
$age(X,Y)$, $quarterOfBirth(X,q4)$, $maritalStatus(X,single)$
$\cdots$
$age(X,Y)$, $quarterOfBirth(X,q4)$, $maritalStatus(X,married)$
$\cdots$

Figure 10.4: Example of correlation lattice for the numerical attribute *age*

boundaries of the histograms must be consistent through the whole lattice. These boundaries are defined in the root node based on the overall population. In our implementation, the number of buckets is arbitrarily defined, and we use equal frequencies as discretization method, because of its robustness with regard to skewed distributions.

The correlation lattice is built with an Apriori-based algorithm, similar to that introduced by (2) , which takes advantage of the anti-monotonicity property of the support of the clauses in the lattice. Thus, it is possible to safely prune the nodes which do not satisfy the support threshold. We also limit the number of levels in the lattice to d_{max}, which is upper bounded by the maximum number of literals in clauses allowed in the ILP algorithm.

10.4.1 Independence Test

With the information contained in the lattice, we can test whether two categorical literals are independent given a certain clause. Let us assume we have two nodes n and m with common parent p, and common child q, with clauses c_p, $c_n = \{c_p, l_n\}$, $c_m = \{c_p, l_m\}$ and $c_q = \{c_p, l_m, l_n\}$, where l_m and l_n are literals not contained in the clause c_p. It is possible to estimate the histogram $\hat{h}$ of a node q, assuming that l_n and l_m are conditionally independent given c_p, as $\hat{h}(c_q) = h(c_n)h(c_m)/h(c_p)$. With the estimated $\hat{h}(c_q)$ and the observed actual $h(c_q)$, we can perform a Pearson's chi-squared test of independence.

Figure 10.5 shows an example where we check whether *maritalStatus(X,single)* and *employmentStatus(X,unemployed)* are independent. There the test yields a p-value of 0.96, which means that we have a high confidence that the two literals are conditionally independent.

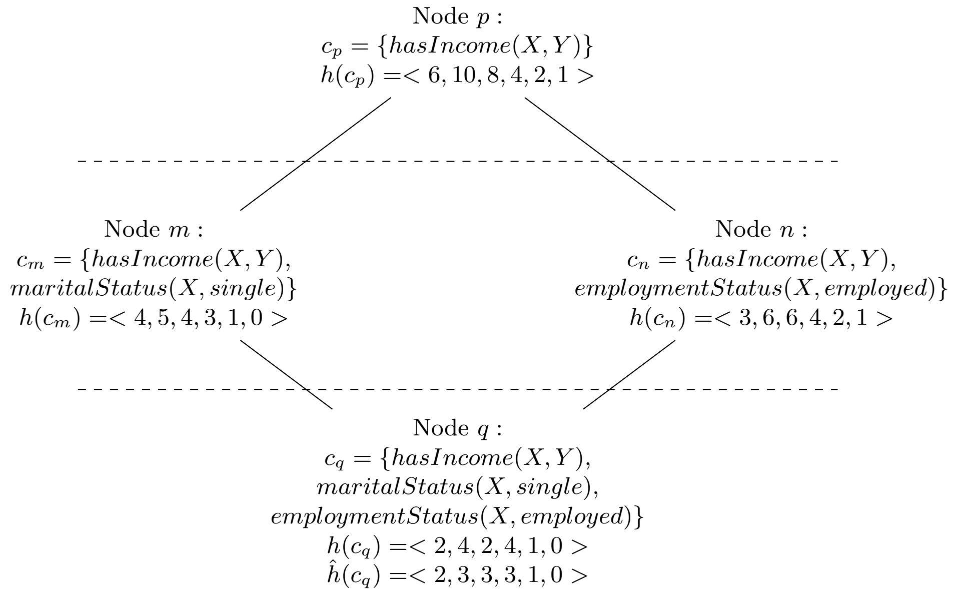

Node p :
$c_p = \{hasIncome(X, Y)\}$
$h(c_p) =< 6, 10, 8, 4, 2, 1 >$
Node m :
$c_m = \{hasIncome(X, Y),$
$maritalStatus(X, single)\}$
$h(c_m) =< 4, 5, 4, 3, 1, 0 >$
Node n :
$c_n = \{hasIncome(X, Y),$
$employmentStatus(X, employed)\}$
$h(c_n) =< 3, 6, 6, 4, 2, 1 >$
Node q :
$c_q = \{hasIncome(X, Y),$
$maritalStatus(X, single),$
$employmentStatus(X, employed)\}$
$h(c_q) =< 2, 4, 2, 4, 1, 0 >$
$\hat{h}(c_q) =< 2, 3, 3, 3, 1, 0 >$

Figure 10.5: Example of a test for independence between the literals *maritalStatus(X,single)* and *employmentStatus(X,unemployed)*

Detecting independence is important because, when we refine a clause in the specialization loop by adding a new literal, the confidence distribution is not likely to change significantly, if this new literal is conditionally independent of the head given the body. For example, if we refine l_m:-c_p by adding l_n to the body, the confidence distribution does not change significantly because of the conditional independence ($P(l_m|c_p) \approx P(l_m|c_p, l_n)$). Therefore, it is useful to set a maximum p-value threshold in order to define and detect independence. Note that for deeper levels in the lattice, a particular node can be generated from different pairs of joining nodes, therefore we cannot prune the conditionally independent nodes. Instead, we mark the two edges joining the conditionally independent nodes as independent, and take that into account when refining a clause in order to prune a conditionally independent refinement.

10.4.2 Scalability

Building a complete lattice can become an unfeasible task even with the apriori pruning and the maximum depth. In this case the lattice with k categories can have up to $\sum_{i=1}^{d_{max}} \binom{k}{i}$ nodes. As employing a very large k would be prohibitive, we suggest to use the interestingness measures introduced in the previous section as pruning heuristics. We also evaluate their performance in the experiments section.

Although the lattice construction can take a significant amount of time, it is important to point out that the clauses contained in the lattice comprise a portion of the ILP search space. The information in the lattice can be reused in the core ILP algorithm preventing the execution of unnecessary queries. In addition, we can directly extract rules of the form:

$$h(X, k_h) \text{ :- } b_1(X, k_{b_1}), ..., b_n(X, k_{b_n}), r(X, Y), Y \in [l, u]$$

where r is the root numerical property, l and u are the lower and upper boundaries of Y, $h(X, k_h)$ and $b_1(X, k_{b_1}), \ldots, b_n(X, k_{b_n})$ are the categorical literals, with $n < d_{max}$. Rules of this form are semantically equivalent to association rules with numerical attributes, with every instantiation of X considered as a transaction.

The correlation lattice can be easily extended to incorporate more complex categories by adopting a broader definition of categories. If we use categorical clauses as defined in (119) instead of categorical literals, we are able to access categories which are not directly connected to the lattice join variable X. This allows us to, for instance, incorporate categories from linked datasets by using the *owl:sameAs* property as a linking relation, enabling us to add cross-domain information to the lattice. However, the problem also becomes more challenging as the number of categories considered and joins required can dramatically increase.

10.5 ILP Algorithm

In order to take advantage of the proposed preprocessing when learning datalog rules with numerical attributes, we need to extend the core ILP algorithm. In the specialization loop, whenever we detect a base rule whose numerical attribute has a correlation lattice, we query the lattice in order to check whether it is relevant to consider refining it with numerical intervals.

Algorithm 4 first checks whether a clause is a base rule, then for each base rule we query the lattice to obtain its interestingness value (cf. Algorithm 5). Subsequently, if the interestingness value satisfies the minimum threshold, we search for the numerical interval (cf. Algorithm 6), where the support and confidence distributions are queried.

These distributions are then analyzed by an algorithm which searches for interesting intervals which is represented by the *getInterestingIntervals* function. Note that for the experiments described further below, we did not search for an *optimal* interval as proposed by (21) , for example, but we only checked if there exists *any* interesting interval. This is sufficient, in order to find out whether our interestingness predictions are correct. Finally, we add the refined rules with the optimal intervals to the refinement graph and continue the ILP algorithm normally.

10.6 Experiments

For our experiments we developed a Java-based implementation of the proposed ILP extension, which uses RDF3X (135) as a knowledge base backend. First

Algorithm 4 Numerical Attribute Interval Refinement

```
for literal $l$ in the body of the clause $c$ do
    if $l$ is numerical and $l$ has free numerical attribute variable and $\exists$ a lattice $lattice_l$ for $l$'s property then
        if GETINTERESTINGNESS($c$,$lattice_l$) $\geq min_{interest}$ then SEARCHNUMERICALINTERVALS($c$,$l$);
```

Algorithm 5 getInterestingness(c_{base},$lattice_l$)

```
$Y \leftarrow$ numerical variable of $l_{num}$
$b \leftarrow$ root node of $lattice_l$
for $l_i \in$ body of $c_{base}$ do
    if $l_i \neq l_{num}$ and $l_i$ joins $l_{num}$ and $l_i$ is categorical then
        $b \leftarrow b$.getChild($l_i$)
$l_{head} \leftarrow$ head of $c_{base}$
$c_{body} \leftarrow$ clause of node $b$
$h \leftarrow b$.getChild($l_{head}$)
if $h \neq b$ and edge $\{b, h\}$ is not independent then
    return $I_Y(l_{head}, c_{body})$
else
    return 0;
```

Algorithm 6 searchNumericalIntervals(c_{base},l_{num})

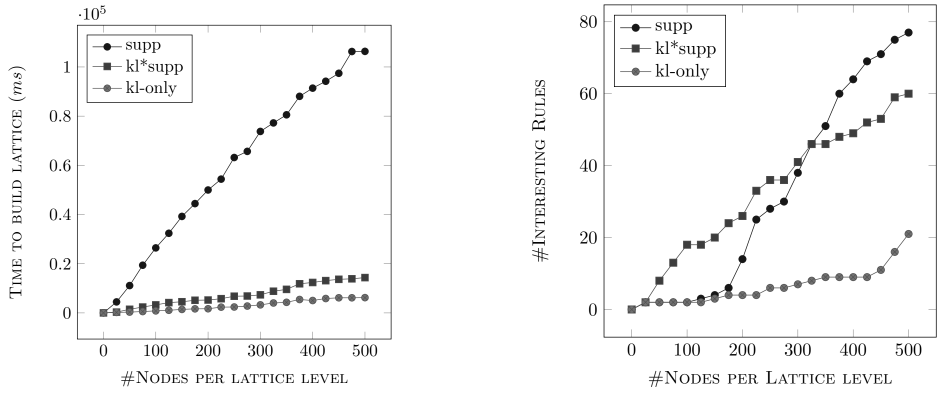
```
$Y \leftarrow$ numerical variable of $l_{num}$
$conf$[] $\leftarrow$ queryConfidenceDistribution($c_{base}$,$Y$)
$supp$[] $\leftarrow$ querySupportDistribution($c_{base}$,$Y$)
$intervals \leftarrow$ getInterestingIntervals($conf$,$supp$)
for $interval_i \in intervals$ do
    $c_{ref} \leftarrow c$ refined with $interval_i$
    add $c_{ref}$ to ILP's refinement graph
```

we focus on evaluating the quality of the correlation lattice and the scalability of its construction in terms of runtime. Afterwards we evaluate how the proposed correlation-based approach affects the core ILP algorithm. Since there is no prior work on the exploration of base rules search space, we compare our approach against the naïve exhaustive search, which searches for all the base rules in the ILP refinement graph. This allows us to evaluate the core contribution of the proposed approach by measuring the runtime reduction and its impact on the number of learned rules in comparison to the naïve approach.

We used support only (*supp*) as a baseline, Kullback-Leibler only (*kl-only*), and Kullback-Leibler combined with support (*kl*supp*). Experiments with other divergence measures such as Jensen-Shannon and Chi-square were carried out, too. However, preliminary results showed no major performance differences. Hence, we only report results for Kullback-Leibler focusing on the comparison of support only, divergence only, and hybrid measures. Our experiments were conducted on a

Intel i7-3770 3.40 GHz with 32 GB RAM.

U.S. Census. In the first experiment, we greedily built limited size correlation lattices using the different interestingness measures as heuristics. We varied the limit size and counted the number of interesting rules with numerical intervals which can be found in the lattice, in order to assess its quality. Finally, we measured the required time for building the lattices. For this experiment, we use the *U.S. Census* (2000) dataset, because of its high quantity of numerical and categorical properties, high completeness and low noise. We built a lattice for the *income* property and 16 categorical properties totalizing 224 categories with various degrees of correlation with *income*. The lattice was constructed on a 1.7 GB partition (with 884,365 person entities) and the extracted rules were tested on a 0.5 GB partition (with 369,024 *person* entities). The partitions were created with random sampling, and they are mutually disjoint.



(a) Build time per lattice size

(b) Learned rules per lattice size

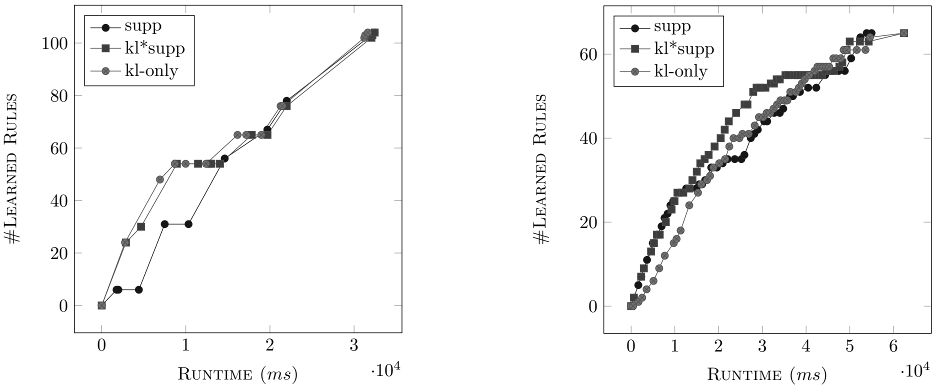
Figure 10.6: Evaluation of the correlation lattice construction

In Figure 10.6a, we see that *supp* takes the longest to build the lattice, *kl-only* takes the shortest, while *kl*supp* lies in-between. This is because the support prioritizes nodes with higher support clauses, while the divergence, because of sampling error, prioritizes lower support.

Figure 10.6b shows that although for the hybrid measure the lattice requires much less time to be built than for *supp*, the number of resulting interesting rules is roughly the same on average, and significantly higher for smaller lattices. The comparatively low efficiency of the *supp* measure for smaller lattice sizes can be accounted for by the fact that the literals with the highest support, such as *hasDeficiency(X,none)*, tend to have very low divergence and low correlation to *income*. We also tested the accuracy of facts predicted by the learned rules. For all measures, the average accuracy was 0.81, with the exception of *kl-only* (0.75), since the sampling error caused a lower average support.

In the second experiment, we evaluate the proposed ILP extension, and compare the effectiveness of the different interestingness measures. When the minimum interestingness thresholds are set to zero, our approach is equivalent to a naïve exhaustive search. Therefore, we know the number of interesting base rules which exist in the search space. This is reflected by the top-right points in Figure 10.7. For the evaluation reported in the following, we use *DBpedia* 3.9[2] and *Freebase*[3]. Although these datasets do not contain as many numerical facts as the U.S. Census data, they consist of richer RDF graphs with a high number of links between different entities. Hence, they enable us to evaluate our extension's performance when it comes to learning more complex datalog rules.

DBpedia. In the DBpedia experiment, we used 27 properties from the domain *movies* and related domains containing 3.5 million facts, and focused on learning rules with numerical intervals for the movie *budget* property with 14,769 facts. Firstly we ran the naïve algorithm where for all base rules in the ILP refinement graph we perform the numerical intervals search, in order to find out how many interesting refined rules exist and how long the exhaustive search takes. Afterwards, we run our proposed extension with different interestingness threshold values, and for each threshold value, we measured the runtime spent with the search for refined rules and the number of interesting rules discovered. Figure 10.7a shows the number of learned rules per runtime for the different measures. We notice that *supp* has a nearly linear growth, while the other measures have an overall better performance, indicating that including divergence in the interestingness measure helps speed up the ILP algorithm.



