Exploiting Semi-Structured Information in Wikipedia for Knowledge Graph Construction

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

vorgelegt von
Nicolas Heist
aus Heidelberg

Mannheim, 2024

Dekan: Prof. Dr. Claus Hertling, Universität Mannheim
Referent: Prof. Dr. Heiko Paulheim, Universität Mannheim
Korreferent: Prof. Dr. Harald Sack, Karlsruher Institut für Technologie

Tag der mündlichen Prüfung: 05.06.2024

Abstract

Knowledge graphs play an important role in today's IT landscape as they serve as a data foundation for a plethora of applications and natively support tasks like question answering or recommendation. Hence, it is more important than ever that the knowledge modelled by knowledge graphs is correct and complete. While this is an elusive goal for many domains, techniques for automated knowledge graph construction serve as a means to approach it.

In this thesis, we address three main challenges in the field of automated knowledge graph construction using semi-structured data in Wikipedia as a data source. To create an ontology with expressive and fine-grained types, we present an approach that extracts a large-scale general-purpose taxonomy from categories and list pages in Wikipedia. We enhance the taxonomy's classes with axioms explicating their semantics.

To increase the coverage of long-tail entities in knowledge graphs, we describe a pipeline of approaches that identify entity mentions in Wikipedia listings, integrate them into an existing knowledge graph, and enrich them with additional facts derived from the extraction context.

As a result of applying the above approaches to semi-structured data in Wikipedia, we present the knowledge graph CaLiGraph. The graph describes more than 13 million entities with an ontology containing almost 1.3 million classes. To judge the value of CaLiGraph for practical tasks, we introduce a framework that compares knowledge graphs based on their performance on downstream tasks. We find CaLiGraph to be a valuable addition to the field of publicly available general-purpose knowledge graphs.

Zusammenfassung

Wissensgraphen spielen eine wichtige Rolle in der heutigen IT-Landschaft, da sie als Datenbasis für eine Vielzahl von Anwendungen dienen und Aufgaben wie Fragebeantwortung oder Empfehlungen nativ unterstützen. Daher ist es wichtiger denn je, dass das von Wissensgraphen modellierte Wissen korrekt und vollständig ist. Obwohl dies für viele Bereiche ein schwer zu erreichendes Ziel ist, dienen Techniken der automatisierten Konstruktion von Wissensgraphen als Mittel, um sich diesem Ziel anzunähern.

In dieser Arbeit behandeln wir drei Hauptherausforderungen im Bereich der automatisierten Konstruktion von Wissensgraphen unter Verwendung semi-strukturierter Daten in Wikipedia als Datenquelle. Um eine Ontologie mit ausdrucksstarken und feingranularen Typen zu erstellen, präsentieren wir einen Ansatz, der eine groß angelegte, allgemeine Taxonomie aus Kategorien und Listen-Seiten in Wikipedia extrahiert. Wir erweitern die Klassen der Taxonomie um Axiome, die ihre Semantik erklären.

Um die Abdeckung von Entitäten mit geringer Häufigkeit in Wissensgraphen zu erhöhen, beschreiben wir eine Reihe von Ansätzen, die Nennungen solcher Entitäten in Auflistungen in Wikipedia identifizieren, sie in einen vorhandenen Wissensgraphen integrieren und mit zusätzlichen Fakten aus dem Extraktionskontext anreichern.

Als Ergebnis der Anwendung der oben genannten Ansätze auf semi-strukturierte Daten in Wikipedia präsentieren wir den Wissensgraphen CaLiGraph. Der Graph beschreibt mehr als 13 Millionen Entitäten mit einer Ontologie, die fast 1,3 Millionen Klassen umfasst. Um den Wert von CaLiGraph für praktische Aufgaben zu beurteilen, führen wir ein Framework ein, das Wissensgraphen anhand ihrer Leistung bei tatsächlichen Aufgaben vergleicht. Wir stellen fest, dass CaLiGraph eine wertvolle Ergänzung im Bereich der öffentlich verfügbaren, allgemeinen Wissensgraphen ist.

Acknowledgments

I want to extend my heartfelt thanks to my advisor, Professor Heiko Paulheim, for his exceptional support throughout my PhD journey. His emphasis on the human aspect of academic guidance ensured that personal well-being always came first. The lighthearted atmosphere he fostered, along with his prompt and detailed feedback, greatly enriched the research process. I particularly appreciated our engaging discussions and his unconventional ideas that significantly influenced my work.

I am very grateful to Professor Harald Sack for agreeing to be the second examiner for my dissertation. His dedication to reviewing and thereby improving my work is deeply appreciated.

A special thank you to my colleague, Sven Hertling, whose willingness to listen and assist with any problem was invaluable. His consistent support was crucial in overcoming the challenges encountered along the way.

I thank my co-authors and colleagues in the Data and Web Science Group and beyond. Your collaboration and insights were instrumental to the success of this research.

Finally, I want to express my deepest gratitude to my wonderful wife, Isabell, for her unwavering support and understanding throughout this academic journey. Her encouragement and patience were essential in making this achievement possible.

Contents

List of Publications **xv**

List of Figures **xvii**

List of Tables **xix**

List of Acronyms **xxi**

I Introduction and Foundations **1**

1 Introduction **3**
1.1 Motivation 4
1.2 Outline and Contributions 5

2 Theoretical Background **11**
2.1 The Semantic Web 11
2.1.1 Semantic Web Stack 11
2.1.2 Linked Open Data 15
2.1.3 Vocabularies 17
2.2 Knowledge Graphs 18
2.2.1 Components 19
2.2.2 Construction and Extension 20
2.2.3 Paradigms 22
2.3 Wikipedia 25
2.3.1 Editions 25
2.3.2 Wiki Markup 25
2.3.3 Content Types 26
2.3.4 Versions 27

3 Knowledge Graphs on the Web 29
3.1 Overview . . . 30
3.1.1 Manual Curation . . . 30
3.1.2 Creation from (Semi)-Structured Sources . . . 30
3.1.3 Creation from Unstructured Sources . . . 31
3.2 Comparison . . . 32
3.2.1 General Metrics . . . 32
3.2.2 Contents . . . 34
3.2.3 Looking into Details . . . 35
3.3 Linkage and Overlap . . . 38
3.3.1 Method . . . 39
3.3.2 Findings . . . 42
3.4 Conclusion and Outlook . . . 43

4 Automated Knowledge Graph Construction 47
4.1 A Pipeline for Automated Knowledge Graph Construction . . . 47
4.2 Construction of General-Purpose Knowledge Graphs . . . 49
4.2.1 DBpedia . . . 49
4.2.2 YAGO . . . 51
4.2.3 NELL . . . 51
4.2.4 BabelNet . . . 51
4.2.5 DBkWik . . . 52
4.3 Limitations and Challenges . . . 52

II Ontology Construction 55

5 Deriving a Fine-Grained Ontology from Categories and Lists 57
5.1 Motivation . . . 58
5.2 Related Work . . . 59
5.3 Categories and List Pages in Wikipedia . . . 59
5.4 Distantly Supervised Entity Extraction from List Pages . . . 62
5.4.1 Training Data Generation . . . 63
5.4.2 Entity Classification . . . 67
5.5 Results and Discussion . . . 69
5.5.1 List Page Extraction . . . 69
5.5.2 Evaluation . . . 70
5.6 Conclusion . . . 72

6 Learning Defining Axioms for Wikipedia Categories 73
6.1 Motivation . . . 74
6.2 Related Work . . . 75
6.3 Preliminaries . . . 76
6.4 Approach . . . 79

6.4.1 Candidate Selection 79
6.4.2 Pattern Mining 80
6.4.3 Pattern Application 82
6.4.4 Axiom Application and Post-Filtering 83
6.5 Experiments 84
6.5.1 Axiom Extraction using DBpedia 85
6.5.2 Comparison with Related Approaches 86
6.6 Conclusion 88

III Knowledge Graph Population 91

7 Subject Entity Detection in Wikipedia Listings 93
7.1 Motivation 94
7.2 Related Work 95
7.2.1 Named Entity Recognition 95
7.2.2 Subject Entity Detection 96
7.3 Preliminaries 97
7.3.1 Listings in Wikipedia 97
7.3.2 Training Data Generation for List Pages 97
7.3.3 Transformers for Token Classification 99
7.4 Subject Entity Detection with Transformers 99
7.4.1 Token-level Subject Entity Detection 100
7.4.2 Coarse-grained Entity Type Prediction 101
7.4.3 Negative Sampling through Shuffled Listings 101
7.4.4 Fine-Tuning on Noisy Page Labels 102
7.5 Experiments 102
7.5.1 Metrics 103
7.5.2 Datasets 103
7.5.3 Evaluation on Wikipedia List Pages 103
7.5.4 Evaluation on Wikipedia Page Listings 105
7.5.5 Ablation Study 105
7.5.6 Subject Entity Extraction over Wikipedia 106
7.6 Conclusion 107

8 NASTyLinker: NIL-Aware Entity Linker 109
8.1 Motivation 110
8.2 Related Work 111
8.3 Task Formulation 113
8.4 NASTyLinker: NIL-Aware and Scalable EL 113
8.4.1 Entity Linking Model 114
8.4.2 Cluster Initialization 115
8.4.3 Cluster Conflict Resolution 115
8.5 Experiments 116

8.5.1 Datasets 117
8.5.2 Metrics 118
8.5.3 Evaluated Approaches 119
8.5.4 Entity Linking Performance 120
8.5.5 Linking Entities in Wikipedia Listings 122
8.6 Conclusion 124

9 Information Extraction from Co-Occurring Similar Entities 125
9.1 Motivation 126
9.2 Related Work 127
9.2.1 Knowledge Graph Completion from Listings 128
9.2.2 Exploiting the Context of Listings 128
9.2.3 Rule-based Knowledge Graph Completion 129
9.3 Information Extraction From Co-Occurrences 130
9.3.1 Task Formulation 130
9.3.2 Learning Descriptive Rules for Listings 131
9.3.3 Quality Metrics for Rules 133
9.4 Exploiting Co-Occurrences in Wikipedia 134
9.4.1 Approach Overview 135
9.4.2 Data Corpus 135
9.4.3 Subject Entity Discovery 135
9.4.4 Descriptive Rule Mining 137
9.4.5 Assertion Generation and Filtering 140
9.5 Evaluation 142
9.5.1 Evaluation Procedure 142
9.5.2 Type and Relation Extraction 143
9.5.3 Novel Entities 144
9.5.4 Error Analysis 144
9.6 Conclusion 144

IV Knowledge Graph Evaluation and Usage 147

10 KGrEaT: Evaluating Knowledge Graphs via Downstream Tasks 149
10.1 Motivation 150
10.2 Framework 150
10.2.1 Purpose and Limitations 150
10.2.2 Design 151
10.2.3 Preprocessing Stage 152
10.2.4 Mapping Stage 152
10.2.5 Task Stage 152
10.3 Experiments 153
10.3.1 Experimental Setup 153
10.3.2 Results and Discussion 155

10.4 Conclusion 157

11 CaLiGraph: Statistics, Evaluation and Usage 159
11.1 Description 160
11.1.1 Purpose and Coverage 160
11.1.2 Vocabulary 160
11.2 Extraction Procedure 161
11.2.1 Data Sources 161
11.2.2 Provenance 161
11.2.3 Stability 162
11.2.4 Sustainability 162
11.3 Usage 162
11.3.1 Access 162
11.3.2 Use Cases 163
11.4 Statistics 163
11.4.1 General Metrics 163
11.4.2 Contents 165
11.5 Data Quality 167
11.5.1 Metadata 168
11.5.2 Five Star Rating 168
11.5.3 Class and Instance Data 168
11.6 Evaluation via Downstream Tasks 170
11.6.1 Experimental Setup 170
11.6.2 Results and Discussion 170
11.7 Conclusion 172

V Conclusion and Outlook 175

12 Summary 177

13 Limitations and Future Work 179
13.1 Limitations 179
13.2 Future Work 180

Bibliography 183

A Data Sources 209
A.1 Data Sources for Knowledge Graph Comparison 209
A.1.1 DBpedia 209
A.1.2 YAGO 209
A.1.3 Wikidata 210
A.1.4 BabelNet 210
A.1.5 NELL 210
A.1.6 OpenCyc 210

A.1.7 VoldemortKG 210
A.2 Data Sources for KGrEaT 210
A.2.1 DBpedia 211
A.2.2 Wikidata 211
A.2.3 DBkWik 211
A.2.4 CaLiGraph 211

B Experimental Details for KGrEaT 213
B.1 Results for General-Purpose Knowledge Graphs 213
B.2 Results for CaLiGraph 213

List of Publications

Parts of the work presented in this thesis were previously published in international journals and proceedings of international conferences. We reference the corresponding publications in the respective chapters and list them here in inverse chronological order. Equal contributions are indicated with a †.