(a) DBpedia

(b) Freebase

Figure 10.7: Evaluation of the ILP extension

Freebase. We performed the same experiment on 27 properties of the Freebase domain *people* and related domains containing a total of 10 million facts, and learned rules with numerical intervals for the properties *weight*, *dateOfBirth*, *height* and *salary*, with 99.918, 1 351 870, 160 326 and 16 452 facts, respectively.

[2] `http://dbpedia.org/Downloads39`

[3] `https://developers.google.com/freebase/data` (2013-10-06)

This difference in terms of performance highly depends on characteristics of the data, such as the degree of correlation between the numerical attributes and categories or the average support of the interesting base rules. Nevertheless, we notice that the hybrid measure are the most robust of the proposed measures, consistently being the best overall performer in the experiments.

10.7 Conclusion

In this chapter, we presented an extension to the ILP algorithm which uses correlation lattices to enable the efficient discovery of rules with numerical attribute intervals. We suggested different interestingness measures for constructing these lattices and evaluated their performance on the lattice construction and interestingness prediction tasks. Our experiments indicate that the combination of divergence and support is the best performing measure both as a pruning heuristic and for speeding up the core ILP algorithm.

Chapter 11

Thesis Conclusion

In this thesis different approaches for automatic refinement of knowledge graphs have been presented, both at TBox and ABox level. At ABox level, we have made contributions in the detection and correction of relation assertion errors and prediction of missing type assertions. Additionally we propose a model for synthesizing knowledge graphs for completion benchmarking. At TBox level, we have proposed a method for learning intuitive class expressions using lexical information, the generation of relation constraints for data validation and a method for efficiently learning rules with numerical attributes.

In the following sections we give a more detailed summary of the conclusions drawn from each part of the thesis.

11.1 Part I: ABox Refinement

In Part I we propose methods for enrichment and detection of errors in the ABox. In Chapter 4 we addressed the research question RQ4 by proposing a knowledge graph model and synthesis process which is able to capture essential characteristics of existing knowledge graphs, allowing the creation of replicas of those graphs at different scales. Extensive experiments comparing the replicas and original datasets in the link and type prediction tasks were conducted. We have performed evaluations with five different methods for each task and comparisons of distances and methods rankings between replicas and original datasets. Overall, the model M3 was the best performer, and the use of horn rules significantly improved the results. The use of a bias to selection of subject and object individuals did not show any significant improvement. In general, we recommend the use of M3, unless the objective is to replicate the results of type prediction for a single method, without performing a comparison to other methods. In that case M2, which does not include horn rules, would be the best option.

In Chapter 5 we addressed the research question RQ1. We have formulated type prediction as a hierarchical multilabel classification problem. We propose SLCN, which is a local classifier per node hierarchical multilabel classifier. It takes advantage of the type hierarchy and uses local feature selection and sampling. We compare our approach to popular hierarchical multilabel classifiers and to SDType (which is currently one of the strongest and best scalable algorithms for the type prediction task). The experiments indicate that the local feature selection and local sampling can significantly improve scalability without sacrificing the quality of the prediction, and they also show that SLCN can perform better than SDType, and scales better than the other multilabel classifiers evaluated. Given a smaller datasets and enough computing power available, state-of-the-art hierarchical multilabel classifiers, such as HOMER, HMC and MLC4.5, are the best choice. However, on larger datasets with high number of features, and where training time is an important factor, SLCN is the best option. We also evaluated the use of entity embeddings as features for our proposed type prediction approach and the initial results indicate that under the settings used in the experiment, the simple graph-features, such as ingoing and outgoing relations and qualified relations, yield significantly better results, and combining entity embeddings with them does not provide any significant improvement.

In Chapter 6 we address the research question RQ1 by proposing a method for detecting relation assertion errors. We have shown that although the error detection problem is similar to knowledge graph completion problem, methods which perform well in the latter might not necessarily perform well on the error former. We propose PaTyBRED, a robust supervised error detection method which relies on type and path features, and compare it with state-of-the-art error detection and knowledge graph completion methods. We demonstrate the importance of combining those path and type features together, and showed that our proposed method outperformed all the compared methods.

In Chapter 7 we address the research question RQ1 and proposed CoCKG, an approach for correcting erroneous facts originated from entity confusions. The experiments show that CoCKG is capable of correcting wrong triples with confused instances, with estimated precision of 21% of the produced corrections in DBpedia and 14% in NELL. The low precision values obtained do not allow this process, as of now, to be used for fully automatic KG enrichment. Nevertheless, it works as a proof of concept and can be useful, e.g., as suggestions from which a user would ultimately decide whether to execute.

11.2 Part II: TBox Refinement

In Part II of the thesis we proposed methods for automatically learning ontologies. In Chapter 8 the research question RQ2 is addressed by inducing relation constraints from ABox data. We propose a method for learning relation constraints which is based on the relation assertion error detection method PaTyBRED. We compare the learned SHACL constraints with RDFS domain and range restriction learned with statistical schema induction. We performed a manual comparison of the two approaches on DBpedia, and we show that our SHACL constraints are better at detecting wrong relation assertions while being more robust when handling noise and incompleteness of subject and object type assertions.

In Chapter 9 we address the research question RQ2 and present the first steps towards combining the previously distinct logical and lexical ontology learning areas. By extending a formerly pure logic based approach with statistical methods which can be used on text corpora, we were able to foster the generation of more intuitive class expressions. An extensive manual evaluation with four human annotators showed that the integration of relevance measures can significantly improve results. Nevertheless, we also discovered and analyzed several problems for which we were partially able to trace back to data quality issues.

The research questions RQ2 and RQ3 were considered in Chapter 10, where we presented an extension to the ILP algorithm which uses correlation lattices to enable the efficient discovery of rules with numerical attribute intervals. We suggested different interestingness measures for constructing these lattices and evaluated their performance on the lattice construction and efficiency of the interestingness measure for exploring the rules search space. Our experiments indicate that the combination of divergence and support is the best performing measure both as a pruning heuristic and for speeding up the core ILP algorithm. The proposed correlation lattice has also been shown to be useful when combined with outlier detection methods for discovering numerical errors in Linked Open Data (62). It allows the efficient exploration of subpopulations defined by categories in the search for outliers which cannot be detected when considering the whole population.

11.3 Open Issues and Limitations

As mentioned in the individual chapters, there are many open issues and limitations. In many occasions it was necessary to restrict the scope of our proposed approaches in order to focus on specific aspects of a problem.

Our proposed method for generating artificial knowledge graphs cannot yet generate artificial ontologies from scratch. At the moment it uses an already existing TBox and generates ABox assertions based on the given TBox. Another limitation is that, when using horn rules to create graph path patterns, scalability becomes an issue. Generating large datasets with millions of instances and facts becomes expensive since the knowledge graph needs to be queries for every added fact in order to verify if the antecedents of candidate rules are satisfied.

Both in the relation assertion error detection and type prediction approaches we restrict ourselves to considering only relationships between entities, in order to focus on the multirelational aspect of knowledge graphs. These approaches would most likely benefit from other kinds of feature which can be found on many knowledge graphs, such as numerical and textual properties or even images. Both methods can be easily extended to support other kinds of features. Exploiting multimodality is a promising direction to be followed, and at the moment there are a few works which go towards this direction.

Another important limitation of our approach for correction of confusions between entities is the fact that it often detects correct facts with wrongly typed subject or objects as erroneous. This happens because type information is often the most relevant feature for error detection, and because datasets often have missing or wrong type assertions. Although our proposed approach was able to mitigate this kind of issue by integrating a type predictor into the algorithm, it remains a major limitation. Additionally, a more extensive evaluation including other knowledge graph models to generate triple scores as well as ensembles of different models would be desirable. Since the experiments only used PaTyBRED to calculate the triple scores, we cannot evaluate the full potential of the proposed approach. Another issue is the rather low accuracy obtained in the experiments, which makes it prohibitive to be used in a fully automatic way.

Learning ontologies is a challenging problem, and fully automatic approaches can struggle to produce highly accurate output, often requiring it to be reviewed by a specialist. Introducing human knowledge in the learning process, with active learning e.g., can lead to a higher level of performance while minimizing human supervision.

11.4 Future Work

In the future, we intend to further work on our proposed knowledge graph synthesis model and enable the synthesis of data from scratch, including schemas. We plan to create a system which enables users to synthesize data based on a set of parameters that gives control on important characteristics of a knowledge base, such as number of entities, types, relations, assertions of types and relations, density, connectivity. We also want to synthesize a set of knowledge bases of different characteristics in order to create a benchmark for link prediction and type prediction.

Regarding type prediction, since our approach assumes independence between sibling classes, we plan to consider a post processing step to take disjointness axioms into account. Combining our approach with specific feature selection methods for Semantic Web datasets would be a promising refinement. We also plan to exploit dataset specific features, such as DBpedia features from abstracts, and evaluate their impact on method's performance. Furthermore, we want to adapt our approach to support DAGs as type hierarchies, instead of trees only, and investigate the impact it has on the quality of the predictions and runtime.

Incorporating other kinds of features that can be extracted from knowledge graphs, such as numerical attributes and textual data, would be an interesting extension to our PaTyBRED error detection approach. It would also be interesting to evaluate the impact of each kind of features on the error detection performance. As discussed in Chapter 10, numerical attributes can be extremely relevant for some relations. For instance, incorporating the attribute age can help better detect errors on the relation married.

Adapting our approach for correction of confusions between entities to support active learning is another idea worth being investigated. Since guaranteeing the quality of the newly generated facts is crucial, having input from the user to clarify borderline cases and improve the overall results would be highly valuable. Furthermore, using an ensemble of different KG models with different characteristics, e.g. KG embeddings, instead of a single model may potentially increase the robustness of the system. It would also be worth adding textual features from entities descriptions to help determine if a pair of entities is related or not.

We also consider the generation of SHACL constraints for numerical and textual data in the future. For numerical data constraints we can extend previous numerical data outlier detection works (125; 62) to derive intervals which can be used as constraints. For textual data we plan to use, for instance, regular expression learning from data (60) in order to learn SHACL string pattern constraints.

We also plan to closely integrate the output of the lexical analysis into the ILP refinement process in order to more efficiently produce intuitive class expressions. This might positively influence the search in the hypotheses space, therefore resulting in the generation of more intuitive solutions first. Additionally, we are going to extend our approach by employing more sophisticated word sense disambiguation techniques, which will help the more accurate identification of entity mentions in the text. We will also include WordNet and other lexical resources that can facilitate the detection of words which are synonymous to ontology entity labels.

Possible next steps in the direction of efficiently learning rules with numerical attributes are to study the lattice build time and ILP speed up trade-off, investigate numerical domain discretization methods and their effect on our approach, and determine optimal minimum interestingness threshold values by analyzing the lattices structure. We envision an application of our approach to multiple Linked Data sources. By building correlation lattices with categories from distinct but related sources we hope to enable an efficient discovery of cross-domain rules.

11.4.1 General Perspective

Automatic knowledge graph refinement has several challenges which need to be investigated in the future. In this subsection give a general perspective of some important problems which were not necessarily addressed in this thesis.

In terms of knowledge graph completion, there is a lot of research on developing new models. However, a problem which received little attention is the exploration of the search space of possible candidate triples. In Section 7 we briefly discuss that problem, and show that on large-scale datasets this search space is huge. For instance, in DBpedia, the search space size is of the order of 10^{17} candidates triples. It is virtually impossible to compute triples scores for all candidates in order to find out which triples to add to the knowledge graph. Therefore, investigating possible heuristics to efficiently explore the search space is crucial for link prediction to become feasible on larger datasets.

Another important problem concerning the link prediction research is the lack of a thorough comparison of the methods proposed so far. The small number of datasets used in the evaluation has been subject to criticism (82; 191). Some of the major problems are the lack of large real-world datasets and unfair comparisons between models, where different ways of training the embeddings, gradient descent methods, number of negative examples and different dimensionalities are used. Firstly, it would be necessary to create a set of benchmarking knowledge graphs of different characteristics, so we can better understand what kind of model is best suited for different kinds of graphs. Secondly, the various knowledge graph models should be implemented under the same framework, in order to minimize the effect of external factors which can influence the performance, and ultimately have a fairer comparison.

Incorporating information from multimodal sources into the knowledge graph model is also an interesting problem. Although most large-scale knowledge graphs have a lot of data properties, such as numerical information, geo-location, text and images, most knowledge graph models focus exclusively on the relationships between instances. Although there has been some works in that direction (189; 17; 93; 68), the problem deserves more attention.

Regarding the detection of relation assertion errors, an important problem is what to do with the detected errors. Deleting the errors is not always the best option. As shown in Section 7, some of those errors can provide hints for their correction. Moreover, error detection methods may wrongly detect a correct relation assertion as erroneous because of secondary facts, for instance a subject with missing type assertions. In such case, the best thing to do would be to address the actual source of the problem instead of deleting the correct relation assertion. Understanding what causes a relation assertion to be detected as erroneous and find out the best way to address the problem is a research direction which can be very useful.

In terms of TBox refinement, an interesting research question is whether knowledge graph embeddings can be used to learn constraints in the latent space. For instance, it is possible to learn relation constraints that restrict the region in the latent space where the subjects and objects should be located (e.g. a hypersphere or a hypercube). That would allow the introduction of knowledge graph constraints which cannot be represented in languages such as OWL and RDFS.

All in all, automatic knowledge graph refinement is a challenging but highly relevant research topic. Knowledge graphs are being used on a wide variety of applications, such as medical diagnosis, question answering, information retrieval. Improving the quality of such datasets can not only provide a better representation of the application domain, but ultimately also positively impact its various applications.

Appendix A

Local vs. Global Feature Selection in Multilabel Classification

A.1 Introduction

Multilabel classification denotes a classification problem where a single instance cannot be assigned only one, but multiple classes. It has gradually attracted significant attention from various communities and has been widely applied to diverse problems from automatic annotation for multimedia contents to bioinformatics, web page classification, information retrieval, tag recommendation and many others. In this thesis, we use multilabel classification for the type prediction and relation assertion error detection problems in knowledge graphs (c.f. Chapters 5 and 6 respectively).

There are two general families of multilabel classification algorithms: (1) *adaptations* of single-label machine learning algorithms which deal with multilabel data directly, and (2) *transformation methods*, which decompose the multilabel problem into a set of simpler learning problems, usually binary classification. Transformation methods have been widely used and allow standard binary classifiers to be used on the transformed problems, which are independent from each other and can be easily parallelized.

Binary Relevance (BR) is one of the simplest and most popular transformation methods. Its main drawback is that it does not consider dependencies between labels, which can be modeled with more complex transformation methods such as Classifier Chains (CC), Label Power-sets (LP), or Pruned Problem Transformation (PPT). On the other hand, taking those dependencies into account results in higher computational complexity, and scalability requirements might prohibit the use of more sophisticated methods which model dependencies. However, it has has been shown that in many cases, BR can yield predictive performance as good as more complex methods depending on the characteristics of the data (115). Hence, we focus on BR as a transformation method.

Feature selection is an important part of machine learning, allowing the reduction of training time, as well as the improvement of predictive quality (46; 85; 129). When using a transformation approach for multilabel classification, the selection of features can be performed locally or globally. In the global approach, the feature selection is performed only once on the original dataset, and the set of selected features is the same for all local transformed datasets. In the local approach, the selection is performed separately for each local classifier on its correspondent transformed dataset, which means that different local classifiers may work on different sets of class-specific features specialized for each transformed problem. This can be particularly interesting for some datasets where the features which are relevant for predicting one class might not be relevant for another.

In hierarchical multilabel classification, where the labels are structured in a hierarchy, the use of the hierarchies can also influence the predictive performance and runtime of the local and global feature selection processes (179). In this chapter, we examine the difference between local and global feature selection not only with flat multilabel data, but also on hierarchical multilabel classification problems. So far, the global feature selection approach has been widely studied in the literature (52; 182; 221), while the local approach, although already considered in the context of multiclass classification (47), has received little attention and has not been systematically evaluated in the multilabel classification problem. The idea of *generating* class-specific features for the problem transformation method has been considered by LIFT (222), however, there are no works considering class-specific feature *selection* and performing a comparison with the global approach in terms of predictive performance and runtime on transformation based classifiers.

In this work, we conduct an empirical comparison between the local and global feature selection approaches. We show that when using transformation methods, the local approach results in a better overall predictive performance with similar runtime. To the best of our knowledge, although local and global methods have been discussed in the literature, this is the first systematic empirical evaluation and comparison of the two approaches.

The rest of this chapter is structured as follows. Section A.2 gives an introduction to multi-label classification, feature selection, and the evaluation metrics used in this chapter. Section A.3 introduces related works, parts of which are also used in the experiments. Section A.4 discusses the use of local feature selection with transformation methods, and section A.5 presents the empirical results. We end with a discussion of results and an outlook on future work.

A.2 Background

In this section, we present the background work relevant to the understanding of this work. We define the flat and hierarchical multilabel classification problems and briefly present various state-of-the-art methods, as well as evaluation measures used in the experiments.

A.2.1 Multilabel Classification

In the multi*label* classification problem, there are multiple classes and, contrary to the multi*class* classification problem, instances are allowed to have more than one class. We define the set of classes as $C = \{c_1, ..., c_{|C|}\}$, and we represent the multilabel of an instance x with a binary vector $y = (y_1, ..., y_{|C|}) \in \{0, 1\}^{|C|}$.

Multilabel classification approaches can be divided into transformation and adaptation methods. These will be discussed in more details later in this section. A comprehensive review on multilabel classification algorithms is given in Zhang et al. (225).

Some of the existing multilabel classification approaches are variations of standard binary classification algorithms, which have been adapted to the multilabel task without requiring problem transformations. This include, for example, *AdaboostMH* (175), *MLkNN* (224), and MLC4.5 (37), which are the respective multilabel versions of Adaboost, k-Nearest Neighbors, and C4.5 (a decision tree algorithm):

- AdaBoostMH is an adaptation of AdaBoost with its weak-learning conditions being based on a one-against-all reduction to binary, which was originally designed to use weak-hypotheses that return a prediction for every example and every label.

- The retrieval of the multilabel k-nearest neighbors is the same as in the traditional k-NN algorithm. The main difference is the determination of the label set of a test example, where the prior and posterior probabilities of each label within the k-nearest neighbors are considered and the Bayesian rule is use to derive the predicted set of labels.

- The multi-label C4.5 algorithm (ML-C4.5) adapted the original C4.5 by modifying the formula for calculating entropy to consider distributions over all labels, which is equivalent to the sums of the entropies for each individual class label, and allowing multiple labels in the leaves of the trees.

Adaptation methods usually learn a single model capable of predicting all classes. In comparison, transformation methods will have as many models as transformed datasets. Therefore, generally speaking, transformation methods require more memory space than adaptation methods.

Transformation methods decompose the multilabel problem into a set of binary classification problems. There are mainly three categories of transformation methods according to Madjarov et al. (115): binary relevance, label power-set, and pair-wise methods.

The most popular method is called *Binary Relevance* (194) (BR), which trains a binary classifier for each class (against the others), inherently assuming independence between the classes. *Classifier Chains* (164) (CC) arrange the local classifiers in a chain where the outcome of a classifier is used as a feature on the next classifiers in the chain, allowing some dependency between labels to be modeled.

In the label power-set category, every different combination of labels occurring in the training data is considered an individual class and the transformed classification problem is multiclass. The potentially high number of label combinations poses scalability challenges, as the number of label combinations can grow exponentially with the number of labels resulting in up to $2^{|C|}$ transformed labels. This can significantly increase the complexity of the problem, making it prohibitive for datasets with a high number of classes.

Pruned Problem Transformation (PPT) (162), in order to reduce the complexity of the LP approach, selects only the transformed labels that occur more than a minimum number of times. *HOMER* (195) (acronym for Hierarchy Of MultilabEl leaRners) generates a hierarchy for the labels, with meta-labels being non-leaf nodes representing the union of a subset of labels. A multilabel classifier is then trained for every non-leaf node in the hierarchy having the node's children as labels.

Ensemble methods use multiple adaptation and problem transformation methods as base classifiers and combine the output of the different trained models to make a prediction. *Random k-Labelsets (RAKeL)* (196) generates random sets of labels and trains a label power-set classifier for each randomly generated subset. *Ensemble of Pruned Sets* (EPS) (165) uses pruning to reduce computational complexity as well as example duplication method in order to reduce the error rate. *Ensemble of Classifier Chains* (ECC) (164) has classifier chains as base classifiers, the output of each base classifier is summed, and a threshold is applied to select the labels predicted.

An extensive experimental comparison of multilabel classifiers including adaptation transformation and ensemble methods is reported in (115). The authors recommend the use of four different classifiers, three of which are transformation methods, i.e., HOMER, BR, and CC.

A.2.2 Hierarchical Multilabel Classification

The hierarchical multilabel classification problem is similar to the multilabel classification problem, but the classes C are additionally arranged in a hierarchy G. The labels of an instance should be consistent with G, i.e., if an instance belongs to a non-root class then it must also belong to its ancestors (i.e., $\forall c_i \sqsubseteq c_j$, if $y_i = 1$ then $y_j = 1$). The class hierarchy can be of two types: a tree, which allows nodes to have a single parent only, and a directed acyclic graph (DAG) which allows nodes to have multiple parents.

Two important aspects of hierarchical multilabel classification approaches are whether they allow partial path predictions, i.e., mandatory vs. non-mandatory leaf node prediction, and whether they guarantee to output labels that are consistent with hierarchy or not.

As for multilabel classification, there are mainly two types of hierarchical multilabel classification approaches: local and global classifiers. The main difference is that the former is basically a transformation method which breaks down the classification problem into smaller and simpler problems exploiting the class hierarchy, while the latter considers the problem as a whole, learning a single more complex model.

The local hierarchical classification algorithms share a similar top-down approach in their prediction phase, where the classifier first predicts its first-level (most generic) classes of an instance, then it uses each predicted class to reduce the choices of classes to be predicted at the second level (the children of the classes predicted at the first level), and so on, recursively, until the most specific prediction is made. According to the hierarchical classification survey from Silla et al. (179), there are mainly two standard ways of using the local information: a *local classifier per node* and a *local classifier per parent node*. The *local classifier per level* approach, where one local multilabel classifier is trained for every level of the hierarchy, is also mentioned in the survey from Silla et al. (179). However, very few works consider this approach since it suffers from inherent consistency problems: LCL has a single classifier per level and the children of nodes classified as false might also be classified as true. In contrast, LCN and LCPN guarantee consistency: in the prediction phase, they are applied top down, only predicting labels for lower levels in the hierarchy if their parent(s) have been predicted.

Local Classifier Per Node (LCN): The local classifier per node approach consists of training one binary classifier for each node of the class hierarchy. Similarly to Binary Relevance, each local binary classifier predicts whether an instance belongs to the class associated with the node or not. There are two main ways to define the training set of of the local binary classifiers. When training a local binary classifier for one label at hand, there are two strategies for selecting the negative examples for the local classifier: the *all* approaches, which uses all instances with all other labels as negative examples, and *siblings*, which only uses instances of the label's siblings in the hierarchy, reducing the size of the transformed datasets for classes in the lower levels of the hierarchy. A comparison of different negative example selection approaches is made in Eisner et al. (54) and Fagni et al. (57). The results indicate that both approaches have roughly the same predictive performance, however, *siblings* is more scalable than *all*, as it reduces the average size of the local training sets.

Local Classifier Per Parent Node (LCPN): In this approach, a local multilabel classifier is learned for every parent (i.e., non-leaf) node in the hierarchy. The labels are the direct child nodes, and the training instances are those which belong to the parent node class. Depending on the choice of the local multilabel classifier, it is possible to model dependencies between sibling nodes. LCPN with Binary Relevance as a base multilabel classifier is equivalent to LCN using the siblings negative example selection policy.

In contrast to local classifier approaches, the global classifier approach (also known as *big bang approach*) learns one single classification model built from the training set, taking into account the class hierarchy as a whole during a single run of the classification algorithm. When used during the prediction phase, each instance is classified by the induced model, a process that can assign classes at potentially every level of the hierarchy to the instance.

One example for a global approach based on the Rocchio classifier is used in Labrou (95). Some global methods do not guarantee consistency with the hierarchy and therefore need a post processing step in order to ensure consistency (86; 87).

Clus-HMC (201) is a version of the previous method featuring predictive clustering trees (13) to generate a label hierarchy, which is not necessarily existent at first. Dimitrovski et at. (49) use ensemble approaches with Clus-HMC and report that the use of ensembles results in increased predictive performance. Otero et al. (144) propose a global hierarchical Ant-Miner classification method which is able to handle DAG hierarchies.

Further details about hierarchical multilabel classification methods can be found in the survey by Silla et al. (179), where an extensive comparison of classifiers, evaluation measures, as well as negative example selection policies is presented.

A.2.3 Evaluation Measures

In order to evaluate the predictive performance of the transformation based multilabel classifiers with different feature selection approaches, we use some popular measures recommended in the literature (179; 115) for our experiments. The μP, μR and μF (86) are the micro-averaged measures of precision, recall and F-measure per class. By using the micro average, each class is weighted according to the frequency it occurs in the test data. The macro-average measures mF, mR and mF are the average with uniform weights for the classes. Equations A.1 and A.2 show the definition of these measures, where tp_i, fp_i and fn_i denote respectively the number of true positives, false positives and false negatives of the class c_i.

$$\mu P = \frac{\sum_{i=1}^{|C|} tp_i}{\sum_{i=1}^{|C|} tp_i + fp_i} \qquad \mu R = \frac{\sum_{i=1}^{|C|} tp_i}{\sum_{i=1}^{|C|} tp_i + fn_i} \qquad \mu F = 2\frac{\mu P \times \mu R}{\mu P + \mu R} \tag{A.1}$$

$$mP = \frac{1}{|C|}\sum_{i=1}^{|C|} \frac{tp_i}{tp_i + fp_i} \qquad mR = \frac{1}{|C|}\sum_{i=1}^{|C|} \frac{tp_i}{tp_i + fn_i} \qquad mF = 2\frac{mP \times mR}{mP + mR} \tag{A.2}$$

Equation 5.4 shows the Hamming loss (l_h) for one instance. We denote the true label vector of an instance as y, and the predicted vector as $\hat{y}$, with $y_i = 1$ if the instance is of class c_i, $y_i = 0$ otherwise. Hamming loss reports how many times on average, a class label is incorrectly predicted, i.e., the number of false positives and false negatives, normalized over total number of classes and total number of examples.

Equation 5.5 shows the hierarchical loss (hl_H) (30), which is a hierarchical multilabel classification measure that extends hamming loss to account for any existing underlying hierarchical structure of the labels.

The idea of hierarchical loss is based on the notion that, whenever a classifier makes a mistake at a given node in a hierarchy, no further loss should be counted for any mistake in its descendants, therefore ignoring all wrong predictions for nodes which are descendants of a wrongly predicted node.

Costa et al. (44) review evaluation measure for hierarchical classification, and the authors conclude that there is no consensus on what measure should be used, and none of them have been widely adopted, while most works use the standard flat measures. Brucker et al. (22) make an empirical comparison of hierarchical and flat multilabel classification evaluation measures and search for relations between them. The authors report that hierarchical measures improve the quality assessment for hierarchical classification over flat measures, at the same time they state that flat and hierarchical measures agree on whether a classification result is good or not.

Cerri et al. (29) experimentally analyze methods for multilabel classification which are based on decision trees. Various evaluation measures are investigated in terms of consistency, discriminancy and indifferency. The authors suggest the use of hF, hP and hR as evaluation measures. These are equivalent to μF, μP and μR with all observed and predicted labels consistent with the class hierarchy as described in Section A.2.2, i.e., if one instance is assigned a non-root class, it must also be assigned all the ancestors of the given class.

Kosmopulos et al. (91) make a detailed study of the problems of hierarchical classification evaluation measures. The authors categorize the evaluation measures into pair-based, which uses graph distance measures to compare predicted and true labels, and set-based, which use hierarchical relations to augment the sets of predicted and true labels (includes hF, hP, hR). Their results indicate that set-based measure F_{LCA} (F-measure with lowest common ancestor), which is highly correlated with hF, with the main difference being in DAG cases, which we do no consider in this work.

For hierarchical classification we choose to use hF, because it is widely recommended in the literature. Since in our hierarchical classification experiments the labels are always consistent with the class hierarchy, which makes hF equivalent to μF, we choose to call it μF in this chapter. In order to emphasize the performance on less frequent classes we also use the its macro-averaged version mF.

It is important to note that we do not use AUC and ROC, although these measures would be relevant in our evaluation. In order to use AUC and ROC in the multilabel context, it is necessary that all the classes share the same confidence threshold, however, in the transformation methods we allow local classifiers to have their own confidence threshold values as they are trained independently from the other local classifiers.

A.2.4 Feature Selection Methods

Traditional feature selection methods can be divided into filter, wrapper, and hybrid methods (46; 129). Filter methods rank the features based on some relevance measure, and select the k highest ranked features according to that measure. It is the simplest and most scalable of the methods. Two of the most popular relevance measures used are mutual information (MI), which is equivalent to information gain, and chi-squared (χ^2). Other filter techniques include, for instance, relief (85), which is based on separation capabilities of randomly selected instances, and ensembles of different measures (172; 142), where different ranking measures are used, and the feature ranks are combined. This approach has been extended for the hierarchical multilabel classification problem by Slavkov et al. (180).

In wrapper methods, a classifier is repetitively invoked and evaluated with different feature subsets. Exhaustive search is a method where all possible combinations of features are tested, and the one combination which yields the best performance is selected. This, of course, is utterly expensive and feasible only on datasets with small number of features. A popular wrapper method is the greedy forward search, where features are incrementally selected one at time in a greedy way, considering dependencies between features and reducing redundancy. It uses heuristics in order to reduce the search space, and therefore does not guarantee that the selected set of features is optimal.

Hybrid methods, as the name suggests, combine characteristics of both filter and wrapper methods. Huda et al. (78) include the filter's feature ranking score into the wrapper stage to speed up the search process. Zhu et al. (227) incorporate a filter ranking method into the memetic evolutionary algorithm accelerating the search and identifying core feature subsets.

A.3 Related Work

The quantitative effect of local feature selection has not been empirically researched, but some approaches follow similar ideas. Label specific sets of features has been exploited by the Label Specific Features (LIFT) (222). The label specific features are generated by clustering the set of negative and positive instances (N_k and P_k) into m_k clusters each, and for each instance $2m_k$-dimensional features are generated by computing the distance from the instance to each of the centroids. Qu et al. (159) use the concept of local feature selection, and propose a relevance measure based on the density of negative and positive instances. It requires the distances between all pair of instances to be calculated, therefore the complexity grows with the number of instances squared, making it prohibitive for larger datesets.

Doquire and Verleysen (52) perform a comparison between multidimensional mutual information (MI) and chi-squared (χ^2) using greedy forward search strategy with problem transformation method (PPT). A nearest neighbors based MI estimator is used combined with a simple greedy forward search strategy to achieve feature selection. Their experimental results indicate that the MI based approach has an advantage over the method based on the χ^2 statistic. Particularly, the proposed approach generally leads to an increase in performance both for the Hamming loss and the accuracy compared with the case where no feature selection is considered.