Nicolas Heist and Heiko Paulheim. CaLiGraph: A Knowledge Graph from Wikipedia Categories and Lists. In *Semantic Web Journal (SWJ)*, 2024. [under review]

Nicolas Heist,† Sven Hertling† and Heiko Paulheim. KGrEaT: A Framework to Evaluate Knowledge Graphs via Downstream Tasks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (CIKM), pp. 3938-3942, Birmingham, UK, October 2023, ACM Press.

Nicolas Heist and Heiko Paulheim. NASTyLinker: NIL-Aware Scalable Transformer-based Entity Linker. In *The Semantic Web - ESWC 2023. Lecture Notes in Computer Science*, vol. 13870, pp. 174-191, Hersonissos, Greece, May 2023, Springer, Cham.

Nicolas Heist and Heiko Paulheim. Transformer-based Subject Entity Detection in Wikipedia Listings. In *Proceedings of the 6th International Workshop on Deep Learning for Knowledge Graphs* (DL4KG @ ISWC'22), vol. 3342, October 2022, CEUR Workshop Proceedings.

Nicolas Heist and Heiko Paulheim. The CaLiGraph Ontology as a Challenge for OWL Reasoners. In *Proceedings of the 1st Semantic Reasoning Evaluation Challenge* (SemREC @ ISWC'21), vol. 3123, Virtual Event, October 2021, CEUR Workshop Proceedings. *Challenge Winner of Task 1.*

Nicolas Heist and Heiko Paulheim. Information Extraction From Co-occurring Similar Entities. In *Proceedings of the Web Conference 2021* (WWW'21), pp. 3999-4009, Ljubljana, Slovenia, April 2021, ACM Press.

Nicolas Heist, Sven Hertling, Daniel Ringler and Heiko Paulheim. Knowledge Graphs on the Web – an Overview. In *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, vol. 47, pp. 3-22, 2020, IOS Press.

Nicolas Heist and Heiko Paulheim. Entity Extraction from Wikipedia List Pages. In *The Semantic Web - ESWC 2020. Lecture Notes in Computer Science*, vol. 12123, pp. 327-342, Virtual Event, May 2020, Springer, Cham.

Nicolas Heist and Heiko Paulheim. Uncovering the Semantics of Wikipedia Categories. In *The Semantic Web – ISWC 2019. Lecture Notes in Computer Science*, vol. 11778, pp. 219–236, Auckland, New Zealand, October 2019, Springer, Cham.

Nicolas Heist. Towards Knowledge Graph Construction from Entity Co-occurrence. In *Proceedings of the EKAW Doctoral Consortium 2018*, vol. 2306, Nancy, France, November 2018, CEUR Workshop Proceedings.

In addition to the publications listed before, which are directly related to the content of this thesis, the author contributed to and published other research work during his doctoral studies:

Jan Portisch, **Nicolas Heist** and Heiko Paulheim. Knowledge Graph Embedding for Data Mining vs. Knowledge Graph Embedding for Link Prediction–Two Sides of the Same Coin? In *Semantic Web Journal* (SWJ), vol. 13, no. 3, pp. 399-422, 2022, IOS Press.

Russa Biswas, Radina Sofronova, Mehwish Alam, **Nicolas Heist**, Heiko Paulheim and Harald Sack. Do judge an entity by its name! Entity typing using language models. In *The Semantic Web: ESWC 2021 Satellite Events, Lecture Notes in Computer Science*, vol. 12739, pp. 65-70, Virtual Event, May 2021, Springer, Cham.

Niklas Lüdemann, Ageda Shiba, Nikolaos Thymianis, **Nicolas Heist**, Christopher Ludwig and Heiko Paulheim. A Knowledge Graph for Assessing Aggressive Tax Planning Strategies. In *The Semantic Web – ISWC 2020. Lecture Notes in Computer Science*, vol. 12507, pp. 395-410, Virtual Event, November 2020, Springer, Cham.

Florian Schrage, **Nicolas Heist** and Heiko Paulheim. Extracting Literal Assertions for DBpedia from Wikipedia Abstracts. In *Semantic Systems. The Power of AI and Knowledge Graphs. Lecture Notes in Computer Science*, vol. 11702, pp. 288-294, November 2019, Springer, Cham.

List of Figures

1.1 Overview of the thesis' main content and chapter assignment. 6

2.1 The *Semantic Web Language Stack* according to Berners-Lee . 12
2.2 Example graph taken from Fig. 1.1 on page 6. 14
2.3 Linked Open Data Cloud from September 2023. 16
2.4 Ontology learning layer cake (adapted from [32]). 20
2.5 Excerpt of *Gilby Clarke* page taken from Fig. 1.1 on page 6. . . 22

3.1 Depiction of the size and linkage degree of open KGs 34
3.2 Instances in DBpedia 35
3.3 Instances in YAGO 36
3.4 Instances in Wikidata 38
3.5 Instances in BabelNet 39
3.6 Instances in OpenCyc 40
3.7 Instances in NELL 41
3.8 Instances in Voldemort 42
3.9 Potential gain of combining KGs 44
3.10 Existing entities in two KGs in relation to the number of links. 45

4.1 A pipeline for AKGC (taken from Fig. 1.1 on page 6). 48
4.2 A timeline with major milestones of popular public KGs. . . . 50

5.1 Wikipedia list page in enumeration layout 58
5.2 Wikipedia list page in table layout 61
5.3 Pipeline of the approach 63
5.4 Invalid nodes and edges in category and list graph 64
5.5 Examples of non-taxonomic nodes and edges 64
5.6 Possible connections between category and list graph 65
5.7 Extension of the category-list taxonomy with DBpedia mappings. 66

5.8 Distribution of entities new to DBpedia 70
5.9 The 15 most important features used by XG-Boost 70
5.10 Comparison of existing and additional statements for DBpedia 71

6.1 Excerpt of the WCG showing a category with subcategories . . 74
6.2 Overview of the Cat2Ax approach. 78
6.3 Pattern application performance for varying confidence intervals 86
6.4 Comparison of the extracted results. 88

7.1 Wikipedia page of Gilby Clarke 94
7.2 Examples of Wikipedia page listings 98

8.1 Listings in Wikipedia containing the mention *James Lake* . . . 110
8.2 Main phases of the NASTyLinker approach 114
8.3 Runtime of NASTyLinker components 121

9.1 Wikipedia page of Gilby Clarke 126
9.2 An overview of the approach with exemplary outputs 136
9.3 *tagfit* of assertions generated from rules 140

10.1 An overview of the KGrEaT framework. 151

11.1 A sunburst diagram of frequent entity types in CaLiGraph. . . 165
11.2 A sunburst diagram of frequent properties in CaLiGraph . . . 167

List of Tables

3.1 General metrics of open KGs. 33
3.2 Detail statistics for selected classes in open KGs. 37

4.1 Advantages and Limitations of public general-purpose KGs. . . 53

5.1 Features of the ML model grouped by list page and feature type 68
5.2 Performance measures for various classification models 69

6.1 Examples of discovered textual patterns and implications . . . 85
6.2 Total number of axioms/assertions and precision scores 87

7.1 Statistics of the datasets used for the experiments 104
7.2 Evaluation results for SE detection on Wikipedia list pages . . 104
7.3 Performance metrics on page listings for the best model . . . 105
7.4 Evaluation results for SE detection on Wikipedia page listings 106
7.5 Number of extracted mentions of SEs for Wikipedia listings . 107

8.1 Mention and entity occurrences in the partitions of the datasets117
8.2 Results for the test partition $\mathcal{D}^N_{test}$ of the NILK dataset. 120
8.3 Results for the test partition $\mathcal{D}^L_{test}$ of the LISTING dataset . . . 123
8.4 Results of the manual evaluation of 100 clusters and mentions 124

9.1 Exemplary vectors for a set of listings 132
9.2 Number of generated assertions before and after filtering . . . 142
9.3 Correctness of manually evaluated assertions. 142
9.4 Error types partitioned by cause 145

10.1 Tasks implemented in KGrEaT 154
10.2 Evaluation results of the KGs with average rank per task type 156

11.1 Basic metrics of all CaLiGraph versions and other KGs 164
11.2 Comparison of counts and ranks among CaLiGraph versions . 166
11.3 Collection of evaluation results of CaLiGraph data. 169
11.4 Evaluation results of the KGs with average rank per task type 171
11.5 Dataset coverage of the KGs evaluated with KGrEaT 173

B.1 Dataset coverage of the KGs evaluated with KGrEaT 214
B.2 KGrEaT evaluation results for the *PK* scenario 215
B.3 KGrEaT evaluation results for the *PA* scenario 216
B.4 KGrEaT evaluation results for the *RA* scenario 217
B.5 KGrEaT evaluation results for the precision-oriented scenario 218
B.6 KGrEaT evaluation results for the recall-oriented scenario . . 219

List of Acronyms

AI Artificial Intelligence. 3, 20, 33–35, 43

AKGC Automated Knowledge Graph Construction. 4, 7, 47, 177, 179, 180, 182

ARI Adjusted Rand Index. 118

BERT Bidirectional Encoder Representations from Transformers. 24, 99, 182

CDC Cross-Document Coreference Resolution. 112

CLI Command Line Interface. 152

CSV Comma-Separated Values. 16

CWA Closed-World Assumption. 23

DistilBERT Distilled version of BERT. 24, 99

EL Entity Linking. 21, 22, 24, 94, 95, 109–114, 117, 119, 120, 122, 124

FOAF Friend Of A Friend. 17, 160, 168

HTTP HyperText Transfer Protocol. 15

IE Information Extraction. 4, 21, 22, 25, 59, 125, 144, 181, 182

ILP Inductive Logic Programming. 129

KB Knowledge Base. 19

KG Knowledge Graph. 3–5, 7–9, 18–24, 29–35, 38–40, 42, 43, 46–49, 51, 52, 54, 58, 60, 73–76, 79, 88, 89, 94–96, 106, 107, 109, 110, 112, 113, 115–117, 119, 122–130, 132, 135, 143–145, 149–153, 155, 157, 159–161, 163, 168, 170–172, 177–182, 209, 210, 213

KGC Knowledge Graph Construction. 20, 22, 30

KGE Knowledge Graph Extension. 20, 94, 105, 127

KGP Knowledge Graph Population. 47, 49, 51, 52, 109, 111, 113, 116, 179, 180

LCWA Local Closed-World Assumption. 23, 118, 122, 131, 143

LLM Large Language Model. 182

LM Language Model. 24, 182

LOD Linked Open Data. 16, 17, 33, 168

ML Machine Learning. 23, 24, 47, 57, 72, 95

NED Named Entity Disambiguation. 21, 49

NEL Named Entity Linking. 21

NELL Never-Ending Language Learner. 31, 33, 34, 36, 43, 47, 51, 54

NER Named Entity Recognition. 21, 49, 95, 96, 99, 101, 103

NLP Natural Language Processing. 20, 24, 59, 99, 182

NMI Normalized Mutual Information. 118, 121

OC Ontology Construction. 4, 20, 47–49, 51, 52, 179, 180

OIE Open Information Extraction. 4

OWA Open-World Assumption. 23, 39, 76, 101, 126

OWL Web Ontology Language. 12, 13, 15, 18, 160, 161

PROV-O PROV Ontology. 18, 160, 161, 168

RDF Resource Description Framework. 12–17, 19, 152, 160, 178

RDFS Resource Description Framework Schema. 12–14

RIF Rules Interchange Format. 13

RML RDF Mapping Language. 47

RoBERTa Robustly Optimized BERT Pre-Training Approach. 24

SE Subject Entity. 58–62, 69, 93–97, 99–103, 105–107, 117, 125–128, 130, 131, 134, 135, 137–139, 141, 142, 144, 145, 181

SHACL Shapes Constraint Language. 13, 51

SKOS Simple Knowledge Organisation System. 17, 18, 160, 168

SPARQL SPARQL Protocol And RDF Query Language. 12, 16, 168

SQL Structured Query Language. 3

SW Semantic Web. 11–13, 15, 17, 23

URI Uniform Resource Identifier. 12, 13, 15, 16, 152, 161, 162

W3C World Wide Web Consortium. 11, 13–16

WCG Wikipedia Category Graph. 27, 51, 57, 59, 60, 62, 72, 73, 76, 79, 163

XML eXtensible Markup Language. 12, 161

YAGO Yet Another Great Ontology. 30, 31, 36, 39, 43, 46, 47, 51, 52, 54, 59, 74, 75, 153, 155, 157, 163, 165, 170, 171, 181

Part I

Introduction and Foundations

CHAPTER 1

Introduction

An essential property of a truly intelligent application is its ability to access all the information necessary to solve the task it is designed for. With the advent of Knowledge Graphs (KGs), this long-standing objective in the Artificial Intelligence (AI) field of supplying machines with relevant information is gradually becoming a reality [107, 201]. KGs are the key technology to tie together data and knowledge [59]. Thereby, they diminish the effort of combining data with other sources [156] or using it in applications of various domains (e.g., agriculture [29], manufacturing [25], or tourism [95]) and task types (e.g., advertising [62], question answering [85] or recommendation [197]).