Zhang et al. (221) compare the performance of feature selection strategies based on principal component analysis (PCA) and genetic algorithms (GA) on MLNB (an adaptation of Naïve Bayes for the mutilabel problem), and on Naïve Bayes with binary relevance transformation. Their experiments indicate that the feature selection methods studied lead to an improved performance of the Naïve Bayes based classifiers. The authors raise scalability issues concerning the complexity of GA feature selection and its applicability on larger data with higher dimensional feature spaces. It is important to point out that the PCA requires the features to be numerical, therefore alternative methods or additional preprocessing steps should be considered for handling nominal attributes.

On the multi-class classification problem, de Lannoy et al. (47) study the local feature selection approach on the one-vs-all and one-vs-one approaches. They focus on correcting a potential bias caused by the fact that the binary classifiers are trained on different feature sets. On the hierarchical multilabel classification problem, Kosmopoulos et al. (90) uses a more scalable version of PCA for dimensionality reduction on sibling nodes at higher levels of the hierarchy which are the most expensive local classifiers because of the high number of instances. They perform experiments on the large-scale hierarchical text classification challenge data (145) (LSHTC[1]).

In this work we focus on the comparison of local and global feature selection approaches on transformation methods for multilabel classification, in particular binary relevance. To that end, we apply similar feature selection techniques locally and globally, evaluate the performance of transformation based methods with different base classifiers, and compare the results for flat and hierarchical multilabel classification datasets in terms of predictive performance and scalability.

A.4 Feature Selection on Transformed Multilabel Classification

When transforming a multilabel classification problem into a set of binary problems, there are two possible ways of performing feature selection: global feature selection, where all local binary classifiers are trained on the same set of globally selected features, and local feature selection, where the feature selection is performed separately for each local binary classifier.

A.4.1 Global Feature Selection

The global approach selects a set of features which is shared by all the local classifiers. This approach, in contrary to the local approach, can also be applied on

[1] `https://www.kaggle.com/c/lshtc`

adaptation methods. This may be one reason why the global approach has generally received more attention in the literature.

Many global feature selection methods involve the computation of relevance measures individually for every class, followed by the aggregation of the class specific values. Spolaor and Tsoumakas (182) compare aggregation methods for the binary relevance approach. The approach consists of computing the relevance of each feature for every class individually, exactly like in the binary classification problem. Afterwards the global relevance of a given feature is computed by aggregating the relevance values of the feature for all classes.

The feature selection methods evaluated in the study were Mean, Min, Max, Round-Robin, and Rand-Robin. In Mean, the aggregated value is simply the average relevance of the feature over all classes, in Min the aggregation value is the lowest relevance and in Max the greatest. The features are then ranked by their aggregated values, and those with highest aggregated relevance are selected. Round-robin selects the best feature in the ranking related to each label in turn, while Rand-Robin selects the best feature for a label randomly chosen with probability inversely proportional to its frequency. Each feature taken in turn is removed from the rankings so that they cannot be selected again. The motivation for the Rand and Round-Robin approaches is to reduce the bias to selection of features more relevant to frequent classes to the detriment of less frequent ones.

The authors evaluate these approaches with both Chi-squared and Bi-normal Separation as relevance measures. The experimental results indicate the Max and Mean aggregation methods with chi-squared measure were the best performers.

Other feature selection methods adapts the traditional feature relevance for binary classification to consider all classes at the same time. Doquire and Verleysen (52) compare the multidimensional mutual information (MI), computed with Kozachenko-Leonenko estimation (92), and chi-squared (χ^2) relevance measures on a problem transformed with PPT and greedy forward search strategy for feature selection. The authors report a better performance of the MI based approach in comparison with the method based on the χ^2 statistic in terms of both hamming loss and accuracy. The use of greedy forward selection ensures dependencies between features are considered, however, that also affects scalability, which is an important aspect in our setting.

A.4.2 Local Feature Selection

The local feature selection approach can be applied on any transformation based multilabel classifier. The idea of local feature selection is to perform the selection for every local transformed dataset separately, i.e., different local classifiers may work on different sets of features, which are specialized for each subproblem. Given a dataset D with features set F, transformed labels $t \in T$, where t is a binary class attribute and T is the set of transformed labels, and transformed datasets $\{D_t | t \in T\}$, where D_t is a transformed dataset with binary class t. We define the set of locally selected features for each label as $F_t \subseteq F$ for each transformed

Rank	Global	Local(B.B2)	Local(C.C6)	Local(C.C10)	Local(B.B9)
1	prices	subject	california	attorney	http
2	california	enron	commission	confidential	kaminski
3	price	pmto	price	received	forward
4	power	forwarded	diego	committee	email
5	utilities	steven	prices	include	issue
6	generators	kean	plants	continue	www
7	plants	original	customers	policy	ferc
8	federal	cc	generators	williams	meeting
9	electricity	message	market	50	mailto
10	davis	na	electricity	27	drive

Table A.1: Enron features ranked by information gain in descending order

dataset D_t, and F_{local} as the collection of subsets $\{F_t | t \in T\}$.

In order to demonstrate the practical difference between local and global feature selection approaches, we show as an example the top 10 features selected for the Enron dataset with the filter method and information gain. The dataset is a subset of Enron email corpus[2], labeled with a set of categories. It is based on a collection of email messages exchanged by Enron's board of directors that were categorized into 53 topic categories, such as company strategy, humour, and legal advice. The example from Table A.1 shows the set of globally selected features and the sets of locally selected features for the labels B.B2 (Forwarded email(s) including replies), C.C6 (California energy crisis / California politics), C.C10 (legal advice), B.B9 (pointers to url). It is particularly noteworthy that the most relevant local features are very different, in fact, the sets of top-10 most relevant local features for the classes B.B2, C.C6, C.C10 and B.B9 are completely disjoint.

The Enron dataset is an example of a dataset where the local feature selection approach clearly outperforms the global approach. Different classes require very different sets of features for the local classifiers, and especially for small numbers of selected features, the difference in the performance is very significant. The predictive performance results for classifiers with the local and global feature selection approaches can be seen in the experiments section in Figure A.1.

A.4.3 Analysis of Local Feature Sets

In this chapter, we define F as the set of all features in a dataset, and $F_{\text{global}} \subseteq F$ as the set of globally selected features. In our setting, all the local feature sets $F_t, t \in T$, as well as F_{global} have the same number of features k.

In order to measure the differences between the set of globally selected features and the locally selected features we need to define measures which compare a set with a collection of set. We propose two similarity measures. The first measure (D_1), c.f. Equation A.3, is the average Jaccard index of the set of global features

[2] `http://bailando.sims.berkeley.edu/enron_email.html`

F_{global} with each set of local features F_t. The second measure (D_2), c.f. Equation A.4, is the ratio between the number of selected features k and the size of the union of all local feature sets $|\bigcup_{t \in T} F_t|$ and it indicates how similar with each other the local feature sets are.

The D_1 measure is bounded by the interval $[0, 1]$, while the D_2 measure is bounded by the interval $[1/|C|, 1]$. For D_1, a value of 1 means that the local sets are exactly equal to the global set, while a value close to 0 means that local feature sets and the global feature set have small intersections. For D_2, a value of 1 means that all the local sets are the same, while a value of $1/|C|$ means that the local features are completely disjoint.

$$D_1(F_{\text{global}}, F_{\text{local}}) = \frac{1}{|C|} \sum_{t \in T} \frac{|F_{\text{global}} \cap F_t|}{|F_{\text{global}} \cup F_t|} \tag{A.3}$$

$$D_2(F_{\text{local}}) = \frac{k}{|\bigcup_{t \in T} F_t|} \tag{A.4}$$

A.5 Experiments

In our experiments we make an extensive comparison of global and local feature selection approaches on standard flat and hierarchical multilabel datasets, as well as large-scale hierarchical multilabel datasets for type prediction on knowledge bases. We use binary relevance (BR) for flat multi-label classification, and its counterpart (i.e., local classifier per node, LCN) for hierarchical multi-label classification. We also run experiments with different popular binary classifiers as local classifier, and different feature relevance measures.

We vary the number of features to be selected, perform the feature selection with equivalent techniques using the local and global approaches, then run the same transformation classifiers on the different sets of selected features and compare their predictive performance, as well as runtime. We restrict the transformation approaches to binary relevance, which is a simple, popular, and generally well performing method, as discussed above. Other transformation methods might transform the labels in different ways, but at the end one classifier will be trained for each transformed label similarly to binary relevance. Therefore, for the goal of comparing the local and global feature selection approaches, the binary relevance method is sufficient.

The choice of the local binary classifier is an important factor in the evaluation, as the feature selection can have different impacts on different classifiers. In the experiments we choose four different popular binary classifiers available in Weka 3.7.10: Naïve Bayes, Decision Tree (J48), K-Nearest Neighbors (IBk), Support Vector Machine (LibSVM), and AdaboostM1 (with decision stump).

A comparison against popular adaptation methods and the transformation methods with the adapted methods as local classifier is also performed. We select three popular adaptation method classifiers for our comparison: Multilabel k-nearest neighbors (MLkNN) and Multilabel C4.5 decision tree (MLC4.5). We compare them with Binary Relevance having their adapted binary classifiers (respectively k-NN, C4.5 and AdaboostM1) as local classifiers.

For our experiments, we use MULAN 1.5, which is an open-source Java library for learning from multilabel datasets based on WEKA. It includes a variety of state-of-the-art multilabel classification algorithms, and offers the global feature selection methods for binary relevance with Mean, Min and Max aggregation approaches. It also contains multilabel evaluation measures described in Section A.2.3.

A.5.1 Datasets

We perform our experiments on popular flat and hierarchical multilabel classification datasets commonly used in performance benchmarking of multilabel classifiers. Furthermore, we created some additional benchmark datasets for hierarchical classification, which we extracted from large-scale cross-domain Linked Open Data sets, such as Wikidata, DBpedia, YAGO and NELL (124), as well as the smaller domain-specific datasets AIFB portal[3] and Mutagenesis.

For creating the benchmark datasets for hierarchical classification, we randomly sampled 10 000 instances from the larger Linked Open Data sets. There, instances usually come with types, which form a hierarchy in an ontology. Hence, predicting the type of an instance is a typical hierarchical classification task.[4] By doing the sampling randomly, we ensure that we do not introduce any bias which could be beneficial for one or the other method.

Multilabel Datasets

The flat multilabel datasets we use were obtained from MULAN datasets repository[5]. The datasets used include bibtex (84), birds (20), cal500 (198), corel5k (53), emotions (192), enron (165), genbase (50), imdb (163) medical (154), scene (18) and yeast (55), rcv1 subsets (109), the Yahoo datasets Arts1, Business1, Computers1, Education1, Entertainment1, Heath1, Science1, Social1, Society1 (199), delicious (195), tmc2007 (184), and slashdot[6]. Table A.2 show statistics about the aforementioned datasets, where we state the number of labels, instances, features, cardinality (the average number of labels an instance has) and label dependency, which indicates the proportion of label pairs which are dependent (we consider de-

[3] http://www.aifb.kit.edu/web/Web_Science_und_Wissensmanagement/Portal

[4] Note that this approach is only for generating benchmarks for hierarchical classification, and for comparing approaches to each other. However, we cannot transfer the results trivially to make a statement about how well the approaches work for the actual type prediction task in the original datasets.

[5] http://mulan.sourceforge.net/datasets-mlc.html

[6] http://meka.sourceforge.net/#datasets

Table A.2: Statistics about the flat datasets used

Dataset	Instances	Labels	Features	Cardinality	Label Dep
Arts1	7484	26	23146	1.654	0.338
bibtex	7395	159	1836	2.402	0.111
birds	645	19	260	1.014	0.123
Business1	11214	30	21924	1.599	0.230
CAL500	502	174	68	26.044	0.192
Computers1	12444	33	34096	1.507	0.364
Corel5k	5000	374	499	3.522	0.030
delicious	16105	983	500	19.02	0.116
Education1	12030	33	27534	1.463	0.216
emotions	593	6	72	1.868	0.934
enron	1702	53	1001	3.378	0.141
Entertainment1	12730	21	32001	1.414	0.338
flags	194	7	19	3.392	0.381
genbase	662	27	1186	1.252	0.157
Health1	9205	32	30605	1.644	0.192
imdb	120919	28	1001	2	0.487
mediamill	43907	101	120	4.376	0.213
medical	978	45	1449	1.245	0.040
rcv1subset1	6000	101	47236	2.88	0.202
rcv1subset2	6000	101	47236	2.634	0.179
rcv1subset3	6000	101	47236	2.614	0.183
rcv1subset4	6000	101	47236	2.484	0.163
rcv1subset5	6000	101	47236	2.642	0.170
Recreation1	12828	22	30324	1.429	0.455
Reference1	8027	33	39679	1.174	0.169
scene	2407	6	294	1.074	0.934
Science1	6428	40	37187	1.45	0.196
slashdot	3782	22	1079	1.181	0.273
Social1	12111	39	52350	1.279	0.186
Society1	14512	22	49060	1.67	0.402
tmc2007	28596	22	49060	2.158	0.641
yeast	2417	14	103	4.237	0.670

pendent those pairs which fail the χ^2 test of independence, and the computation was performed using MULAN's UnconditionalChiSquareIdentifier class).

Hierarchical Multilabel Datasets

For the hierarchical mutilabel experiments we use the datasets from the biological domain which are available at the Clus datasets page[7]: cellcycle, church, derisi, eisen, gasch2, pheno and struc. These datasets are available in clus format and were converted to the Mulan format for our experiments using the converter existent in the Mulan library.

Additionally, we use datasets from Linked Open Data (11) extracted for the type prediction task, which is another example of hierarchical multilabel classifi-

[7] `https://dtai.cs.kuleuven.be/clus/hmcdatasets/`

Table A.3: Statistics about the hierarchical datasets used

Dataset	Instances	Labels	Features	Card.	Fanout	Depth	Label Dep
aifb	27100	57	825	2.189	14.25	2.038	0.083
cellcycle	3757	498	78	8.716	2.846	3.709	0.286
church	3755	498	28	8.702	2.846	3.709	0.286
dbpedia-yago	2886305	474	1946	10.894	1.773	8.050	0.347
derisi	3725	498	64	8.759	2.846	3.709	0.286
eisen	2424	460	80	9.202	2.788	3.685	0.284
gasch2	3779	498	53	8.689	2.846	3.709	0.287
mutagensis	14157	86	132	2.398	7.167	2.486	0.041
NELL	120720	264	505	4.608	5.739	4.298	0.052
pheno	1591	454	70	9.175	2.751	3.682	0.236
struc	3838	498	19629	8.674	2.846	3.709	0.289
Wikidata	19254100	474	1866	2.007	5.386	1.948	0.003

cation problem. The types, which are the labels to be predicted, are organized in a hierarchy, and various features can be extracted from the knowledge base graph and textual description of entities which are often available. The datasets we use are NELL (27), Wikidata (205), DBpedia (12), and YAGO (117), AIFB, and Mutagenesis (167) As features we use binary attributes which indicate the existence of ingoing and outgoing properties (152; 168) for the first four datasets, and qualified relations (152), which indicate the existence of pairs of outgoing relations and object type, as well as pairs of ingoing relation and subject type.

For DBpedia, the types assigned to instances are only single-path, so that the hierarchical classification problem is not multi-label . On the other hand, relation features extracted from YAGO are too scarce to be meaningful. Therefore, we combine the two datasets by using types from YAGO and features from DBpedia. Since there are 384 174 types in YAGO, we select the top-474 most frequent types. We chose 474 because it was the original number of classes in DBpedia. We also use Wikidata, from wich we select the top-474 most frequent types, similarly to what was done to the YAGO types. The NELL dataset (08m.690) we used has only 10.3% of its originally 1 168 998 instances, since the properties are very sparse, and 89.7% of the instances have no features or only have the property *haswikipediaurl* which provides no information gain. The AIFB portal dataset describes the AIFB research institute in terms of its staff, research group, and publications. The data is an export of the AIFB website and contains around 270 thousand triples. The type hierarchy is originally a wide and shallow tree with average fanout 14.25 and average depth 2.04. The MUTAG dataset is distributed as an example dataset for the DL-Learner toolkit[8]. It contains information about 340 complex molecules that are potentially carcinogenic, which is given by the isMutagenic property. The molecules can be classified as “mutagenic” or “not mutagenic”, and the main entity types atoms bonds and compounds which define the molecules.

[8] http://dl-learner.org

These hierarchical knowledge bases have a directed acyclic graph (DAG) as hierarchy. Since the MULAN framework only support trees, we have to simplify the problem and convert the DAGs into trees. It is important to mention that in the evaluation we ignore the original DAGs and consider only the converted trees. Therefore, for the nodes which have multiple parents, we retain only the subsumption relation with the parent class which is most frequent, i.e. the one with most instances, and remove the other edges. Alternatively, one could replicate the subtree of a node with multiple parents, leaving each replica with a single parent node. This, however, may significantly increase the number of classes, and most importantly generate consistency problems in case a classifier produces different predictions for the subtree replicas, which is a problem we do not address in this chapter. Although supporting DAGs instead of converting them to trees can improve results (10), and the aforementioned conversion is an interesting approach to be considered, in this chapter we restrict ourselves to the first conversion approach because of its simplicity.

Table A.3 shows the same statistics from the Table A.2 of flat datasets, plus average fanout, and the average depth of nodes. The average fanout is computed as the average number of children over all non-leaf nodes, and the average depth is computed over all the nodes in the hierarchy. For the average depth, we define depth of root nodes as one and the depth of non-root nodes as the depth of its parent plus one. The labels dependency is calculated similarly to the flat datasets case, however, instead of considering all possible label pairs, we consider only pairs of labels which are siblings.

A.5.2 Scalability

In our experiments, we use the *filter* method for feature selection because it is a simple, popular, and highly scalable method. As discussed in Section A.2.4, the filter method basically consists of the computation of relevance measures for every feature w.r.t. every label, and the ranking of these features by their relevance value.

For non-hierarchical multilabel classification problems, the computation of relevance measures on the local and global approach have the sample complexity of $O(|C| * |F| * |D|)$, where $|C|$ is the number of classes, $|F|$ the number of features (assumed to be binary) and $|D|$ the number of instances. This is because the relevance measure needs to be computed for every pair of a label and a feature, on all instances in the dataset. In the local case, this is done for a binary class on each transformed dataset ($|C|$ in the case of binary relevance). In the global case with aggregation method, the same computation has to be performed before aggregating the relevance values of each class into global relevance values, while in the multidimensional case, instead of 2-dimensional distributions, distributions over the $|C|$ needs to be computed. Therefore, in any of the cases mentioned before, the computation time grows linearly with the number of classes. The major difference is that the local approach has the disadvantage of having to sort the features by their relevance measure values $|C|$ times, while in the global approach it is performed only once. However, this does not change the overall complexity of the whole feature selection process: the additional effort of sorting the features is $|F| * log|F|$, and for almost every dataset, $log|F| < |D|$ holds (otherwise, the dataset would be very degenerate, having a few orders of magnitude more features than instances).

For hierarchical classification with siblings examples selection policy, the local feature selection is assumed to scale better since, for labels deeper in the hierarchy, the binary relevance measures are calculated only on a subset of the data. Typically, the number of labels in the lower levels of the hierarchy is higher, and the lower the level of the label node, the smaller is the subset of instances. Assuming that the labels hierarchy has a fanout b and the instances have a single path only, the complexity for computing the relevance measures in the local approach would be $O(b*log_b(|C|)*|D|)$, since the average transformed dataset size would be $|D|*(b*log_b(|C|))/|C|$ instead of $|D|$. The average size of the transformed datasets also increases with the cardinality of the multilabel dataset. However, for simplicity and because the cardinality is normally low in most real datasets, we ignore this factor when calculating the average size.

It is important to notice that the use of feature selection can reduce the overall runtime, as the multilabel classifier benefits from the dimensionality reduction. However, depending on the local classifier used and the feature selection method, the cost of performing the feature selection may be higher than the benefit of caused by the dimensionality reduction. Therefore, when employing a simple and highly scalable local classifier, such as Naïve Bayes, the overall runtime for some datasets without feature selection may be lower than with feature selection.

A.5.3 Results

Figure A.1 shows a comparison between the local and global feature selection approaches for different numbers of selected features. The results reported were obtained with J48 as local classifier, the mean aggregation approach for the global feature selection, and ranking based feature selection with information gain as relevance measure. The reported runtime consists of the sum of feature selection and training time. The results reported in this section were obtained with 5-fold cross-validation. Because of space constraints, we show diagrams for only nine of the evaluated datasets[9].

The plots show that the local feature selection approach performs consistently better than the global approach, with a similar runtime for flat multilabel classification, and with significantly lower runtime for hierarchical. The difference in runtime for the hierarchical case is due to the siblings negative examples selection policy, as discussed above.

The difference in the micro F_1-measure is notably higher for smaller sets of selected features, where the average Jaccard index between locally and globally selected features is lower. Genbase is an example for a dataset which does not

[9] The complete set of plots can be found at `http://data.dws.informatik.uni-mannheim.de/hmctp/plots/report.pdf`

			J48		NaiveBayes		AdaBoost		kNN		LibSVM	
Dataset	D_1	D_2	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}
Arts1	0.339	0.237	1.553	1.736	1.135	1.539	1.343	1.549	1.488	1.585	1.572	1.630
bibtex	0.196	0.225	1.159	1.465	1.203	1.321	1.122	1.337	1.057	1.091	1.164	1.773
birds	0.308	0.284	0.996	1.143	1.208	1.174	1.156	1.307	0.996	0.971	1.634	1.218
Business1	0.265	0.238	1.027	1.423	0.998	1.435	1.007	1.306	1.034	1.359	1.023	1.480
CAL500	0.278	0.250	1.008	1.030	1.022	1.025	1.002	1.018	1.026	1.036	0.988	0.977
Computers1	0.242	0.233	1.110	1.448	1.192	1.593	1.079	1.701	1.088	1.287	1.109	1.606
Corel5k	0.168	0.225	2.223	1.054	1.848	1.358	2.942	1.015	1.548	1.093	1.109	1.006
delicious	0.146	0.227	1.736	1.312	1.352	1.543	2.524	1.474	1.297	0.948	2.099	1.518
Education1	0.285	0.245	1.491	1.666	1.062	1.464	1.297	1.422	1.418	1.650	1.579	1.780
emotions	0.386	0.365	1.022	1.044	1.008	1.011	1.030	1.053	1.021	1.023	0.953	0.937
enron	0.162	0.230	1.267	1.326	1.176	1.285	1.161	1.189	1.222	1.228	1.274	1.177
Entertainment1	0.357	0.256	1.754	1.732	1.161	1.539	1.836	1.770	1.512	1.440	1.811	1.744
flags	0.484	0.440	0.996	0.960	1.003	1.000	0.991	0.974	0.996	0.986	0.984	0.965
genbase	0.766	0.729	1.003	1.029	1.008	1.046	1.005	1.026	1.005	1.030	1.004	1.029
Health1	0.439	0.254	1.101	1.313	1.134	1.307	0.978	1.202	1.084	1.315	1.081	1.249
imdb	0.197	0.255	0.989	1.015	1.189	1.205	1.000	1.000	0.974	0.987	0.936	0.896
mediamill	0.117	0.238	1.026	1.063	0.999	1.149	1.025	1.045	1.013	1.013	1.028	1.042
medical	0.435	0.242	1.007	1.042	1.076	1.078	1.016	1.042	0.975	1.032	1.005	1.025
rcv1subset1	0.362	0.229	1.445	1.564	1.485	1.688	1.353	1.467	1.237	1.378	1.138	1.011
rcv1subset2	0.346	0.229	1.213	1.579	1.592	1.814	1.278	1.511	1.198	1.532	1.217	1.031
rcv1subset3	0.339	0.228	1.216	1.478	1.685	1.763	1.291	1.417	1.227	1.476	1.190	1.010
rcv1subset4	0.379	0.229	1.245	1.455	1.562	1.752	1.385	1.436	1.210	1.439	1.452	1.126
rcv1subset5	0.344	0.228	1.197	1.562	1.482	1.778	1.324	1.494	1.216	1.110	1.542	1.052
Recreation1	0.263	0.242	1.409	1.614	1.609	1.804	1.358	1.656	1.359	1.487	1.428	1.697
Reference1	0.464	0.242	1.167	1.210	1.246	1.142	1.095	1.095	1.159	1.285	1.141	1.185
scene	0.258	0.343	1.258	1.293	1.145	1.135	1.366	1.491	1.158	1.153	1.518	1.549
Science1	0.347	0.231	1.616	1.717	1.157	1.753	1.605	1.605	1.611	1.914	1.744	1.621
slashdot	0.381	0.239	1.548	1.289	1.551	1.469	1.213	1.129	1.408	1.319	1.726	1.356
Social1	0.334	0.245	1.022	1.759	1.239	1.666	0.974	1.480	1.029	1.582	1.035	1.765
Society1	0.301	0.251	1.404	2.307	1.313	1.579	1.504	2.007	1.341	1.837	1.374	2.168
tmc2007	0.188	0.241	1.157	1.757	1.145	1.819	1.153	1.843	1.155	1.546	1.157	1.970
yeast	0.164	0.267	1.021	1.056	1.017	1.040	1.030	1.085	0.997	1.003	1.052	1.185
AVG	0.314	0.269	1.262	1.389	1.250	1.415	1.295	1.348	1.189	1.285	1.283	1.337

Table A.4: Comparison of local and global feature selection with *mean* aggregation on flat multilabel datasets

significantly profit from the local feature selection. Its average Jaccard index show that the locally selected features are not very different from the globally selected ones, converging very rapidly to a average Jaccard index of 1. Only for small numbers of selected features, the local approach shows an improvement in comparison to the global approach.

Tables A.4 and A.6 show a summary of the results for the datasets used in the experiments. We computed curves as in Figure A.1 for all the datasets, local classifiers, i.e., micro/macro average F-measure graphed against the ratio of features selected. Instead of showing the plots as in Figure A.1, we report the ratio of the area under the curves of local and global feature selection for the *micro-averaged*

			J48		NaiveBayes		AdaBoost		kNN		LibSVM	
Dataset	D_1	D_2	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}
aifb	0.401	0.383	1.024	1.165	1.187	1.135	1.025	1.234	1.024	1.180	1.023	1.213
cellcycle	0.171	0.224	1.174	1.004	1.092	1.013	1.338	1.006	1.022	1.006	1.501	1.002
church	0.475	0.279	1.416	1.001	1.654	1.004	1.658	1.003	1.034	1.042	1.651	1.002
dbpedia-yago	0.271	0.299	1.010	1.146	1.005	1.210	1.004	1.130	1.014	0.901	0.952	0.336
derisi	0.128	0.228	0.992	1.001	1.095	1.011	1.013	1.002	1.026	0.978	0.609	0.998
eisen	0.085	0.225	1.091	1.001	1.098	0.997	1.178	0.994	1.007	1.012	1.030	1.001
gasch2	0.232	0.224	1.049	0.998	1.014	1.018	1.034	1.002	0.987	1.041	1.045	0.988
mutagenesis	0.203	0.333	1.194	1.130	1.201	1.127	1.190	1.118	1.196	1.139	0.781	1.021
NELL	0.228	0.248	1.022	1.085	1.019	1.036	1.000	1.023	1.019	1.092	1.021	1.040
pheno	0.326	0.256	1.177	1.003	1.094	1.002	1.709	1.007	1.055	1.010	1.017	1.000
struc	0.198	0.224	1.032	0.970	1.088	0.892	1.075	0.998	1.012	0.976	1.581	1.002
wikidata	0.215	0.389	1.004	1.027	1.022	1.150	1.004	1.023	1.006	1.033	1.003	1.009
AVG	0.244	0.276	1.099	1.044	1.131	1.050	1.186	1.045	1.033	1.034	1.101	0.967

Table A.5: Comparison of local and global feature selection with *mean* aggregation on hierarchical multilabel datasets

F_1*-measure* ($R_{\mu F_1}$) and *macro-averaged* F_1*-measure* (R_{mF_1}) . Values greater than 1 for both $R_{\mu F_1}$ and R_{mF_1} means that the local approach outperforms the global approach. We report the normalized area under the curve of the D_1 and D_2 measures. The closer to 1, the more the local feature sets are similar to the global set. The last row in the tables (AVG) show the average value of the measures over all the datasets evaluated.

The averages show that that the vast majority of the reported ratios are larger than one, indicating that the local feature selection performs better overall. Depending on the method used as local classifier, the impact of the local approach can vary. Adaboost is the method which benefits the most in both flat and hierarchical datasets, while kNN benefits the least. When comparing the ratios of micro and macro-averaged F_1-measure ($R_{\mu F_1}$ and R_{mF_1}), the ratio of macro-averaged F_1-measure is higher than that of micro-averaged. This reveals that on flat datasets, the less frequent classes benefit from the local approach more than the more frequent classes. This can be explained by the fact that global feature selection approaches prefer features which are relevant to the more frequent classes, which means that, in general, the set of global features is more relevant to frequent classes than to infrequent ones.

On the hierarchical datasets, however, the $R_{\mu F_1}$ is in general greater than R_{mF_1}, indicating that more frequent classes benefit more strongly from the local feature selection. The same fact that global feature selection approaches prefer features relevant to more frequent classes should apply in the hierarchical case as well. One possible explanation is that less frequent classes are at lower levels of the hierarchy, and errors for these classes can be caused by classification errors in the ascendant classes, since we used the top-down approach in our experiments. That means the improvement in the local classifier of a leaf-node class, for example, is

			J48		NaiveBayes		AdaBoost		kNN		LibSVM	
Dataset	D_1	D_2	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}
Arts1	0.339	0.237	1.525	1.690	1.176	1.543	1.315	1.450	1.478	1.600	1.537	1.574
bibtex	0.167	0.225	1.119	1.426	1.084	1.247	1.087	1.257	0.987	1.066	1.127	1.670
birds	0.282	0.284	1.048	1.209	1.346	1.343	1.199	1.317	1.022	1.038	1.654	1.270
Business1	0.265	0.238	1.029	1.444	0.994	1.408	1.003	1.240	1.041	1.452	1.021	1.463
CAL500	0.250	0.250	1.010	1.022	1.024	1.042	1.005	1.006	1.014	1.002	0.987	0.957
Computers1	0.242	0.233	1.242	1.824	1.431	1.794	1.178	1.644	1.249	1.660	1.244	2.051
Corel5k	0.113	0.225	1.606	1.033	1.820	1.554	1.130	1.004	1.357	1.053	2.972	1.002
delicious	0.146	0.227	1.872	1.297	1.570	1.642	1.895	1.134	1.650	1.273	1.989	1.295
Education1	0.285	0.245	1.583	1.649	0.853	1.291	1.431	1.461	1.569	1.874	1.622	1.844
emotions	0.380	0.365	1.032	1.047	1.007	1.012	1.040	1.067	1.022	1.023	0.953	0.936
enron	0.150	0.230	1.192	1.280	1.147	1.331	1.082	1.146	1.162	1.232	1.273	1.155
Entertainment1	0.357	0.256	1.703	1.729	1.088	1.472	1.851	1.906	1.492	1.463	1.737	1.728
flags	0.452	0.440	0.998	0.971	1.002	1.005	0.991	0.982	0.989	0.976	0.982	0.968
genbase	0.764	0.729	1.005	1.024	1.005	1.031	1.005	1.022	1.004	1.027	1.004	1.029
Health1	0.439	0.254	1.072	1.298	1.129	1.237	0.959	1.187	1.044	1.338	1.059	1.237
imdb	0.183	0.255	1.045	1.057	1.286	1.293	1.000	1.000	1.020	1.038	0.927	0.879
mediamill	0.144	0.228	1.032	0.866	0.968	0.958	1.007	0.450	0.995	1.045	1.028	1.042
medical	0.300	0.242	0.985	1.009	1.006	0.960	1.008	1.026	0.929	0.976	0.993	1.006
rcv1subset1	0.362	0.229	1.535	1.603	1.893	1.999	1.556	1.513	1.440	1.563	1.005	1.001
rcv1subset2	0.346	0.229	1.268	1.614	2.027	2.180	1.335	1.515	1.333	1.671	1.003	0.974
rcv1subset3	0.339	0.228	1.229	1.502	2.028	2.089	1.300	1.408	1.361	1.654	0.972	0.965
rcv1subset4	0.379	0.229	1.220	1.430	1.831	1.985	1.299	1.385	1.326	1.559	1.064	1.050
rcv1subset5	0.344	0.228	1.140	1.503	1.822	1.947	1.313	1.487	1.284	1.560	1.052	0.998
Recreation1	0.263	0.242	1.460	1.828	1.548	1.925	1.362	1.670	1.422	1.724	1.456	1.810
Reference1	0.464	0.242	1.164	1.177	1.276	1.132	1.085	1.034	1.138	1.207	1.132	1.137
scene	0.244	0.343	1.146	1.178	1.094	1.104	1.174	1.274	1.091	1.093	1.207	1.289
Science1	0.347	0.231	1.452	1.681	1.222	1.660	1.424	1.515	1.437	1.777	1.469	1.504
slashdot	0.381	0.239	1.576	1.293	1.533	1.470	1.214	1.125	1.394	1.295	1.706	1.342
Social1	0.334	0.245	1.041	1.965	1.291	1.736	0.959	1.515	1.058	1.850	1.044	2.000
Society1	0.301	0.251	1.456	2.207	1.245	1.540	1.475	1.873	1.392	1.814	1.448	2.064
tmc2007	0.188	0.241	1.161	1.676	1.133	1.799	1.152	1.749	1.160	1.580	1.145	1.748
yeast	0.168	0.267	1.020	1.059	1.009	1.049	1.038	1.123	0.999	1.003	1.047	1.166
AVG	0.304	0.269	1.249	1.393	1.309	1.462	1.215	1.296	1.214	1.359	1.277	1.317

Table A.6: Comparison of local and global feature selection with *max* aggregation on flat multilabel datasets

limited to the cases where the predictions of all the local classifiers of ascendant classes are correct.