The core idea of KGs is to represent data as a labelled directed graph, with nodes representing concepts or concrete instances and edges representing their relations. Using graphs to represent data has several advantages over relational or NoSQL alternatives, like the flexible definition and reuse of schemas and the large variety of graph-based techniques for querying, search or analytics [83].

As shown in Fig. 1.1, nodes in a KG may represent concepts (e.g., the type *Album* or the relation *artist*) or entities (e.g., the album *California Girl* or the artist *Nancy Sinatra*). Relations may exist between concepts (*Guns N' Roses album* is a subclass of *Album*), between a concept and an entity (*California Girl* is an *Album*) or between entities (*California Girl* has the artist *Nancy Sinatra*). All this information is stored in the form of (*subject*,*predicate*,*object*) triples.

1.1 Motivation

The trend of entities added to publicly available KGs in recent years indicates they are far from complete. The number of entities in Wikidata [195], for example, grew by 26% in the time from October 2020 (85M) to October 2023 (107M) [206]. Wikidata describes the largest number of entities and comprises – in terms of entities – other public KGs to a large extent [66]. Consequently, this challenge of incompleteness applies to all public KGs, particularly when it comes to less popular entities [44].

The field of Automated Knowledge Graph Construction (AKGC) bundles all efforts to create KGs with only minimal human involvement. This includes approaches for automated Ontology Construction (OC) and Information Extraction (IE). The latter can help mitigate the above problem if the approaches ensure that the extracted information is of high quality. While the performance of Open Information Extraction (OIE) systems (i.e., systems that extract information from general web text) has improved in recent years [159, 97, 112], the quality of extracted information has not yet reached a level where integration into public KGs like Wikidata or DBpedia [104] should be done without further filtering.

The extraction of information from semi-structured data is generally less error-prone and has already proven to yield high-quality results. DBpedia is an example of an influential open KG extracted primarily from Wikipedia infoboxes; further approaches use the category system of Wikipedia [120, 211] or focus on tables (in Wikipedia or the web) as a semi-structured data source to extract entities and relations [215]. As highlighted by Weikum [201], first "picking low-hanging fruit" by focusing on premium sources like Wikipedia to build a high-quality KG is crucial as it can serve as a solid foundation for approaches that target more challenging data sources. The extracted information may then be used as an additional anchor to make sense of less structured data.

In our review of automatically constructed, open KGs in Chapter 4, we identify several challenges with high potential impact in AKGC. In this thesis, we describe our efforts to address the following three challenges using semi-structured data in Wikipedia as a data source:

(C1) creating an ontology with expressive, fine-grained types,
(C2) increasing coverage of long-tail entities, while
(C3) maintaining high data quality.

In particular, the data structures in Wikipedia that we use to extract additional information are categories (example in the top part of Fig. 1.1) and listings like enumerations and tables (example in the middle part of Fig. 1.1).

(C1) Ontology with expressive, fine-grained types The ontology of a KG contains the concepts that are described in the graph. Categories in Wikipedia can serve as a large hierarchy of concepts with existing links to articles in Wikipedia. The links can be used to assign articles to the concept hierarchy. Vice versa, they may also be used to derive knowledge about the concepts. For example, based on the articles in the category *Guns N' Roses albums*,[1] we can derive that the category contains only articles with *Guns N' Roses* as an artist.

Apart from categories, Wikipedia contains *list pages*, which may serve as an additional source for concepts as they contain listings of related articles. For example, the *List of greatest hits albums*[2] contains compilation albums from various bands. By connecting list pages to categories, the concept hierarchy can be extended. Derived from the list page, *Greatest hits album* could become another subclass of *Album* in the hierarchy.

(C2) Coverage of long-tail entities For long-tail entities, little information is available, and their mentions in data sources are scarce. Listings (in list pages or articles) often contain sets of related entities, which can be identified easily through the semi-structured nature of listings. In the example in Fig. 1.1, entities are always mentioned at the beginning of a listing item. In some cases, the context of a listing provides additional information about the mentioned entities, for example, the information that the album *California Girl* has the artist *Nancy Sinatra*. This is especially useful when extracting long-tail entities, as information about them is scarce.

(C3) Maintain high data quality When creating a KG or refining an existing KG with additional information, it is essential to ensure that data quality criteria like a certain level of correctness or consistency are maintained. We use semi-structured data in Wikipedia as a data source to be able to meet such criteria. Nevertheless, we perform intrinsic and extrinsic evaluations in this thesis to judge the quality of the extraction results.

1.2 Outline and Contributions

This thesis is structured into five parts. The remainder of **Part I** introduces the necessary theoretical background and reviews open KGs. **Chapter 2** gives an introduction to key concepts of the *Semantic Web*, *Knowledge Graphs*, and *Wikipedia*. **Chapter 3** provides an overview and a comparison of publicly available KGs and gives insights into their contents, size, coverage, and

[1] `https://en.wikipedia.org/wiki/Category:Guns_N'_Roses_albums`
[2] `https://en.wikipedia.org/wiki/List_of_greatest_hits_albums`

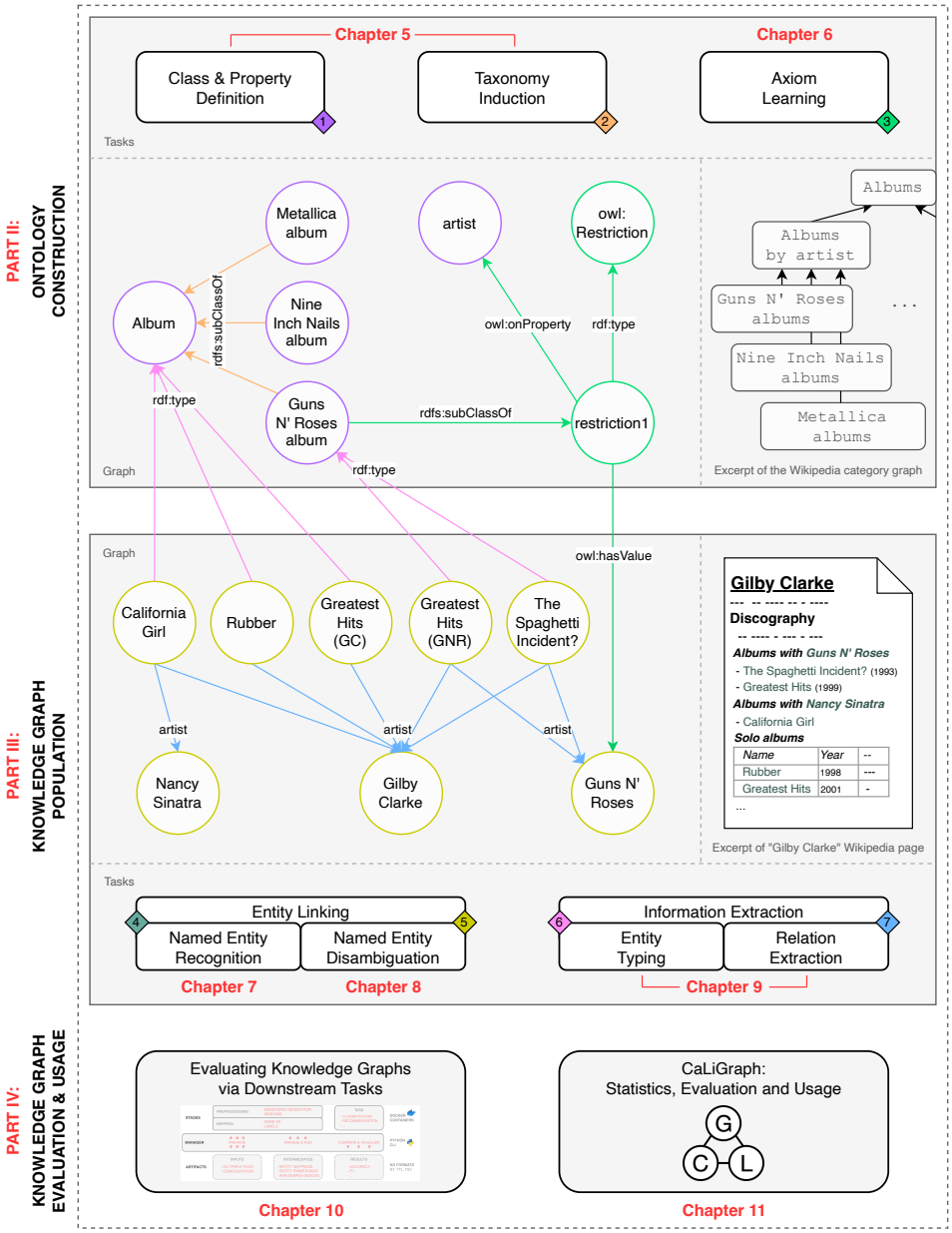


Figure 1.1: Overview of the thesis' main content and chapter assignment.

overlap. In **Chapter 4**, we define a pipeline for AKGC, discuss the construction procedures of automatically generated public KGs, and identify open challenges.

The main findings of this thesis are presented in Parts II to IV, as visualised in Fig. 1.1. The approaches and findings presented in these three parts are gradually integrated into CaLiGraph.[3]

Part II: Ontology Construction This part addresses challenge (C1) as it describes how to compile Wikipedia categories and list pages into a large-scale taxonomy that is enriched with descriptive semantic axioms.

Chapter 5: Deriving a Fine-Grained Ontology from Wikipedia Categories and Lists This chapter presents the approach for combining Wikipedia categories with list pages to form a large-scale taxonomy. The taxonomy has the advantage of being deeply interlinked with categories, list pages and the DBpedia ontology. These links bear the potential to enrich the taxonomy with additional information derived from these sources at a later stage. To demonstrate the potential of this taxonomy, we show in this chapter how it can be used to derive a large number of new entities and assertions from Wikipedia list pages.

The key contributions of this chapter are (1) an approach for generating a taxonomy from categories and list pages and (2) an approach for extracting entities from list pages using distant supervision.

Chapter 6: Learning Defining Axioms for Wikipedia Categories This chapter introduces the *Cat2Ax* approach to enrich Wikipedia-based KGs by explicating the semantics in category names. The approach combines the category graph structure, lexical patterns in category names, and instance information from a KG to learn patterns in category names and map these patterns to type and relation axioms.

The key contributions of this chapter are (1) an approach that extracts axioms for Wikipedia categories using features derived from the instances in a category and their lexicalisations and (2) more than 700K axioms explicating the semantics of Wikipedia categories.

Part III: Knowledge Graph Population This part addresses challenge (C2) as it describes how to exploit semi-structured data in Wikipedia to identify novel entities and derive assertions from page structure and entity co-occurrence.

[3] A KG created from Wikipedia **Ca**tegories and **Li**sts: `http://caligraph.org`

Chapter 7: Subject Entity Detection in Wikipedia Listings This chapter describes an approach for detecting the subject entities of Wikipedia listings. The approach uses a Transformer network to detect entities without requiring mention boundaries.

The key contributions of this chapter are (1) a generic subject entity detection approach applicable to arbitrary listings in Wikipedia and (2) a set of almost 20M mentions of unknown entities extracted from Wikipedia listings.

Chapter 8: NASTyLinker: NIL-Aware Scalable Transformer-based Entity Linker This chapter addresses the task of disambiguating entity mentions by mapping them to existing entities in the KG or creating new ones in case the entity is missing. Our proposed approach, *NASTyLinker*, can handle missing entities and produces corresponding mention clusters while maintaining high linking performance for known entities. We apply NASTyLinker to the subject entity mentions extracted in the previous chapter to create a set of properly disambiguated entities.