We also compared transformation methods with local feature selection and their correspondent adaptation methods over all hierarchical and flat multilabel datasets. The adaptation methods used global feature selection since the local approach is not applicable. We performed the comparison for MLC4.5 and MLkNN varying the number of selected features. We also tried to use AdaboostMH implementation in MULAN, however, the classifier did not seem to work properly, and the results remained constant over all the different sets of selected features used. The results obtained were not conclusive about whether the adaptation or transfor-

			J48		NaiveBayes		AdaBoost		kNN		LibSVM	
Dataset	D_1	D_2	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}	$R_{\mu F_1}$	R_{mF_1}
aifb	0.346	0.383	1.024	1.166	1.187	1.135	1.025	1.235	1.024	1.180	1.023	1.213
cellcycle	0.163	0.224	1.115	1.002	1.087	1.002	1.385	1.006	1.008	1.005	1.141	1.000
church	0.344	0.279	1.031	0.999	1.226	1.010	1.068	1.000	0.845	1.049	1.141	1.00
dbpedia-yago	0.259	0.299	1.011	1.160	0.998	1.231	1.002	1.121	1.014	0.898	0.953	0.331
derisi	0.151	0.228	1.164	1.002	1.126	1.010	1.098	1.000	1.031	0.990	0.899	1.002
eisen	0.086	0.225	1.117	0.999	1.137	1.001	1.268	0.995	1.019	1.013	1.090	1.003
gasch2	0.192	0.224	1.077	0.997	1.036	1.025	1.079	1.003	1.007	1.034	1.037	0.987
mutagenesis	0.150	0.333	1.194	1.130	1.201	1.127	1.191	1.118	1.196	1.139	0.782	1.022
nell	0.198	0.244	1.021	1.072	1.017	1.027	1.000	1.024	1.019	1.077	1.018	1.034
pheno	0.242	0.256	1.150	1.002	1.306	0.995	1.229	1.003	1.169	1.005	1.202	1.000
struc	0.187	0.224	1.082	0.962	1.053	0.840	1.045	0.998	1.046	0.986	1.260	1.001
wikidata	0.206	0.389	1.009	1.027	1.019	1.148	1.004	1.020	1.011	1.031	1.005	1.009
AVG	0.210	0.276	1.083	1.043	1.116	1.046	1.116	1.044	1.032	1.034	1.045	0.967

Table A.7: Comparison of local and global feature selection with *max* aggregation on. popular hierarchical multilabel datasets

mation methods have a better general performance, with no approach consistently outperforming the other. In order to draw any conclusions, a study dedicated to the comparison between adaptation methods and transformation methods with local feature selection would be required, which, however, is out of scope of this work.

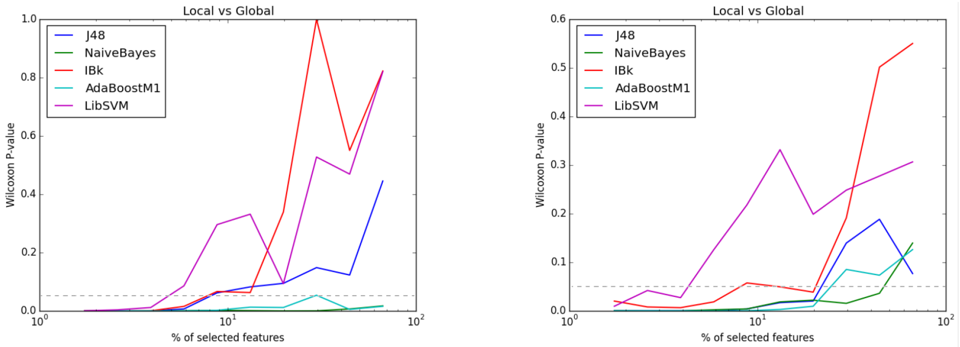
A.5.4 Statistical Analysis

After running all the experiments presented in this section, we need to test the significance of the results. For that we perform the Wilcoxon signed-rank test (212) comparing micro-averaged and macro-averaged F_1-measure of local and global feature selection approaches with transformation methods. Figure A.2 shows the p-value of the Wilcoxon test over the 44 datasets reported (flat and hierarchical) for the five different local classifiers. This is done for different portions of the features selected, which is represented as percentage of total number of features in the horizontal axis.

For p-values under 0.05 the difference between local and global feature selection methods is statistically significant according to the Wilcoxon Test. We can observe that, for all local classifiers, the difference is highly significant with a p-value far below the 0.05 line for smaller numbers of features. In particular, for Naïve Bayes, this difference is the most significant amongst the evaluated local classifiers. Adaboost, LibSVM, and J48 also benefit significantly from the local feature selection approach, while IBk profits the least.

	J48	Naive Bayes	AdaBoost	kNN	LibSVM	AVG
corr(D_1,μF_1)	-0.5364	-0.4791	-0.5348	-0.5273	-0.3909	-0.4937
corr(D_2,μF_1)	-0.5425	-0.4880	-0.5264	-0.5405	-0.4120	-0.5019
corr(D_1,mF_1)	-0.6069	-0.5356	-0.5769	-0.5190	-0.4580	-0.5393
corr(D_2,mF_1)	-0.6341	-0.5427	-0.5849	-0.4982	-0.4905	-0.5501
corr(D_1,$hamm$)	0.3808	0.2178	0.3576	0.3778	0.4476	0.3563
corr(D_2,$hamm$)	0.4119	0.2209	0.3457	0.4299	0.4955	0.3808

Table A.8: Correlations between D_1 and D_2 measures and the ratio between global and local feature selection approaches for different evaluation measures



(a) Micro-averagedF-Measure

(b) Macro-averagedF-Measure

Figure A.2: Wilcoxon test on flat and hierarchical multilabel datasets

The results indicate that for all classifiers on a small set of selected features, the difference is the most significant, while for larger portions of selected features the significance is slightly lower. With that, we can confirm that when performing feature selection on transformation methods, the local approach is a better choice than the global method, especially when the portion of selected features is small. We also point out that the choice of local classifier can influence the benefits of local over global feature selection methods.

Another investigation we made is how strong the correlation between the variance of local feature selection and the performance gain over global selection is. The hypothesis is that for problems where there are strong differences in the best local feature sets, local feature selection will lead to a more significant performance improvement. In order to test that hypothesis, we measure the correlation of D_1 and D_2, which capture the variety of features in the different local sets, with the ratio between the performance measures of classifiers with local and global feature selection approaches. For a given dataset and local classifier we compute the values of D_1, D_2, and the values of μF_1, mF_1 and $hamm$ with local feature selection divided by the correspondent measure values with global feature selection for the different numbers of selected features k. Then the correlation is calculated for every dataset and with the different local classifiers considered in the experiments.

Table A.8 shows the correlation values across all the datasets with different local classifiers, and, in the last column, the correlation over all classifiers and datasets.

For μF_1 and mF_1, ratios greater than one mean that the local approach performed better, for $hamm$ ratios less than one mean that the local approach performed better. For both measures D_1 and D_2, the smaller the value, the more distinct the local feature sets are from the global feature sets. Therefore, the negative values of corr(D_1,μF_1), corr(D_1,mF_1), corr(D_2,μF_1) and corr(D_2,mF_1), and the positive values of corr(D_1,$hamm$) and corr(D_2,$hamm$) show that the more local and global feature sets differ, the better the local approach will be in comparison to the global approach.

In all cases, the correlations are significant, which confirms the original hypothesis. Furthermore, we have observed that the correlation between D_1 and D_2 is 0.957, i.e., both measures are essentially very similar in measuring the variety of the local feature selection.

A.6 Conclusion and Future Work

In this chapter, we have presented an experimental comparison of global and local feature selection methods on transformation methods for multilabel classification with flat and hierarchical labels. Although transformation methods are very popular, and allow the feature selection to be performed locally on each transformed dataset, this alternative has not been very extensively explored in the literature. Our experiment results indicate that the local approach performs consistently better than the global approach in terms of predictive performance. Both approaches have similar runtimes and scalability on flat multilabel classification, and for hierarchically structured labels, the local approach scales better than the global approach. Based on these results, the local feature selection approach is considered superior to the global approach. When comparing the local feature selection approach with transformation methods to adaptation methods with global approaches using a global feature selection method, the results are not generally conclusive. However, for many of the datasets, the local approach also performs better in that case.

So far, we have only considered binary relevance as a transformation technique. Performing a similar comparison with other transformation methods, such as classifier chains, would be interesting for future work. Furthermore, it would be interesting if it is possible to compile a better global feature set for adaptation methods, using local feature selection on a number of transformed problems. Another possibility which the use of transformation brings is the use of different local classifiers for different transformed subproblems.

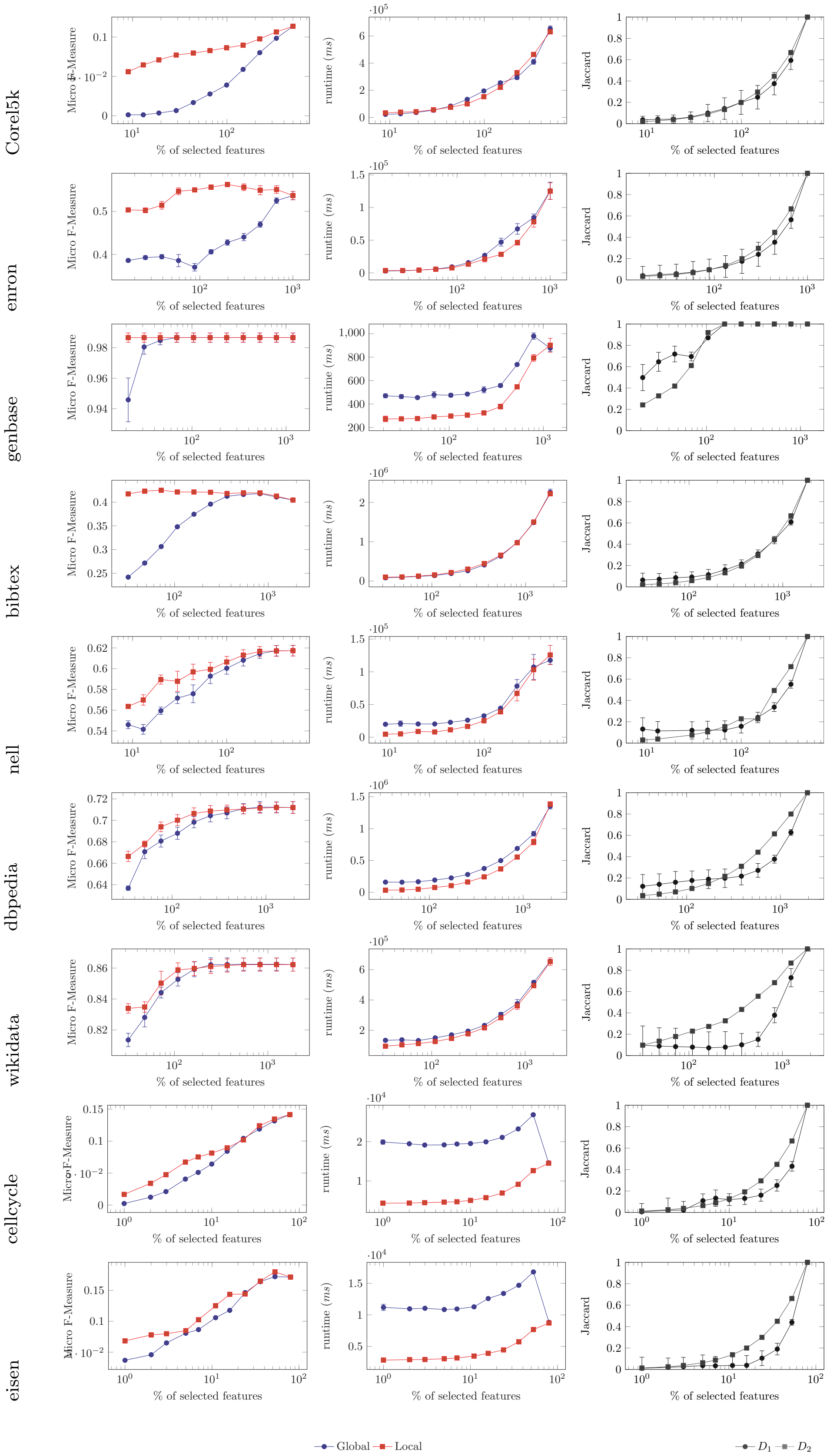


Figure A.1: Global vs local feature selection comparison with J48 and measures D_1 and D_2.

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB, pages 487–499, 1994.

[3] Georgia Albuquerque, Thomas Löwe, and Marcus Magnor. Synthetic generation of high-dimensional datasets. *IEEE Transactions on Visualization and Computer Graphics (TVCG, Proc. Visualization / InfoVis)*, 17(12):2317–2324, Dec 2011.

[4] Renzo Angles, Peter Boncz, Josep Larriba-Pey, Irini Fundulaki, Thomas Neumann, Orri Erling, Peter Neubauer, Norbert Martinez-Bazan, Venelin Kotsev, and Ioan Toma. The linked data benchmark council: A graph and rdf industry benchmarking effort. *SIGMOD Rec.*, 43(1):27–31, May 2014.

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

[6] Dörthe Arndt, Ben De Meester, Anastasia Dimou, Ruben Verborgh, and Erik Mannens. Using rule-based reasoning for RDF validation. In Stefania Costantini, Enrico Franconi, William Van Woensel, Roman Kontchakov, Fariba Sadri, and Dumitru Roman, editors, *Proceedings of the International Joint Conference on Rules and Reasoning*, volume 10364 of *Lecture Notes in Computer Science*, pages 22–36. Springer, July 2017.