The key contributions of this chapter are (1) the NASTyLinker approach for effectively disambiguating entity mentions and (2) the disambiguated set of 7.6 million new entities extracted from Wikipedia listings.

Chapter 9: Information Extraction from Co-Occurring Similar Entities In this chapter, we explore how information extracted from similar entities that co-occur in structures like tables or enumerations can help to increase the coverage of KGs. We propose a descriptive rule-mining approach that uses distant supervision to derive rules for these relationships based on a listing's context.

The key contributions of this chapter are (1) an approach for learning descriptive rules for listing characteristics based on the listing context and (2) a set of 30.4M assertions extracted for subject entities in Wikipedia listings with an overall correctness of over 90%.

Part IV: Knowledge Graph Evaluation and Usage In this part, we present a framework for the extrinsic evaluation of KGs, comprehensively describe and evaluate CaLiGraph, and so address the data quality challenge (C3).

Chapter 10: KGrEaT: Evaluating Knowledge Graphs via Downstream Tasks This chapter presents KGrEaT – a framework to estimate the quality of KGs through evaluation on downstream tasks like classification, clustering, or recommendation. The evaluation results (e.g., the accuracy of a classification model trained with the KG as background knowledge) can serve as extrinsic task-based quality metrics for the KG.

The key contributions of this chapter are (1) KGrEaT, a framework to judge the utility of KGs using extrinsic task-based metrics, and (2) a comparison of several well-known cross-domain KGs w.r.t. these metrics.

Chapter 11: CaLiGraph: Statistics, Evaluation and Usage This chapter gives an overview of CaLiGraph as a data source. We describe its versions, purpose, and vocabulary structure. We detail the extraction procedure of CaLiGraph, including sources, provenance, stability, and sustainability. We explain how CaLiGraph can be accessed and how it is used already. We present statistics of the KG, summarize evaluation results of the included approaches, and apply KGrEaT to compare CaLiGraph to other general-purpose KGs.

The key contributions of this chapter are (1) a comprehensive overview of CaLiGraph as a data source, including general details, usage information, and statistics, and (2) a comparison of the latest version of CaLiGraph with existing general-purpose KGs.

We conclude this thesis in **Part V** with a summary of the presented findings in **Chapter 12**, followed by existing limitations and potential future work in **Chapter 13**. The acronyms used throughout this thesis are summarized in the List of Acronyms starting on page *xix*.

CHAPTER 2

Theoretical Background

This chapter introduces the fundamental concepts of the Semantic Web and Knowledge Graphs. Further, it describes the key elements of Wikipedia, serving as the main data source for this thesis.

2.1 The Semantic Web

The *Semantic Web (SW)* is a field of research with the long-term goal of enhancing the World Wide Web with machine-understandable information [15]. Intelligent agents can consume this information to solve all kinds of tasks. As a means to reach this goal, methods and tools are developed that are oftentimes referred to as *Semantic Web Technologies*. More recently, these technologies have not only been used to achieve the previously mentioned goal but rather for general information management (i.e., data discovery, sharing, integration, and use [80]).

The *World Wide Web Consortium (W3C)* develops standards (or, officially, *W3C Recommendations*) to ensure seamless interaction and consistency between those technologies. In the following, we give an overview of the standards and vocabularies relevant to this thesis.

2.1.1 Semantic Web Stack

Fig. 2.1 depicts the *Semantic Web Language Stack* as proposed by Berners-Lee [14] in 2009. Since then, the W3C has updated existing recommendations (e.g., *OWL*[1]) and published additional ones (e.g., *SHACL*[2]), but the core

[1] `https://www.w3.org/TR/owl2-overview/`
[2] `https://www.w3.org/TR/shacl/`

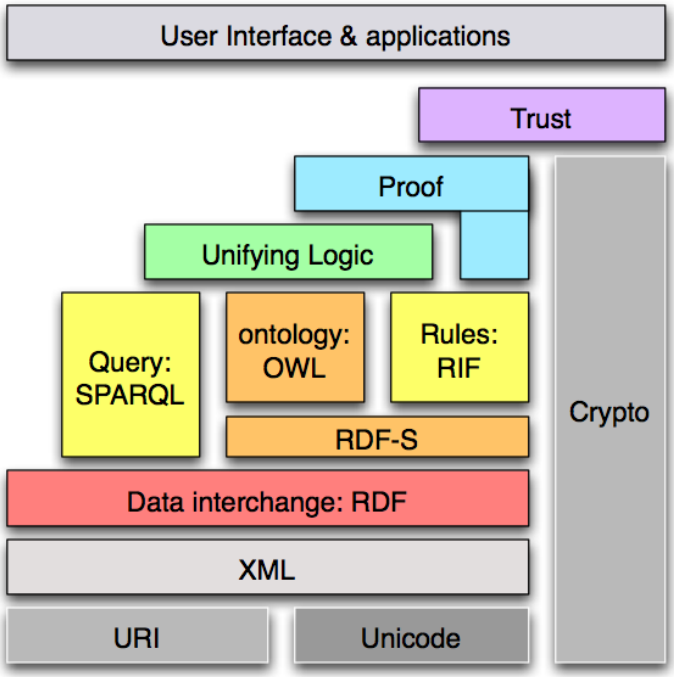


Figure 2.1: The *Semantic Web Language Stack* according to Berners-Lee [14].

concepts have persisted.

From the bottom to the top, the layers address increasingly complex topics in the SW field. *Uniform Resource Identifiers (URIs)*[3] serve as identifiers for the things to describe; encoding in *Unicode* ensures interoperability. The *eXtensible Markup Language (XML)*[4] is a serialization format to exchange information. However, there are other formats like *N-Triples*[5] or *Turtle*[6] explicitly developed to serialize information in the SW field.

The *Resource Description Framework (RDF)*[7] is a format to exchange data using simple statements. The *Resource Description Framework Schema (RDFS)*[8] enhances RDF with additional language elements to express simple ontologies; the *Web Ontology Language (OWL)*[9] defines vocabulary to describe increasingly complex ontologies. The *SPARQL Protocol And RDF Query Language (SPARQL)*[10] is a query language to retrieve and manipulate data

[3] `https://www.w3.org/Addressing/URL/4_URI_Recommentations.html`
[4] `https://www.w3.org/TR/xml/`
[5] `https://www.w3.org/TR/rdf11-n-triples/`
[6] `https://www.w3.org/TR/rdf11-turtle/`
[7] `https://www.w3.org/TR/rdf11-concepts/`
[8] `https://www.w3.org/TR/rdf11-schema/`
[9] `https://www.w3.org/TR/owl2-overview/`
[10] `https://www.w3.org/TR/sparql11-query/`

stored in RDF. Rules addressing the data described in RDF can be formulated in the *Rules Interchange Format (RIF)*[11]. The Shapes Constraint Language (SHACL)[12] is a recent addition that ensures the validity of RDF graphs by formulating so-called shapes.

Logic and *Proofs* (e.g., in the form of reasoners like HermiT [54]) can be used to extend and apply the information from ontologies and data. As anybody can publish data about arbitrary topics, *Trust* in authoritative sources is necessary to have confidence in information and the actions derived thereof [73]. Interaction with information and execution of actions is performed through applications built on SW standards.

A main contribution of this thesis concerns expressing and publishing data with RDF, RDFS, and OWL, which is why we describe them in more detail in the following.

Resource Description Framework (RDF)

RDF is recommended by the W3C in version 1.1; a draft for version 1.2 was published in December 2023. RDF can be used to express simple statements (also called *assertions*) about resources in the form of `(subject, predicate, object)` triples. The `subject` is either a resource identified by a URI or a blank node (i.e., a resource without URI); the `predicate` is a resource identified by a URI that is defined as RDF property; the `object` can be a resource identified by a URI, a blank node, or a literal (e.g., a string, number, or date). A set of RDF assertions form a directed graph with `subjects` and `objects` as nodes and `predicates` as edges.

Fig. 2.2 shows a graphical depiction of an RDF graph. We use simple labels in the nodes to describe resources for better readability, although their actual URIs may differ. To serialize statements like "*The Spaghetti Incident? has the artist Guns N' Roses*", we either use N-Triple notation (cf. Example 2.1.1) or Turtle notation (cf. Example 2.1.2) in this work. The `@prefix` notation defines namespaces (i.e., shortcuts for vocabularies), which we will define once and omit afterwards.

Example 2.1.1. A statement in N-Triple notation

<http://example.org/The_Spaghetti_Incident?>
<http://example.org/artist>
<http://example.org/Guns_N'_Roses> .

Example 2.1.2. A statement in Turtle notation

@prefix ex: <http://example.org/> .
ex:The_Spaghetti_Incident? ex:artist ex:Guns_N'_Roses .

[11] `https://www.w3.org/TR/rif-overview/`
[12] `https://www.w3.org/TR/shacl/`

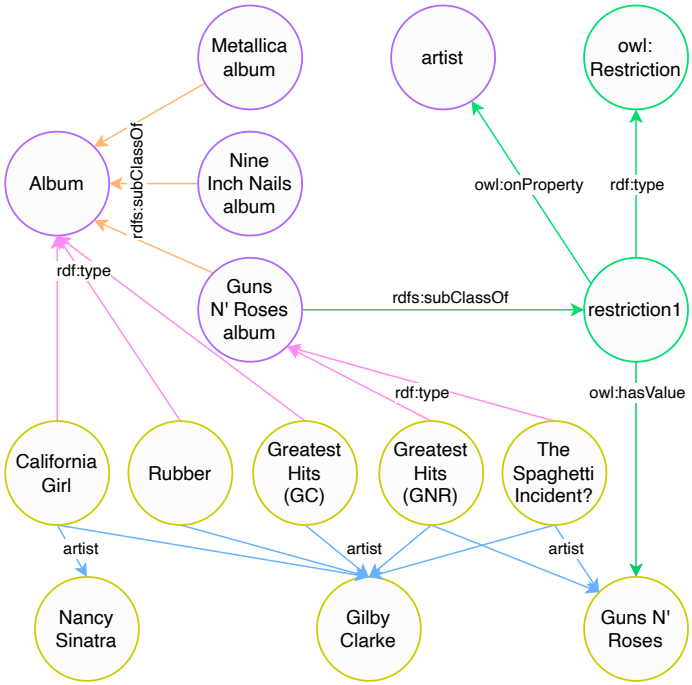


Figure 2.2: Example graph taken from Fig. 1.1 on page 6.

Resource Description Framework Schema (RDFS)

Similar to RDF, RDFS is recommended by the W3C in version 1.1, and a draft for version 1.2 was published in December 2023. While the sole purpose of RDF is to formulate information on a statement level, RDFS adds an object-oriented view by introducing classes together with properties to describe their relations. This enables us to define classes and their hierarchies, i.e., a taxonomy (cf. Example 2.1.3). Further, we can make statements about the type of resources using the taxonomy (cf. Example 2.1.4). While only the first statement in this example is given in the graph, a reasoner can deduce the second statement by exploiting the transitivity of the *rdfs:subClassOf* property. Beyond defining taxonomies, RDFS provides additional vocabulary to describe an ontology, like defining the possible subjects (the *domain*) and objects (the *range*) of a property.

Example 2.1.3. The *Album* class and its subclasses

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
ex:Album rdf:type rdfs:Class .
ex:Metallica_album rdfs:subClassOf ex:Album .
ex:Nine_Inch_Nails_album rdfs:subClassOf ex:Album .
ex:Guns_N'_Roses_album rdfs:subClassOf ex:Album .

Example 2.1.4. Types of the resource *The Spaghetti Incident?*

```
ex:The_Spaghetti_Incident? rdf:type ex:Guns_N'_Roses_album .
ex:The_Spaghetti_Incident? rdf:type ex:Album .
```

Web Ontology Language (OWL)

In the SW field, an ontology is defined as an "*explicit specification of a conceptualization*" with the latter being an "*abstract, simplified view of the world*" [57]. Hence, we use an ontology to describe a domain, including its concepts, relations, and axioms. OWL is recommended by the W3C in version 2 and can be used to express ontologies comprehensively. One can choose between two variants (*DL* and *Full*) and three profiles (*EL*, *QL*, and *RL*) restricting the language features by a certain extent for improved scalability and decidability.

OWL can, for instance, be used to describe that two classes are disjoint. If we knew that *Metallica* and *Nine Inch Nails* never recorded an album together, we could express this fact as shown in Example 2.1.5.

Example 2.1.5. Disjointness between albums of *Metallica* and *Nine Inch Nails*

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
ex:Metallica_album owl:disjointWith ex:Nine_Inch_Nails_album .
```

Another feature is the definition of restrictions for classes. For instance, with *hasValue* restrictions, one can define assertions between all instances of a class and a specific individual. In Example 2.1.6, we express that all instances of *Guns N' Roses albums* have the artist *Guns N' Roses*. The assertions for the concrete instances don't have to be contained in the RDF graph as a reasoner can deduce them.

Example 2.1.6. hasValue restriction for *Guns N' Roses album*

```
ex:restriction1 rdf:type owl:Restriction ;
        owl:onProperty ex:artist ;
        owl:hasValue ex:Guns_N'_Roses .
ex:Guns_N'_Roses_album rdfs:subClassOf ex:restriction1 .
```

2.1.2 Linked Open Data

Linked Data is structured data that is publicly accessible and interlinked with other data. Tim Berners-Lee formulated four core principles for Linked Data in 2006 [13]:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names

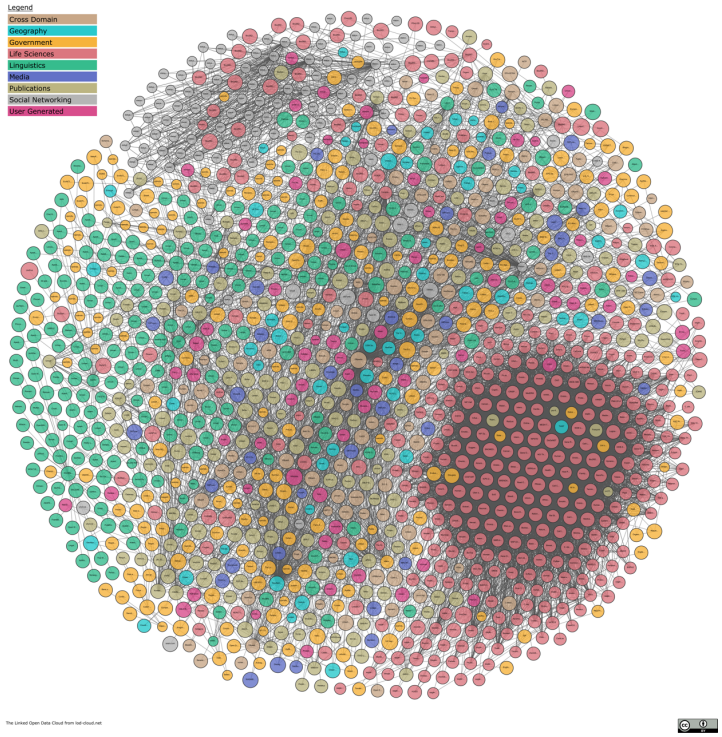


Figure 2.3: Linked Open Data Cloud from September 2023.

3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)

4. Include links to other URIs, so that they can discover more things

Further, he defined *Linked Open Data (LOD)* as "*Linked Data which is released under an open licence, which does not impede its reuse for free.*" Fig. 2.3 shows the LOD Cloud,[13] a large collection of open and connected datasets.

Five Star Linked Open Data

In 2010, Berners-Lee added a five-star rating for LOD to indicate how well a dataset adheres to the principles mentioned above [13]. Higher ratings are only awarded if all previous conditions are fulfilled.
★☆☆☆☆ Available on the web with an open license, to be Open Data
★★☆☆☆ Available as machine-readable structured data
★★★☆☆ Published in a non-proprietary format (e.g., CSV instead of Excel)
★★★★☆ Identify things using open standards from W3C (RDF and SPARQL)
★★★★★ Linked to other people's data to provide context

[13] `https://lod-cloud.net/`

Five Star Linked Data Vocabulary Use

Complementary to the Five Star rating for LOD, Janowicz et al. published another Five Star rating in 2014 targeting the use of LOD vocabularies [91]. With their guidelines, they encourage users to publish their vocabulary with rich metadata and interconnections to other vocabularies in order to provide as much value to the community as possible.
☆☆☆☆☆ Linked Data without any vocabulary
★☆☆☆☆ Vocabulary with dereferencable human-readable description
★★☆☆☆ Machine-readable explicit axiomatization of the vocabulary
★★★☆☆ The vocabulary is linked to other vocabularies
★★★★☆ Metadata about the vocabulary is available
★★★★★ The vocabulary is linked to by other vocabularies

2.1.3 Vocabularies

While some vocabularies describe topical domains (e.g., sports or cooking), others enrich a dataset with meta-information (e.g., provenance). In the following, we describe some of the established vocabularies in the SW field used in this thesis.

Friend Of A Friend (FOAF)

FOAF is "*a project devoted to linking people and information using the Web*".[14] It defines classes and properties in RDF to describe people, groups, organizations, and projects. Example 2.1.7 shows a basic description of the CaLiGraph project with FOAF.

Example 2.1.7. Description of the CaLiGraph project with FOAF

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
ex:caligraph rdf:type foaf:Project ;
    foaf:name "CaLiGraph" ;
    foaf:homepage <http://caligraph.org> .
```

Simple Knowledge Organisation System (SKOS)

SKOS is an RDF vocabulary to express concept schemes like thesauri and taxonomies.[15] As shown in Example 2.1.8, it is oftentimes used to define taxonomies in a simple way and to express a preference of which label to use for an entity.

[14] http://xmlns.com/foaf/spec/
[15] https://www.w3.org/TR/swbp-skos-core-spec/

Example 2.1.8. Album hierarchy with SKOS

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
ex:Guns_N'_Roses_album skos:broader ex:Album ;
    skos:prefLabel "Guns N' Roses album" ;
    skos:altLabel "GNR album" .
```

PROV Ontology (PROV-O)

PROV-O is an ontology defined in OWL that is used to enrich a dataset with provenance information.[16] As shown in Example 2.1.9, we use PROV-O in this thesis to declare the source of new classes or entities. We assume in this example that the class *Guns N' Roses album* has been created from the Wikipedia category *Guns N' Roses albums*.

Example 2.1.9. Source of Guns N' Roses album with PROV-O