[7] Sören Auer, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Amrapali Zaveri. Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–90, 2013.

[8] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniel Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[9] Liviu Badea and Shan hwei Nienhuys-Cheng. A refinement operator for description logics. In *Proc. of the Int. Conf. on Inductive Logic Programming (ILP)*, volume 1866 of *LNAI*, pages 40–59. Springer, 2000.

[10] Wei Bi and James T. Kwok. Multi-label classification on tree- and dag-structured hierarchies. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 17–24, New York, NY, USA, 2011. ACM.

[11] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data – The Story So Far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

[12] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics*, 7(3):154–165, 2009.

[13] Hendrik Blockeel, Luc De Raedt, and Jan Ramong. Top-down induction of clustering trees. In *In Proceedings of the 15th International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.

[14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In Neil D. Lawrence and Mark A. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012*, volume 22 of *JMLR Proceedings*, pages 127–135. JMLR.org, 2012.

[15] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[16] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

[17] Fabian Both, Steffen Thoma, and Achim Rettinger. Cross-modal knowledge transfer: Improving the word embedding of apple by looking at oranges. 2017.

[18] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771, March 2004.

[19] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[20] F. Briggs, Yonghong Huang, R. Raich, K. Eftaxias, Zhong Lei, W. Cukierski, S.F. Hadley, A. Hadley, M. Betts, X.Z. Fern, J. Irvine, L. Neal, A. Thomas, G. Fodor, G. Tsoumakas, Hong Wei Ng, Thi Ngoc Tho Nguyen, H. Huttunen, P. Ruusuvuori, T. Manninen, A. Diment, T. Virtanen, J. Marzat, J. Defretin, D. Callender, C. Hurlburt, K. Larrey, and M. Milakov. The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on*, pages 1–8, Sept 2013.

[21] Sergey Brin, Rajeev Rastogi, and Kyuseok Shim. Mining optimized gain rules for numeric attributes. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144. ACM Press, 1999.

[22] Florian Brucker, Fernando Benites, and Elena Sapozhnikova. *An Empirical Comparison of Flat and Hierarchical Performance Measures for Multi-Label Classification with Hierarchy Extraction*, pages 579–589. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[23] Lorenz Bühmann, Daniel Fleischhacker, Jens Lehmann, André Melo, and Johanna Völker. Inductive lexical learning of class expressions. In Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen, editors, *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, volume 8876 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2014.

[24] Lorenz Bühmann and Jens Lehmann. Universal owl axiom enrichment for large knowledge bases. In *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management*, EKAW'12, pages 57–71, Berlin, Heidelberg, 2012. Springer-Verlag.

[25] Lorenz Bühmann and Jens Lehmann. Pattern based knowledge base enrichment. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, 2013.

[26] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *In AAAI*, 2010.

[27] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM, 2010.

[28] Rose Catherine and William Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 325–332, New York, NY, USA, 2016. ACM.

[29] Ricardo Cerri, Gisele L. Pappa, André Carlos Ponce de Leon Ferreira de Carvalho, and Alex Alves Freitas. An extensive evaluation of decision tree-based hierarchical multilabel classification methods and performance measures. *Computational Intelligence*, 31(1):1–46, 2015.

[30] Nicolò Cesa-bianchi, Luca Zaniboni, and Michael Collins. Incremental algorithms for hierarchical classification. In *Journal of Machine Learning Research*, pages 31–54. MIT Press, 2004.

[31] Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Chris Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. ACL – Association for Computational Linguistics, October 2014.

[32] Dipak Chaudhari, Om P. Damani, and Srivatsan Laxman. Lexical co-occurrence, statistical significance, and word association. In *EMNLP*, pages 1058–1068. ACL, 2011.

[33] Sanjay Chawla and Aristides Gionis. k-means-: A unified approach to clustering and outlier detection. In *Proceedings of the 13th SIAM International Conference on Data Mining, Austin, Texas, USA.*, pages 189–197. SIAM, 2013.

[34] Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Roger Granada, Valentina Ivanova, Ernesto Jiménez-Ruiz, et al. Results of the ontology alignment evaluation initiative 2015. In *10th ISWC workshop on ontology matching (OM)*, pages 60–115, 2015.

[35] Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1999–2008, New York, NY, USA, 2013. ACM.

[36] A. Clare and R. D. King. Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, 19:42–49, 2003.

[37] Amanda Clare and Ross D. King. Knowledge discovery in multi-label phenotype data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD'01, pages 42–53, London, UK, 2001. Springer-Verlag.

[38] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Global RDF vector space embeddings. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, pages 190–207, 2017.

[39] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Global rdf vector space embeddings. In *International Semantic Web Conference*, 2017. to appear.

[40] William W. Cohen and Haym Hirsh. Learnability of description logics. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*. ACM Press, 1992.

[41] William W. Cohen and Haym Hirsh. Learning the CLASSIC description logic. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 121–133. Morgan Kaufmann, 1994.

[42] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[43] Eduardo P. Costa, Ana C. Lorena, André C. P. L. F. Carvalho, Alex A. Freitas, and Nicholas Holden. Comparing several approaches for hierarchical classification of proteins with decision trees. In *Proceedings of the 2nd Brazilian Conference on Advances in Bioinformatics and Computational Biology*, BSB'07, pages 126–137, Berlin, Heidelberg, 2007. Springer-Verlag.

[44] E.P. Costa, A.C. Lorena, A.C.P.L.F. Carvalho, and A.A. Freitas. A review of performance evaluation measures for hierarchical classifiers. In C. Drummond, W. Elazmeh, N. Japkowicz, and S.A. Macskassy, editors, *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop, AAAI Technical Report WS-07-05*, pages 182–196. AAAI Press, July 2007.

[45] Om P. Damani. Improving pointwise mutual information (pmi) by incorporating significant co-occurrence. *CoRR*, abs/1307.0596, 2013.

[46] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[47] Gael de Lannoy, Damien Francois, and Michel Verleysen. Class-specific feature selection for one-against-all multiclass svms. In *ESANN*, 2011.

[48] Jeremy Debattista, Christoph Lange, and Sören Auer. A preliminary investigation towards improving linked data quality using distance-based outlier detection. In Yuan-Fang Li, Wei Hu, Jin Song Dong, Grigoris Antoniou, Zhe Wang, Jun Sun, and Yang Liu, editors, *Semantic Technology - 6th Joint International Conference, JIST 2016, Singapore, Singapore, November 2-4, 2016, Revised Selected Papers*, volume 10055 of *Lecture Notes in Computer Science*, pages 116–124. Springer, 2016.

[49] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Saso Dzeroski. Hierarchical annotation of medical images. *Pattern Recognition*, 44(10-11):2436–2449, 2011.

[50] Sotiris Diplaris, Grigorios Tsoumakas, Pericles A. Mitkas, and Ioannis P. Vlahavas. Protein classification with multiple algorithms. In Panayiotis Bozanis and Elias N. Houstis, editors, *Panhellenic Conference on Informatics*, volume 3746 of *Lecture Notes in Computer Science*, pages 448–456. Springer, 2005.

[51] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610, New York, NY, USA, 2014. ACM.

[52] Gauthier Doquire and Michel Verleysen. Feature selection for multi-label classification problems. In Joan Cabestany, Ignacio Rojas, and Gonzalo Joya Caparrós, editors, *Advances in Computational Intelligence - 11th International Work-Conference on Artificial Neural Networks, IWANN 2011, Torremolinos-Málaga, Spain, June 8-10, 2011, Proceedings, Part I*, volume 6691 of *Lecture Notes in Computer Science*, pages 9–16. Springer, 2011.

[53] P. Duygulu, Kobus Barnard, J. F. G. de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pages 97–112, London, UK, UK, 2002. Springer-Verlag.

[54] Roman Eisner, Brett Poulin, Duane Szafron, Paul Lu, and Russ Greiner. Improving protein function prediction using the hierarchical structure of the gene ontology. In *Proc. IEEE CIBCB*, 2005.

[55] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS-01)*, pages 681–687, 2002.

[56] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing Wikidata to the linked data web. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandečić, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *Proceedings of the 13th International Semantic Web Conference (ISWC'14)*, volume 8796 of *LNCS*, pages 50–65. Springer, 2014.

[57] Tiziano Fagni and Fabrizio Sebastiani. On the selection of negative examples for hierarchical text categorization. In *In Proceedings of The 3rd Language Technology Conference*, pages 24–28, 2007.

[58] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. DL-FOIL: Concept learning in description logics. In F. Zelezný and N. Lavrac, editors, *Proc. of the 18th Int. Conf. on Inductive Logic Programming (ILP)*, volume 5194 of *LNAI*, pages 107–121. Springer, 2008.

[59] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Induction of concepts in web ontologies through terminological decision trees. In José L. Balcázar et al., editors, *Proceedings of ECML PKDD 2010, Part I*, volume 6321 of *LNCS/LNAI*, pages 442–457. Springer, 2010.

[60] Henning Fernau. Algorithms for learning regular expressions from positive data. *Information and Computation*, 207(4):521 – 541, 2009.

[61] J.R. Firth. A synopsis of linguistic theory 1930-1955. *Studies in linguistic analysis*, pages 1–32, 1957.

[62] Daniel Fleischhacker, Heiko Paulheim, Volha Bryl, Johanna Völker, and Christian Bizer. Detecting errors in numerical linked data using cross-checked outlier detection. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web – ISWC 2014*, pages 357–372, Cham, 2014. Springer International Publishing.

[63] J.L. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[64] A.A. Freitas and Andre C.P.F.L. de Carvalho. *A Tutorial on Hierarchical Classification with Applications in Bioinformatics.*, volume Research and Trends in Data Mining Technologies and Applications, chapter VII, pages 182–196. Idea Group, January 2007.

[65] Luis Galárraga, Simon Razniewski, Antoine Amarilli, and Fabian M. Suchanek. Predicting completeness in knowledge bases. *CoRR*, abs/1612.05786, 2016.

[66] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW 2013, Rio de Janeiro, Brazil, 2013*, pages 413–422. ACM, 2013.

[67] Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. Automatic typing of dbpedia entities. In *The 11th International Semantic Web Conference (ISWC2012)*, pages 65–81, 2012.

[68] Alberto García-Durán and Mathias Niepert. KBLRN : End-to-end learning of knowledge base representations with latent, relational, and numerical features. *CoRR*, abs/1709.04676, 2017.

[69] Matt Gardner and Tom M. Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1488–1498. The Association for Computational Linguistics, 2015.

[70] José Emilio Labra Gayo, Eric Prud'hommeaux, Harold R. Solbrig, and Iovka Boneva. Validating and describing linked data portals using shapes. *CoRR*, abs/1701.08924, 2017.

[71] Andrea Giovanni, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. Type inference through the analysis of wikipedia links. In *Linked Data on the Web (LDOW)*, 2012.

[72] Bart Goethals and Jan Van den Bussche. *Relational Association Rules: Getting Warmer*, pages 125–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[73] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semant.*, 3(2-3):158–182, October 2005.

[74] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[75] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[76] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011.

[77] Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *International Journal on Semantic Web and Information Systems*, 5(2):25–48, 2009.

[78] Shamsul Huda, John Yearwood, and Andrew Stranieri. Hybrid wrapper-filter approaches for input feature selection using maximum relevance-minimum redundancy and artificial neural network input gain measurement approximation (annigma). In *Proceedings of the Thirty-Fourth Australasian Computer Science Conference - Volume 113*, ACSC '11, pages 43–52, Darlinghurst, Australia, Australia, 2011. Australian Computer Society, Inc.

[79] Luigi Iannone, Ignazio Palmisano, and Nicola Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

[80] Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3167–3175. Curran Associates, Inc., 2012.

[81] Qiu Ji, Zhiqiang Gao, and Zhisheng Huang. Reasoning with noisy semantic data. In *The Semanic Web: Research and Applications (ESWC 2011), Part II*, pages 497–502, 2011.

[82] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. *CoRR*, abs/1705.10744, 2017.

[83] Dimitris Karampinas and Peter Triantafillou. Crowdsourcing taxonomies. In Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications*, pages 545–559, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[84] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *In: Proceedings of the ECML/PKDD-08 Workshop on Discovery Challenge*, 2008.

[85] Kenji Kira and Larry A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 129–134. AAAI Press, 1992.

[86] Svetlana Kiritchenko, Stan Matwin, and A. Fazel Famili. Functional annotation of genes using hierarchical text categorization. In *in Proc. of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (held at ISMB-05*, 2005.

[87] Svetlana Kiritchenko, Stan Matwin, Richard Nock, and A. Fazel Famili. Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Proceedings of the 19th International Conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence*, AI'06, pages 395–406, Berlin, Heidelberg, 2006. Springer-Verlag.

[88] Tomáš Kliegr. Linked hypernyms: Enriching dbpedia with targeted hypernym discovery. *Web Semantics: Science, Services and Agents on the World Wide Web*, 31:59 – 69, 2015.

[89] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

[90] Aris Kosmopoulos, Georgios Paliouras, and Ion Androutsopoulos. The effect of dimensionality reduction on large scale hierarchical classification. In *Information Access Evaluation. Multilinguality, Multimodality, and Interaction - 5th International Conference of the CLEF Initiative, CLEF 2014, Sheffield, UK, September 15-18, 2014. Proceedings*, pages 160–171, 2014.

[91] Aris Kosmopoulos, Ioannis Partalas, Eric Gaussier, Georgios Paliouras, and Ion Androutsopoulos. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, 29(3):820–865, 2015.

[92] L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Probl. Inf. Transm.*, 23(1-2):95–101, 1987.

[93] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. Incorporating literals into knowledge graph embeddings. *CoRR*, abs/1802.00934, 2018.

[94] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

[95] Yannis K Labrou. Yahoo as an Ontology - using Yahoo categories to describe documents,. In *Proceedings of the 1999 ACM Conference on Information and Knowledge Management (CIKM'99)*, November 1999.

[96] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):pp. 159–174, 1977.

[97] Dustin Lange, Christoph Böhm, and Felix Naumann. Extracting structured information from wikipedia articles to populate infoboxes. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1661–1664, New York, NY, USA, 2010. ACM.

[98] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, Oct 2010.

[99] Jens Lehmann. Hybrid learning of ontology classes. In *Proc. of the 5th Int. Conference on Machine Learning and Data Mining (MLDM)*, volume 4571 of *LNCS*, pages 883–898. Springer, 2007.

[100] Jens Lehmann. Dl-learner: Learning concepts in description logics. *Journal of Machine Learning Research*, 10:2639–2642, 2009.

[101] Jens Lehmann. DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research (JMLR)*, 10:2639–2642, 2009.

[102] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9:71 – 81, 2011.

[103] Jens Lehmann and Christoph Haase. Ideal downward refinement in the EL description logic. In *Proc. of the Int. Conf. on Inductive Logic Programming*, volume 5989 of *LNCS*, pages 73–87. Springer, 2009.

[104] Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the ALC description logic. In *ILP 2007*, volume 4894 of *LNCS*, pages 147–160. Springer, 2008.

[105] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning journal*, 78(1-2):203–250, 2010.

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

[107] Jens Lehmann and Johanna Voelker. An introduction to ontology learning. In Jens Lehmann and Johanna Voelker, editors, *Perspectives on Ontology Learning*, pages ix–xvi. AKA / IOS Press, 2014.

[108] Jens Lehmann and Johanna Völker, editors. *Perspectives on Ontology Learning*. Studies on the Semantic Web. AKA Heidelberg, 2014.

[109] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December 2004.

[110] Yankai Lin, Zhiyuan Liu, and Maosong Sun. Modeling relation paths for representation learning of knowledge bases. *CoRR*, abs/1506.00379, 2015.

[111] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, pages 2181–2187, 2015.

[112] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2181–2187. AAAI Press, 2015.

[113] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag New York, Inc., New York, NY, USA, 1984.

[114] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 21:3–13, 2013.

[115] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and SašO Deroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recogn.*, 45(9):3084–3104, September 2012.

[116] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent systems*, 16(2):72–79, 2001.

[117] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *Conference on Innovative Data Systems Research*, 2015.

[118] J. Mata, J.L. Alvarez, and J.C. Riquelme. An evolutionary algorithm to discover numeric association rules. In *In Proceedings of the ACM symposium on Applied computing (SAC)*, pages 590–594, 2002.

[119] André Melo. Learning rules with numerical and categorical attributes from linked data sources. Master's thesis, Universität des Saarlandes, Saarbrücken, March 2013.

[120] André Melo and Heiko Paulheim. An approach to correction of erroneous links in knowledge graphs. In *Proceedings of the Quality Engineering Meets Knowledge Graph (QEKGraph) workshop, co-located with the International Knowledge Capture Conference, K-CAP 2017*, volume 2065 of *CEUR Workshop Proceedings*, pages 54–57. CEUR-WS.org, 2017.

[121] André Melo and Heiko Paulheim. Detection of relation assertion errors in knowledge graphs. In Óscar Corcho, Krzysztof Janowicz, Giuseppe Rizzo, Ilaria Tiddi, and Daniel Garijo, editors, *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 22:1–22:8. ACM, 2017.

[122] André Melo and Heiko Paulheim. Local and global feature selection for multilabel classification with binary relevance. *Artificial Intelligence Review*, May 2017.

[123] André Melo and Heiko Paulheim. Synthesizing knowledge graphs for link and type prediction benchmarking. In Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig, editors, *The Semantic Web*, pages 136–151, Cham, 2017. Springer International Publishing.

[124] André Melo, Heiko Paulheim, and Johanna Völker. Type prediction in rdf knowledge bases using hierarchical multilabel classification. In *Proceedings*

of the 6th International Conference on Web Intelligence, Mining and Semantics, WIMS '16, pages 14:1–14:10, New York, NY, USA, 2016. ACM.

[125] André Melo, Martin Theobald, and Johanna Völker. Correlation-based refinement of rules with numerical attributes. In William Eberle and Chutima Boonthum-Denecke, editors, *Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference, FLAIRS 2014, Pensacola Beach, Florida, May 21-23, 2014.* AAAI Press, 2014.

[126] Andre Melo, Johanna Völker, and Heiko Paulheim. Type prediction in noisy rdf knowledge bases using hierarchical multilabel classification with graph and latent features. *International Journal on Artificial Intelligence Tools*, 26(02):1760011, 2017.

[127] Stoyan Mihov and Klaus U. Schulz. Fast approximate search in large dictionaries. *Comput. Linguist.*, 30(4):451–477, December 2004.

[128] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[129] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *International Conference on Data Mining (ICDM)*, pages 306–313. IEEE, 2002.

[130] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. DBpedia SPARQL Benchmark—Performance Assessment with Real Queries on Real Data. In *ISWC 2011*, 2011.

[131] S. Muggleton. Inverse Entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[132] Stephen Muggleton. Learning from positive data. In *6th International Workshop on Inductive Logic Programming*, pages 358–376. Springer-Verlag, 1997.

[133] Emir Muñoz and Matthias Nickles. Mining cardinalities from knowledge bases. In *DEXA (1)*, volume 10438 of *Lecture Notes in Computer Science*, pages 447–462. Springer, 2017.

[134] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, March 2001.

[135] Thomas Neumann and Gerhard Weikum. The RDF-3x engine for scalable management of RDF data. *The VLDB Journal*, 19(1):91–113, February 2010.

[136] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[137] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[138] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. *CoRR*, abs/1510.04935, 2015.

[139] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 809–816, New York, NY, USA, June 2011. ACM.

[140] Shan-Hwei Nienhuys-Cheng and Ronald De Wolf. *Foundations of inductive logic programming*, volume 1228. Springer, 1997.

[141] Mathias Niepert. Discriminative gaifman models. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3405–3413. Curran Associates, Inc., 2016.

[142] David W. Opitz. Feature selection for ensembles. In *In Proceedings of 16th National Conference on Artificial Intelligence (AAAI*, pages 379–384. Press, 1999.

[143] Eyal Oren, Sebastian Gerke, and Stefan Decker. Simple algorithms for predicate suggestions using similarity and co-occurrence. In *European Semantic Web Conference (ESWC 2007)*, pages 160–174. Springer, 2007.

[144] Fernando E. Otero, Alex A. Freitas, and Colin G. Johnson. A hierarchical classification ant colony algorithm for predicting gene ontology terms. In *Proceedings of the 7th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, EvoBIO '09, pages 68–79, Berlin, Heidelberg, 2009. Springer-Verlag.

[145] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. Lshtc: A benchmark for large-scale text classification. *CoRR*, abs/1503.08581, march 2015.

[146] Heiko Paulheim. Exploiting linked open data as background knowledge in data mining. In *International Workshop on Data Mining on Linked Data (DMoLD*, 2013.

[147] Heiko Paulheim. Identifying wrong links between datasets by multidimensional outlier detection. In *WoDOOM*, pages 27–38, 2014.

[148] Heiko Paulheim. Data-driven joint debugging of the dbpedia mappings and ontology. In *European Semantic Web Conference*, pages 404–418. Springer, 2017.

[149] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[150] Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul T. Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *International Semantic Web Conference (1)*, volume 8218 of *Lecture Notes in Computer Science*, pages 510–525. Springer, 2013.

[151] Heiko Paulheim and Christian Bizer. Improving the quality of linked data using statistical distributions. *Int. J. Semant. Web Inf. Syst.*, 10(2):63–86, April 2014.

[152] Heiko Paulheim and Johannes Fürnkranz. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, page 31. ACM, 2012.

[153] Heiko Paulheim and Aldo Gangemi. Serving dbpedia with dolce–more than just adding a cherry on top. In *International Semantic Web Conference*, pages 180–196. Springer, 2015.

[154] John P. Pestian, Christopher Brew, PawełMatykiewicz, D. J. Hovermale, Neil Johnson, K. Bretonnel Cohen, and Wlodzislaw Duch. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, BioNLP '07, pages 97–104, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[155] Aleksander Pohl. Classifying the wikipedia articles in the opencyc taxonomy. In *Web of Linked Entities Workshop (WoLE 2012)*, 2012.

[156] Axel Polleres, Aidan Hogan, Andreas Harth, and Stefan Decker. Can we ever catch up with the web? *Semantic Web Journal*, 1(1,2):45–52, 2010.

[157] Jedrzej Potoniec, Piotr Jakubowski, and Agnieszka Lawrynowicz. Swift linked data miner: Mining owl 2 el class expressions directly from on-line rdf datasets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 46(1), 2017.

[158] Valentina Presutti, Francesco Draicchio, and Aldo Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames.

In *Knowledge Engineering and Knowledge Management*, volume 7603 of *LNCS*, pages 114–129. Springer, 2012.

[159] Huaqiao Qu, Shichao Zhang, Huawen Liu, and Jianmin Zhao. A multi-label classification algorithm based on label-specific features. *Wuhan University Journal of Natural Sciences*, 16(6):520–524, 2011.

[160] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[161] Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *Lecture Notes in Computer Science*. Springer, 2008.

[162] Jesse Read. A pruned problem transformation method for multi-label classification. In *In: Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS*, pages 143–150, 2008.

[163] Jesse Read, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, 88(1-2):243–272, 2012.

[164] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD'09, pages 254–269, Berlin, Heidelberg, 2009. Springer-Verlag.

[165] Jesse Read, Bernhard Pfahringer, and Geoffrey Holmes. Multi-label classification using ensembles of pruned sets. In *ICDM*, pages 995–1000. IEEE Computer Society, 2008.

[166] Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In *40th German Conference on Artificial Intelligence*, 2017. to appear.

[167] Petar Ristoski, Gerben Klaas Dirk de Vries, and Heiko Paulheim. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *ISWC*. Springer, 2016.

[168] Petar Ristoski and Heiko Paulheim. A comparison of propositionalization strategies for creating features from linked open data. In *Linked Data for Knowledge Discovery*, 2014.

[169] Petar Ristoski and Heiko Paulheim. *RDF2Vec: RDF Graph Embeddings for Data Mining*, pages 498–514. Springer International Publishing, Cham, 2016.

[170] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, WIMS '15, pages 10:1–10:6, New York, NY, USA, 2015. ACM.

[171] Sebastian Rudolph. *Acquiring Generalized Domain-Range Restrictions*, pages 32–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[172] Yvan Saeys, Thomas Abeel, and Yves Peer. Robust feature selection using ensemble feature selection techniques. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ECML PKDD '08, pages 313–325, Berlin, Heidelberg, 2008. Springer-Verlag.

[173] Ansaf Salleb-Aouissi, Christel Vrain, and Cyril Nortet. Quantminer: A genetic algorithm for mining quantitative association rules. In Manuela M. Veloso, editor, *IJCAI*, pages 1035–1040, 2007.

[174] Behrokh Samadi, Alan Cipolone, Pengyue J. Lin, Rui Xiao, Daniel R. Jeske, Douglas Holt, Carlos Rend, and Sean Cox. Development of a synthetic data set generator for building and testing information discovery systems. *Third International Conference on Information Technology*, pages 707–712, 2006.

[175] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[176] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. *Adoption of the Linked Data Best Practices in Different Topical Domains*, pages 245–260. Springer International Publishing, Cham, 2014.

[177] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. Sp2bench: A SPARQL performance benchmark. *CoRR*, abs/0806.4627, 2008.

[178] Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion, 2017.

[179] Carlos N. Silla, Jr. and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72, January 2011.

[180] Ivica Slavkov, Jana Karcheska, Dragi Kocev, Slobodan Kalajdziski, and Saso Dzeroski. Relieff for hierarchical multi-label classification. In Annalisa Appice, Michelangelo Ceci, Corrado Loglisci, Giuseppe Manco, Elio Masciari, and Zbigniew W. Ras, editors, *New Frontiers in Mining Complex Patterns - Second International Workshop, NFMCP 2013, Held in Conjunction with ECML-PKDD 2013, Prague, Czech Republic, September 27, 2013,*

Revised Selected Papers, volume 8399 of *Lecture Notes in Computer Science*, pages 148–161. Springer, 2013.

[181] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc., 2013.

[182] Newton Spolaôr and Grigorios Tsoumakas. Evaluating feature selection methods for multi-label text classication. In Axel-Cyrille Ngonga Ngomo and George Paliouras, editors, *Proceedings of the first Workshop on Bio-Medical Semantic Indexing and Question Answering, a Post-Conference Workshop of Conference and Labs of the Evaluation Forum 2013 (CLEF 2013) , Valencia, Spain, September 27th, 2013.*, volume 1094 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[183] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. *SIGMOD Rec.*, 25(2):1–12, 1996.

[184] Ashok Srivastava and Brett Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *Proceedings of the 2005 IEEE Aerospace Conference*, 2005.

[185] Fabian M. Suchanek, David Gross-Amblard, and Serge Abiteboul. Watermarking for ontologies. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2011*, pages 697–713, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

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

[187] Aixin Sun, Ee-Peng Lim, Wee Keong Ng, and Jaideep Srivastava. Blocking reduction strategies in hierarchical text classification. *IEEE Trans. Knowl. Data Eng.*, 16(10):1305–1308, 2004.

[188] Yannis Theodoridis and Mario A. Nascimento. Generating spatiotemporal datasets on the www. *SIGMOD Rec.*, 29(3):39–43, September 2000.

[189] Steffen Thoma, Achim Rettinger, and Fabian Both. Towards holistic concept representations: Embedding relational knowledge, visual attributes, and distributional word semantics. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, pages 694–710, 2017.

[190] Gerald Töpper, Magnus Knuth, and Harald Sack. DBpedia Ontology Enrichment for Inconsistency Detection. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 33–40, New York, 2012. ACM. `http://dx.doi.org/10.1145/2362499.2362505`.

[191] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality*. ACL – Association for Computational Linguistics, July 2015.

[192] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P. Vlahavas. Multi-label classification of music into emotions. In Juan Pablo Bello, Elaine Chew, and Douglas Turnbull, editors, *ISMIR*, pages 325–330, 2008.

[193] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357, 2016.

[194] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

[195] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 2008.

[196] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 99(1), 2010.

[197] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[198] Douglas Turnbull, Luke Barrington, David A. Torres, and Gert R. G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, 2008.

[199] Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 737–744. MIT Press, 2003.

[200] Marieke van Erp, Pablo Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Jörg Waitelonis. Evaluating entity linking: An analysis

of current benchmark datasets and a roadmap for doing a better job. In *of the Language Resources and Evaluation Conference. ELRA*, 2016.

[201] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, 73(2):185–214, November 2008.

[202] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of OWL DL axioms from lexical resources. In *ESWC*, pages 670–685, 2007.

[203] Johanna Völker and Mathias Niepert. Statistical schema induction. In *Proc. of the Extended Semantic Web Conference (ESWC)*, pages 124–138, 2011.

[204] Johanna Völker and Mathias Niepert. Statistical schema induction. In *Proceedings of the 8th Extended Semantic Web Conference on The Semantic Web: Research and Applications - Volume Part I*, ESWC'11, pages 124–138, Berlin, Heidelberg, 2011. Springer-Verlag.

[205] Denny Vrandečić and Markus Krötzsch. Wikidata: a Free Collaborative Knowledge Base. *Communications of the ACM*, 57(10):78–85, 2014.

[206] Chengyu Wang, Rong Zhang, Xiaofeng He, and Aoying Zhou. Error link detection and correction in wikipedia. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 307–316, New York, NY, USA, 2016. ACM.

[207] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1–1, 2017.

[208] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. AAAI Press, 2014.

[209] Gabriel Weaver, Barbara Strickland, and Gregory Crane. Quantifying the accuracy of relational statements in wikipedia: a methodology. *2006 IEEE/ACM 6th Joint Conference on Digital Libraries*, 00:358, 2006.

[210] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526, New York, NY, USA, 2014. ACM.

[211] Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In *European Semantic Web Conference*, pages 504–518. Springer, 2014.

[212] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, December 1945.

[213] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text: A look back and into the future. *ACM Comput. Surv.*, 44(4):20, 2012.

[214] Fei Wu, Raphael Hoffmann, and Daniel S. Weld. Information extraction from wikipedia: Moving down the long tail. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 731–739, New York, NY, USA, 2008. ACM.

[215] Feihong Wu, Jun Zhang, and Vasant Honavar. *Learning Classifiers Using Hierarchically Structured Class Taxonomies*, pages 313–320. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[216] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. Transg : A generative mixture model for knowledge graph embedding. *CoRR*, abs/1509.05488, 2015.

[217] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Learning multi-relational semantics using neural-embedding models. *CoRR*, abs/1411.4072, 2014.

[218] Zhenglu Yang, Jianjun Yu, and Masaru Kitsuregawa. Fast algorithms for top-k approximate string matching. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 1467–1473. AAAI Press, 2010.

[219] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370, 2012.

[220] Amrapali Zaveri, Dimitris Kontokostas, Mohamed A Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 97–104. ACM, 2013.

[221] Min-Ling Zhang, José M. Peña, and Victor Robles. Feature selection for multi-label naive bayes classification. *Inf. Sci.*, 179(19):3218–3229, September 2009.

[222] Min-Ling Zhang and Lei Wu. Lift: Multi-label learning with label-specific features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(1):107–120, 2015.

[223] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Trans. on Knowl. and Data Eng.*, 18(10):1338–1351, October 2006.

[224] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, 2007.

[225] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 26(8):1819–1837, 2014.

[226] M.L. Zhang and Z.H. Zhou. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18:1338–1351, 2006.

[227] Zexuan Zhu, Yew-Soon Ong, and Manoranjan Dash. Wrapper-filter feature selection algorithm using a memetic framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):70–76, 2007.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Dissertation ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, August 27, 2018 Unterschrift