```
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix wikicat: <https://en.wikipedia.org/wiki/Category:> .
ex:Guns_N'_Roses_album
    prov:wasDerivedFrom wikicat:Guns_N'_Roses_albums .
```

Schema.org

Schema.org is a collaborative effort to provide a schema for structured data on the Web [58]. It was initially founded by Google, Microsoft, Yahoo, and Yandex to serve as a shared schema for search engines. Since then, it has been updated continuously and is used, for example, as a top-level schema of large ontologies [152].

2.2 Knowledge Graphs

A KG is "*a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities*" [83]. Paulheim [147] formulated the key characteristics of a KG as

1. mainly describing real-world entities and their interrelations, organized in a graph,
2. defining possible classes and relations of entities in a schema,
3. allowing for potentially interrelating arbitrary entities with each other, and
4. covering various topical domains.

[16] `https://www.w3.org/TR/prov-o/`

The phrase Knowledge Graph was used in the literature already decades ago [175], but gained popularity with the announcement of the *Google Knowledge Graph* in 2012 [180]. Given the definition and key characteristics above, Weikum argues that the proper term for this construct is *Knowledge Base (KB)* as plain entity graphs do usually not support higher-arity relations, knowledge provenance, and consistency constraints [201]. In this work, we use the term *Knowledge Graph* inclusively as it is the more common terminology.

2.2.1 Components

A KG $\mathcal{K}$ with

$$\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{E}, \mathcal{L}, \mathcal{A}) \tag{2.1}$$

is a directed graph with types $\mathcal{T}$, properties $\mathcal{P}$, entities $\mathcal{E}$ and literals $\mathcal{L}$ serving as nodes and assertions $\mathcal{A}$ as edges of the graph.[17] Every assertion $a \in \mathcal{A}$ is a *triple* (s, p, o) where s and o are nodes in the graph and p is a property from a finite set of properties. With $\mathcal{P}$, $\mathcal{K}$ may define some of the properties in the form of nodes, but additional ones may be imported from other KGs or vocabularies.

Terminology Component

The *Terminology Component*, or *TBox*, of $\mathcal{K}$ contains the ontology of the graph. This includes the types $\mathcal{T}$, the properties $\mathcal{P}$, and the ontology-related triples in $\mathcal{A}$ with a subject in $\mathcal{T} \cup \mathcal{P}$ (see Examples 2.1.3, 2.1.5 and 2.1.6).

Assertion Component

The *Assertion Component*, or *ABox*, of $\mathcal{K}$ contains the data of the graph. The data is modelled using the concepts of the *TBox*. It describes entities $\mathcal{E}$ and their interrelations. The entities described in a KG are representations of real-world concepts or objects; through this anchoring, they are also called *named entities*.

The ABox contains all assertions related to entities in the graph (i.e., the subject of the assertion is always an element of $\mathcal{E}$). The assertions are either type assertions

$$\mathit{TA} = \{(s, p, o) : (s, p, o) \in \mathcal{A}, s \in \mathcal{E}, p{=}\mathit{rdf{:}type}\} \tag{2.2}$$

as shown in Example 2.1.4, or relation assertions

$$\mathit{RA} = \{(s, p, o) : (s, p, o) \in \mathcal{A}, s \in \mathcal{E}, p{\neq}\mathit{rdf{:}type}\} \tag{2.3}$$

[17] We use the term *assertions* instead of labelled edges as we are dealing with RDF graphs.

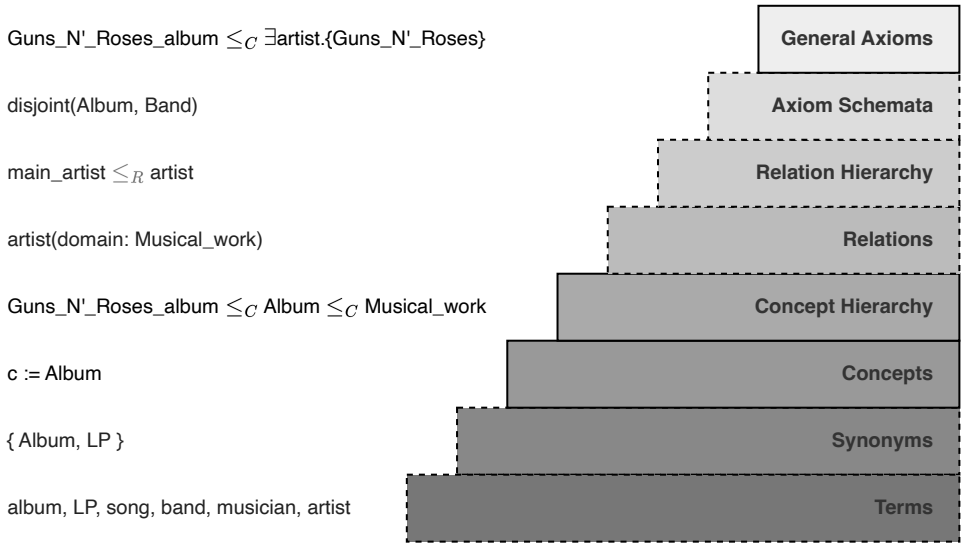


Figure 2.4: Ontology learning layer cake (adapted from [32]).

as shown in Example 2.1.2. We refer to the types of an entity e with

$$\mathcal{T}_e = \{o|(e, p, o) \in TA\}. \tag{2.4}$$

2.2.2 Construction and Extension

Knowledge Graph Construction (KGC), i.e., how to create a KG from scratch, and Knowledge Graph Extension (KGE), i.e., extending a KG with additional information, are broad research topics. In this section, we only introduce the main concepts and refer the interested reader to the related work sections in the dedicated chapters and respective surveys [83, 147, 163, 202, 222].

Ontology Learning

Ontology Learning is a field of research aiming at the automatic construction and maintenance of ontologies [105]. The term was coined by Mädche and Staab in 2001 [119] to describe a set of techniques supporting an ontology engineer in the task of creating and maintaining an ontology using data from structured, semi-structured, and unstructured sources. Due to the rapid progress in AI and Natural Language Processing (NLP) in the last two decades, techniques have been developed that try to minimize human involvement, with the goal of a fully automatic OC process.

The ontology learning layer cake in Fig. 2.4 shows the subtasks of learning an ontology. In this thesis, we focus on three of these layers in the context of Wikipedia:

- **Concepts**: Discover concepts (also: classes, types) of the ontology, i.e., create or extend $\mathcal{T}$.
- **Concept Hierarchy**: Learn a hierarchy of concepts, i.e., add subclass assertions to $\mathcal{A}$.
- **General Axioms**: Derive axioms expressing the semantics of concepts, i.e., add axioms describing discovered types or relations to $\mathcal{A}$.

Entity Linking

Entity Linking (EL)[18] is the process of detecting entity mentions in a given data source and linking them to an entity in a KG [163]. Through this *semantic grounding*, it is possible to (a) retrieve information about the entity from the KG to enrich the source or (b) extract information about the entity from the source to extend the KG. A mention that is mapped to an entity is also called a *surface form* or a *lexicalisation* of the entity.

EL can be broken down into the subproblems Named Entity Recognition (NER) and Named Entity Disambiguation (NED). The former is the task of identifying mentions of named entities in the source and the latter is the task of mapping the mention to an entity in the KG [111]. If there is no match for a mention in the KG, we speak of *NIL mentions* and *NIL entities*.

Example 2.2.1. EL for the *Gilby Clarke* page shown in Fig. 2.5

The following entity mentions in Fig. 2.5 may be identified and linked to a KG:
Guns N' Roses, The Spaghetti Incident?, Greatest Hits, Nancy Sinatra, California Girl, Rubber, Greatest Hits.

Entities can be categorized as *head* or *tail* entities depending on their popularity (e.g., how often a mention refers to one entity instead of other entities with similar lexicalisations or how much information the KG contains about an entity). Following Ilievski [87], we use the term *long-tail entities* for entities where information is scarce and which may or may not be contained in a KG (i.e., the union of tail and NIL entities). There is, however, no exact definition of when an entity is a (long-)tail entity as it strongly depends on the data source and KG at hand. For example, Chen et al. [28] define long-tail entities for Wikidata as entities with fewer than 14 triples, as this is the case for nearly 50% of entities in Wikidata.

Information Extraction

The goal of IE is to make the semantic structure of a data source explicit so that we can make use of it [56]. As this is a very broad objective, it includes

[18] As with *entities* and *named entities*, we synonymously use *Entity Linking* and *Named Entity Linking (NEL)* in the context of KGs.

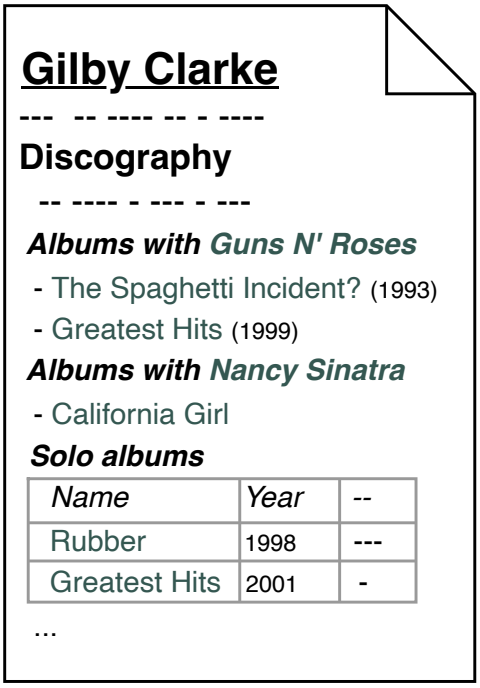


Figure 2.5: Excerpt of *Gilby Clarke* page taken from Fig. 1.1 on page 6.

the previously described field of EL as well, as we need to identify an entity in a text before extracting additional information about it. In this thesis, however, we use the term to refer to the task of extracting information about entities in a data source where the entities are already linked.

As described in Section 2.2.1, we can extend a given KG with additional information by adding types (extending *TA*) or relations (extending *RA*). We refer to the process of extracting types from a data source as *Entity Typing* and to extracting relations as *Relation Extraction*. How exactly types and relations are extracted relies strongly on the nature of the data source. Wikipedia contains a lot of semi-structured data which can be exploited to extract types and relations as shown in Example 2.2.2.

Example 2.2.2. IE for the *Gilby Clarke* page shown in Fig. 2.5

For the entities *The Spaghetti Incident?* and *Greatest Hits*, we derive that they are both albums (from *Albums with..*), and we derive the artist *Guns N' Roses* (from *..with Guns N' Roses*).

2.2.3 Paradigms

In the following, we describe prevalent paradigms that are frequently applied in KGC pipelines and approaches.

Local Closed-World Assumption

As KGs store information about the world and the world changes rapidly, it is safe to say that the majority KGs are incomplete and may contain incorrect information. Approaches that use assertions in KGs have to make assumptions about these assertions (e.g., whether they are correct), but also about assertions that are unavailable. Most KGs do not explicitly model negative assertions, meaning that the absence of a fact either means that it is incorrect or that the KG has no information about it. In the following, we use the theoretical construct $\mathcal{K}^* = (\mathcal{T}^*, \mathcal{P}^*, \mathcal{E}^*, \mathcal{L}^*, \mathcal{A}^*)$ representing the complete version of $\mathcal{K}$.

Adopting the Open-World Assumption (OWA), we don't make any assumptions about the absence of a fact, i.e., it either may be incorrect or simply missing ($\mathcal{K} \subset \mathcal{K}^*$). Adopting the Closed-World Assumption (CWA), in contrast, means that we assume that the KG is complete ($\mathcal{K} = \mathcal{K}^*$). The Local Closed-World Assumption (LCWA)[19] is a compromise between OWA and CWA as it assumes only some parts of the graph to be complete where it is most likely. If $\mathcal{K}$ makes some assertions with a predicate p for a subject s, then we assume that $\mathcal{K}$ contains every p-related information about s. More precisely, if $(s, p, o) \in \mathcal{K}$ then $\forall o' : (s, p, o') \in \mathcal{K}^* \implies (s, p, o') \in \mathcal{K}$. The assumptions are applied in Example 2.2.3.

Example 2.2.3. OWA, CWA, and LCWA in a KG

Given $\mathcal{A} = \{(The_Spaghetti_Incident?, artist, Gilby_Clarke)\}$, the assertion $(The_Spaghetti_Incident?, artist, Guns_N'_Roses)$ is *unknown* according to OWA, and *false* according to CWA and LCWA; the assertion $(Greatest_Hits, artist, Guns_N'_Roses)$ is *unknown* according to OWA and LCWA, and *false* according to CWA.

When considering assertions made in the SW field in general, where everyone can publish information about everything, one has to assume that any missing assumption may still be true, i.e., the OWA holds. When considering a single KG, however, adopting the LCWA may be beneficial as certain approaches (especially Machine Learning (ML) approaches) need definitive negative statements.

Distant Supervision

Distant Supervision is a paradigm for automatically gathering large amounts of training data with a low error rate using existing information in KGs or from other structured resources (e.g., semantic annotations in HTML pages [125]). In the original work, Mintz et al. [130] use Freebase as background knowledge to extract information from Wikipedia. Aprosio et al. [7] extend

[19] The LCWA is also called *Partial Completeness Assumption*.

this approach using DBpedia as background knowledge. The basic idea is to map entities in the KG to a data corpus to provide supervision signals for training without the need for manual annotation. Example 2.2.4 applies this process to a page in Wikipedia.

Example 2.2.4. Distant Supervision in Wikipedia

Given all entities and facts in Fig. 2.5 on page 22 are contained in $\mathcal{K}$, we would be able to learn several useful patterns, e.g.:

- all entities mentioned in listings of *Discography* are albums
- all albums have the artist *Gilby Clarke*
- albums mentioned in *Solo albums* have no other artist

Embeddings

KG embeddings are vector representations of entities and relationships in a KG, designed to capture semantic relationships and similarities between entities. Embeddings convert entities and relationships into continuous vector spaces, allowing for efficient and effective representation in ML models. Embeddings of a KG are static, i.e., they are fixed representations and change only if the underlying KG changes. Examples of notable embedding techniques focusing on link prediction (i.e., correctly predicting links in a KG) are $TransE$ [20], $TransR$ [110], and $DistMult$ [213]; a notable example of techniques focusing on data mining (i.e., correctly modelling semantic entity similarity) is RDF2vec [168].

Embeddings play a crucial role in NLP by assigning meaning to words and phrases (as opposed to entities in KGs). As the meaning of words changes with their context, recent approaches create *contextual embeddings* for words based on how they are used. The currently most successful models creating contextual embeddings are *Language Models (LMs)*, which employ a Transformer architecture [192] and are trained without supervision on massive amounts of data. Bidirectional Encoder Representations from Transformers (BERT) [37] is a seminal model in this area, with adaptations tuned for more robustness like Robustly Optimized BERT Pre-Training Approach (RoBERTa) [115] or efficiency like Distilled version of BERT (DistilBERT) [173].

These models, originally designed to create embeddings for textual resources, are also successfully applied to create embeddings in the KG context. Instead of texts, entity descriptions generated from KGs are fed to the model to create embeddings for entities. BLINK [210] is a seminal approach combining embeddings of texts and entities for EL. Its core idea is to create representations of mentions and entities with a Transformer model, retrieve mention-entity candidates through nearest-neighbour search in the embedding space, and identify matching pairs through binary classification.

2.3 Wikipedia

Wikipedia[20] is a free, open encyclopedia written and maintained by a community of volunteers. The project was started in 2001 and has grown continuously since then. In the last five years, around 200K articles have been added to the English Wikipedia per year [205]. It serves as a tremendously large and independent source of information for everyone. However, most of the information in Wikipedia is provided in a semi-structured or unstructured format. As this information can't be processed by applications directly, Wikipedia has been a focus point of IE efforts for the last two decades [9, 52, 81, 122, 182, 184, 208, 209]. In the following, we describe the Wikipedia structures relevant for the IE efforts in this thesis.

2.3.1 Editions

A language edition of Wikipedia is a Wiki instance in a language with a valid ISO 639 code that is "sufficiently unique" and has a "sufficient number of fluent users" [204]. As of January 2024, Wikipedia has 326 active language editions. With over 6.7M articles, the English Wikipedia is by far the largest edition, generating approximately 50% of the overall page views [203]. Wikipedia uses *interlanguage links* to connect articles describing the same topic across editions.

2.3.2 Wiki Markup

Wikipedia uses a markup language called *Wiki markup* for formatting and structuring its articles. It is designed to be simple and easy to learn, allowing users to contribute content without needing advanced technical knowledge. Example 2.3.1 shows an excerpt of the Wikipedia page of *Gilby Clarke* in Wiki markup.

Example 2.3.1. *Gilby Clarke* page shown in Fig. 2.5 in Wiki markup

```
==Discography==
===Albums with [[Guns N' Roses]]===
* [[The Spaghetti Incident?]] (1993)
* [[Greatests Hits (Guns N' Roses album)]] (1999)
===Albums with [[Nancy Sinatra]]===
* [[California Girl]]
===Solo albums===
{| class="wikitable"
|-
! Name
```

[20] https://www.wikipedia.org/

```
! Year
|-
| [[Rubber]]
| 1998
|-
| [[Greatest Hits (Gilby Clarke album]]
| 2001
|}
```

2.3.3 Content Types

Articles are the main source of information in Wikipedia, containing all the knowledge about a dedicated topic and linking to relevant related articles. Other page types like *List Pages* and *Categories* provide further information, for example, by structuring and grouping articles to facilitate navigation and exploration of specific topics.

Article

An *article* is a page or entry that provides information about a specific subject. Articles are the fundamental units of content in Wikipedia and cover a wide range of subjects, including people, places, events, concepts, or organizations. Wikipedia has guidelines ensuring articles are only created for topics with a certain *notability*.[21] Articles about topics that are widely unknown or deemed irrelevant are hence not published on Wikipedia.

Typically, an article begins with a short abstract about the subject, followed by several sections describing the subject from different perspectives. An article may have an *infobox*, a structured data box with a summary of key information appearing in the upper-right corner. As shown in the example of Fig. 2.5, an article may contain listings in the form of tables or enumerations to present information in a structured way.

List Page

List pages in Wikipedia are dedicated pages that compile and organize information about a set of related items. These pages serve as comprehensive lists, giving readers an overview of various subjects. However, list pages are no formal construct in Wikipedia but rather emerged through the community as a special kind of article. For example, the page *List of greatest hits albums*[22] contains an enumeration of compilation albums in alphabetical order. Contrary to listings in articles, the listings in list pages follow a consistent layout and are only used for listing articles that share a common property.

[21] `https://en.wikipedia.org/wiki/Wikipedia:Notability`
[22] `https://en.wikipedia.org/wiki/List_of_greatest_hits_albums`

Category

Contrary to list pages, *categories* are a formal construct in Wikipedia and serve the purpose of categorizing pages in a hierarchical structure. This structure, the Wikipedia Category Graph (WCG), is a directed but not acyclic graph. It does not only contain categories used for categorising articles thematically but also ones used for administrative purposes (e.g., the category *Wikipedia articles in need of updating*[23]). The WCG has been used extensively for taxonomy induction and has yielded highly accurate results [47, 120]. A subgraph of the WCG contains *list categories* (i.e., categories starting with the prefix *Lists of*), which organize many of the list pages in Wikipedia. For example, the category *Lists of albums*[24] contains the list page *List of greatest hits albums*.

Other

Other content types include templates, which are reusable pieces of code or markup that help maintain consistency across articles and simplify the editing process. They are configured and referenced directly in the Wiki markup of a page. For example, infoboxes are realized via templates. A dedicated infobox template is implemented for every type of article (e.g., *musicians*). Further content types are *disambiguation pages* for the disambiguation of homonymous articles and *redirects* for linking synonymous articles.

2.3.4 Versions

Throughout this thesis, we use three different versions of Wikipedia, which we denote as follows:

- **Wikipedia2016**: A dump of the English Wikipedia from October 2016.
- **Wikipedia2020**: A dump of the English Wikipedia from October 2020.
- **Wikipedia2022**: A dump of the English Wikipedia from August 2022.

Wikipedia dumps can be downloaded at `https://dumps.wikimedia.org/`.

[23] `https://en.wikipedia.org/wiki/Category:Wikipedia_articles_in_need_of_updating`

[24] `https://en.wikipedia.org/wiki/Category:Lists_of_albums`

CHAPTER 3

Knowledge Graphs on the Web

KGs are an emerging form of knowledge representation. While Google placed the term *Knowledge Graph* in the business domain and promoted it as a means to improve their search results, they are used in many applications today. While companies such as Google, Microsoft, and Facebook have their own non-public KGs, there is also a larger body of publicly available KGs, such as DBpedia or Wikidata. This chapter provides an overview and a comparison of those publicly available KGs. We introduce them according to their extraction techniques and give insights into their contents, size, coverage, and mutual overlap.

The work presented in this chapter is based on the following publication:

Nicolas Heist, Sven Hertling, Daniel Ringler and Heiko Paulheim. Knowledge Graphs on the Web – an Overview. In *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, vol. 47, pp. 3-22, 2020, IOS Press.[1,2,3] [66]

[1]The contributions of the author to this publication are as follows: computation of KG statistics, analysis of overlap and gain between KGs.

[2]The original publication includes an initial version of CaLiGraph in the comparison. We omit it here as this thesis has a dedicated chapter with a detailed analysis of CaLiGraph.

[3]The original publication considers only assertions in the namespace of the KG and lacks deduplication at some points; we reran the experiments to consider all assertions exactly once for a more consistent comparison.

3.1 Overview

There are different techniques for creating KGs. The most common ones are (1) manual curation, (2) creation from (semi-)structured sources, and (3) creation from unstructured sources. Some KGs also use a mix of those techniques.

3.1.1 Manual Curation

Cyc [108] is one of the oldest KGs; the Cyc project dates back to the 1990s. Cyc was created along with its own language (CycL), which provides a large degree of formalization.

While a comparatively small group of experts developed Cyc, the idea of *Freebase* [185] was to establish a large community of volunteers, similar to Wikipedia. To that end, the schema of Freebase was kept fairly simple to lower the entrance barrier as much as possible. Freebase was acquired by Google in 2010 and shut down in 2014.

Wikidata [195] also uses a crowd-editing approach. In contrast to Cyc and Freebase, Wikidata also imports large datasets, such as several bibliographies of national libraries. Porting the data from Freebase to Wikidata is also a long standing goal [185].

Curating a KG manually can be a huge effort. The total cost of development for Cyc has been estimated as 120 Million USD. This corresponds to a total cost of 2-6 USD per single axiom in Cyc [148].

3.1.2 Creation from (Semi)-Structured Sources

A more efficient way of KGC is the use of structured or semi-structured sources. Wikipedia is a commonly used starting point for KGs such as *DBpedia* [104] and *Yet Another Great Ontology (YAGO)* [184].

DBpedia mainly uses infoboxes in Wikipedia. Those are manually mapped to a pre-defined ontology; the mapping is crowd-sourced using a Wiki and a community of volunteers. Given those mappings, the DBpedia Extraction Framework creates a graph in which each page in Wikipedia becomes an entity, and all values and links in an infobox become attributes and edges in the graph.

YAGO is built on the idea of combining a small but well-crafted top-level schema with a large but messy taxonomy, thereby creating a unified and cleaned schema. YAGO uses a process similar to DBpedia but classifies instances based on the category structure and WordNet [128] instead of infoboxes. Further, other data sources from various domains are ingested. For example, YAGO integrates various language editions of Wikipedia into a single graph and represents temporal facts with meta-level statements, i.e.,

RDF reification. From version 4 on, YAGO uses Wikidata and Schema.org as primary data sources [152].

A similar approach, i.e., the combination of information in Wikipedia and WordNet, is used by *BabelNet* [138]. The main purpose of BabelNet is the collection of synonyms and translations in various languages, so this KG is particularly well suited for supporting multi-language applications. Similarly, *ConceptNet* [181] collects synonyms and translations in various languages, integrating multiple third-party KGs itself.

DBkWik [75] uses the same codebase as DBpedia but applies it to a multitude of Wikis like Jedipedia[4] or Music Hub.[5] This leads to a graph with larger coverage and level of detail for many long-tail entities and is highly complementary to DBpedia. However, the absence of a central ontology and mappings, as well as the existence of duplicates across Wikis, which might not be trivial to detect, impose several challenges not present in DBpedia.

Another structured data source is contained in structured annotations in Web pages using techniques such as RDFa, Microdata, and Microformats [127]. While the pure collection of those could, in theory, already be considered a KG, that graph would be rather disconnected and consist of a plethora of small, disconnected components [146]. It would require additional cleanup for compensating irregular use of the underlying schemas and shortcomings in the extraction [126]. A consolidated version of this data into a more connected KG has been published under the name *VoldemortKG* [188].

3.1.3 Creation from Unstructured Sources

Extracting a KG from semi-structured sources is considered easier than from unstructured sources. However, there is much more information in unstructured sources (such as text). Therefore, extracting knowledge from unstructured sources has also been proposed.

Never-Ending Language Learner (NELL) [27] is an example of extracting a KG from free text. NELL was originally trained with a few seed examples and continuously runs an iterative coupled learning process. In each iteration, facts are used to learn textual patterns to detect those facts, and patterns learned in previous iterations are used to extract new facts, which serve as training examples in later iterations. NELL introduced a feedback loop incorporating occasional human feedback to improve the quality.

WebIsA [177] also extracts facts from free text but focuses on creating a large-scale taxonomy. For each extracted fact, rich metadata are collected, including the sources, the original sentences, and the patterns used to extract a particular fact. Those metadata are exploited to compute a confidence

[4] `https://jedipedia.fandom.com/`
[5] `https://music.fandom.com/`

score for each fact [74].

3.2 Comparison

Whenever a KG is to be used in an application, it is important to determine which KG is best suitable for the application at hand. The KGs mentioned above differ in their content, level of detail, etc. Hence, this chapter will discuss several characteristics of KGs and provide insights into their differences.

3.2.1 General Metrics

The most straightforward metrics consider the mere amount of information contained in a KG. Measures that may be used include:

- The number of instances in a graph
- The number of assertions (or edges between entities)
- The average and median linkage degree (i.e., how many assertions per entity does the graph contain?)

These metrics hint at the utility of a KG – the more information about the domain at hand is present (i.e., the more instances are represented in the KG and the more detailed that information is), the more can an application benefit in providing better results or better interpretations.

Another set of metrics can be defined for the schema or ontology level of a KG:

- The number of classes defined in the schema
- The number of relations defined in the schema
- The average depth and width (branching factor) of the class hierarchy[6]
- The complexity of the schema

While the instance-based metrics focus more on the coverage of a domain in a KG, these schema-level metrics provide information about the richness and formality of that knowledge. They determine which techniques to use – e.g., while more formal, very complex ontologies will call for using ontology reasoning, lightweight but large-scale ontologies will be better exploited by statistical and distributional approaches.

Table 3.1 depicts those metrics for some of the KGs discussed above. ConceptNet and WebIsA are omitted since they do not distinguish a schema and instance level (i.e., there is no specific distinction between a class and an instance), which does not allow for computing those metrics meaningfully. For Cyc, which is only available as a commercial product today, we used the free version OpenCyc, which has been available until 2017. Appendix A.1 holds a complete list of data sources used in this chapter.

[6]While this could also be done for the property hierarchy, extensive property hierarchies are rather rare in common KGs.

	DBpedia	YAGO	Wikidata	BabelNet
# Instances	5,044,223	6,349,359	52,252,549	7,735,436
# Assertions	71,630,413	263,433,941	1,763,622,910	178,982,397
Avg. linking degree	2.77	1.89	6.30	0.00
Median ingoing edges	0	0	1	0
Median outgoing edges	15	35	73	9
# Classes	760	819,292	2,356,259	6,044,564
# Relations	1355	77	6,236	22
Avg. depth of class tree	3.51	6.61	6.43	4.11
Avg. branching factor of class tree	4.53	8.48	36.48	71.0
Ontology complexity	$\mathcal{SHOFD}$	$\mathcal{SHOIF}$	$\mathcal{SOD}$	$\mathcal{SO}$
	Cyc	NELL	Voldemort	
# Instances	122,441	5,120,688	55,861	
# Assertions	1,123,448	40,341,016	633,997	
Avg. linking degree	3.34	4.20	0	
Median ingoing edges	0	0	0	
Median outgoing edges	3	3	5	
# Classes	116,821	1,187	621	
# Relations	148	440	294	
Avg. depth of class tree	5.58	3.13	3.17	
Avg. branching factor of class tree	5.62	6.37	5.40	
Ontology complexity	$\mathcal{SHOIFD}$	$\mathcal{SROIF}$	$\mathcal{SH}$	

Table 3.1: General metrics of open KGs.

From those metrics, it can be observed that the KGs differ in size by several orders of magnitude. The sizes range from 50,000 instances (in Voldemort) to 50 million instances (in Wikidata), so the latter is larger by a factor of 1,000; the same holds for assertions. Concerning the linkage degree, Wikidata is much more richly linked than the other graphs.

Fig. 3.1 shows an overview of the KGs considered. We follow the conventions of the LOD cloud, which are used to depict linked datasets and their connections (cf. Fig. 2.3 on page 16). The size of the circles is proportional to the number of instances, and the strength of the connecting lines is proportional to the number of links [174].

The KGs also differ strongly in the characteristics of their schema. DBpedia and NELL have comparably small schemas, while Wikidata and BabelNet build deep and detailed taxonomies. For example, while NELL does not define detailed subclasses for *Scientist*,[7] DBpedia defines four subclasses,[8] and Wikidata has more than 600.[9] Voldemort, on the other hand, reuses the Schema.org ontology, which is comparably small [124].

Looking at the complexity, it is unsurprising that Cyc, originating in classic AI research and strongly building on logical rules [24, 108], has the highest complexity. Wikidata, BabelNet, and Voldemort have low complexity; the other graphs are somewhere in between.

[7] `http://rtw.ml.cmu.edu/rtw/kbbrowser/pred:scientist`
[8] `http://dbpedia.org/ontology/Scientist`
[9] `https://www.wikidata.org/wiki/Q15976092`

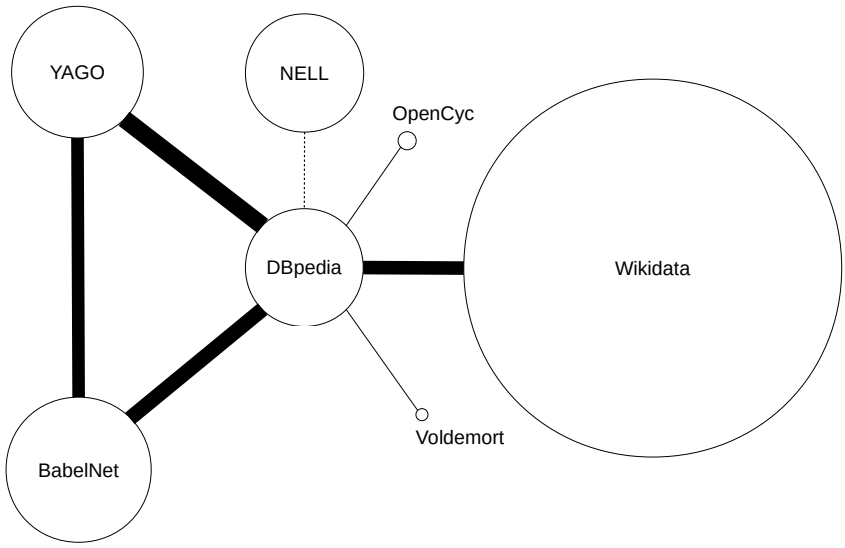


Figure 3.1: Depiction of the size and linkage degree of open KGs. Although NELL and DBpedia are not explicitly interlinked, NELL contains links to Wikipedia, which can be trivially translated to DBpedia links.

3.2.2 Contents

The KGs do not only differ in their size and level of detail but also in their contents. The most straightforward way to assess the content focus of a KG is to look at the size of its classes. Figs. 3.2 to 3.8 show graphic depictions of those class sizes. The diagrams were created starting from the most abstract class and following the class hierarchy to the largest respective subclasses.

At first glance, the figures reveal differences in the development of the taxonomies. While Cyc builds a formal ontology with very abstract top-level categories such as *partially intangible thing* or *thing that exists in time*, the more pragmatic classification in DBpedia and Voldemort (the latter using Schema.org as an ontology) has top-level classes such as *Place* or *Person*. The reason for these differences lies in the origins of the respective KGs: While Cyc's classification was created by AI researchers, the ontology in DBpedia is the result of a crowdsourcing process [148]. The same holds for Schema.org, which is a pragmatic effort of a consortium of search engine developers.

Moreover, the diagrams reveal some differences in the contents. The main focus of DBpedia is on persons (and their careers), as well as places, works, and species. Wikidata also strongly focuses on works (mainly due to the import of entire bibliographic datasets), while Cyc, BabelNet and NELL show a more diverse distribution.

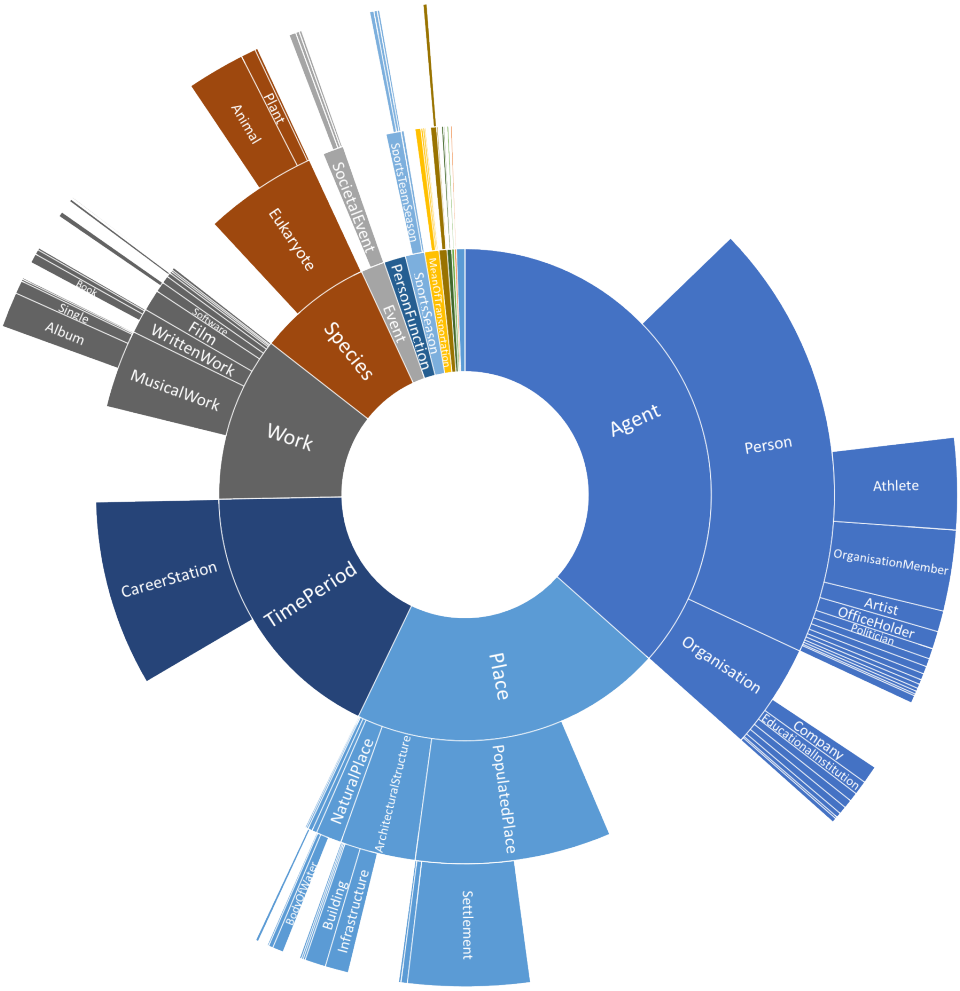


Figure 3.2: Instances in DBpedia

3.2.3 Looking into Details

To obtain deeper insights into which classes are more prominent in which KGs, and, ultimately, which KGs are suitable for building AI systems in a specific domain, it is useful to not only look at the number of instances but also the level of detail in which those instances are represented (i.e., the linkage degree and number of assertions per instance).

Table 3.2 depicts such a detailed view for ten prominent classes:

- Person
- Organization
- Populated place (city, country, etc.)
- Uninhabited place (mountain, lake, etc.)
- Species

Figure 3.3: Instances in YAGO

- Work (book, movie, etc.)
- Building
- Gene
- Protein
- Event

The global trend observed in this table is that Wikidata has the largest number of instances and the largest detail level in most classes. However, there are differences from class to class. While Wikidata contains a large number of works, YAGO is a good source of events. NELL often has fewer instances, but a larger level of detail, which can be explained by its focus on more prominent instances.

The contrast between the average and the median degree reveals a few differences. For example, BabelNet's instance counts are similar to DBpedia's

	DBpedia				YAGO				Wikidata			
Class	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out
Person	1,243,400	1.55	0	19	2,213,431	0.07	0	53	5,250,840	10.14	2	44
Organization	286,482	10.3	0	21	498,750	4.38	0	33	1,665,319	41.40	1	15
Populated place	513,642	7.43	0	18	319,210	4.96	0	40	2,355,559	4.57	1	25
Uninhabited place	67,495	0.93	0	17	160,615	0.65	0	26	1,516,890	1.31	1	19
Species	306,104	2.57	0	16	2,553,369	0.06	0	50	110	22.56	1	41
Work	496,070	0.82	0	18	1,175,125	0.92	0	27	34,585,828	2.84	1	14
Building	197,831	0.42	0	15	274,606	0.27	0	32	2,291,168	1.60	1	15
Gene	4	0.50	0.5	14.5	12,351	0.00	0	12	172,128	2.27	1	20
Protein	2,747	0.05	0	7	10,935	0.00	0	33	84,163	2.15	2	27
Event	76,029	1.97	0	14	562,583	1.22	0	31	579,559	2.74	1	12

	BabelNet				Cyc				NELL			
Class	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out	Instances	Avg. Deg.	Med-in	Med-out
Person	2,384,065	0.00	0	17	12,784	0.04	0	3	90,601	8.93	0	5
Organization	764,662	0.00	0	12	26,276	5.70	0	5	41,646	6.31	0	4
Populated place	509,257	0.01	0	9	8,596	20.63	0	12	28,359	39.98	0	5
Uninhabited place	70,209	0.02	0	11	64	2.05	1	12	158,879	3.83	0	3
Species	6,536	0.01	0	17	0	-	-	-	3,273	0.87	0	4
Work	491,057	0.00	0	12	19,908	0.91	0	2	27,038	1.09	0	4
Building	520	0.00	0	8	786	0.14	0	4	50,699	4.51	0	4
Gene	522	0.00	0	5	8	0	0	3	0	-	-	-
Protein	10,399	0.00	0	3	0	-	-	-	0	-	-	-
Event	9,904	0.00	0	13	685	0.86	0	2	37,203	0.65	0	4

	Voldemort			
Class	Instances	Avg. Deg.	Med-in	Med-out
Person	36,370	0.00	0	5
Organization	5,984	0.00	0	1
Populated place	1,278	0.00	0	5
Uninhabited place	60	0.00	0	4
Species	0	-	-	-
Work	6,673	0.00	0	3
Building	108	0.00	0	5
Gene	0	-	-	-
Protein	0	-	-	-
Event	198	0.00	0	4

Table 3.2: Detail statistics for selected classes in open KGs.

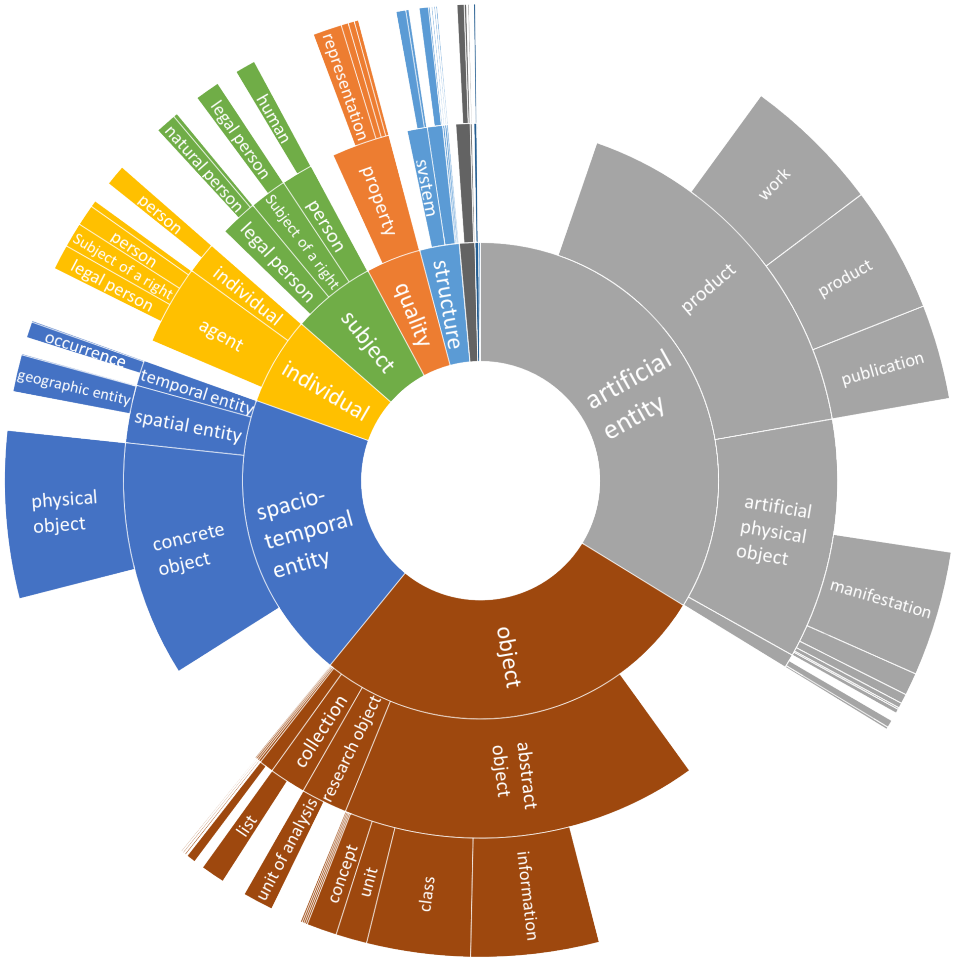


Figure 3.4: Instances in Wikidata

for some classes, e.g., uninhabited places or works. While the average linkage degree is higher in DBpedia, the median is comparable to BabelNet. This hints at a more uneven distribution of information in DBpedia, while BabelNet has a more constant distribution of statements per instance.

3.3 Linkage and Overlap

Since KGs differ so strongly in size, coverage, and level of detail, combining information from multiple KGs for implementing one application is often beneficial. To estimate the value of such a combination, we determine the overlap of the KGs.

As shown in Fig. 3.1, many KGs contain explicit interlinks. Those links, usually in the form of `owl:sameAs` links, express that entities in two KGs are

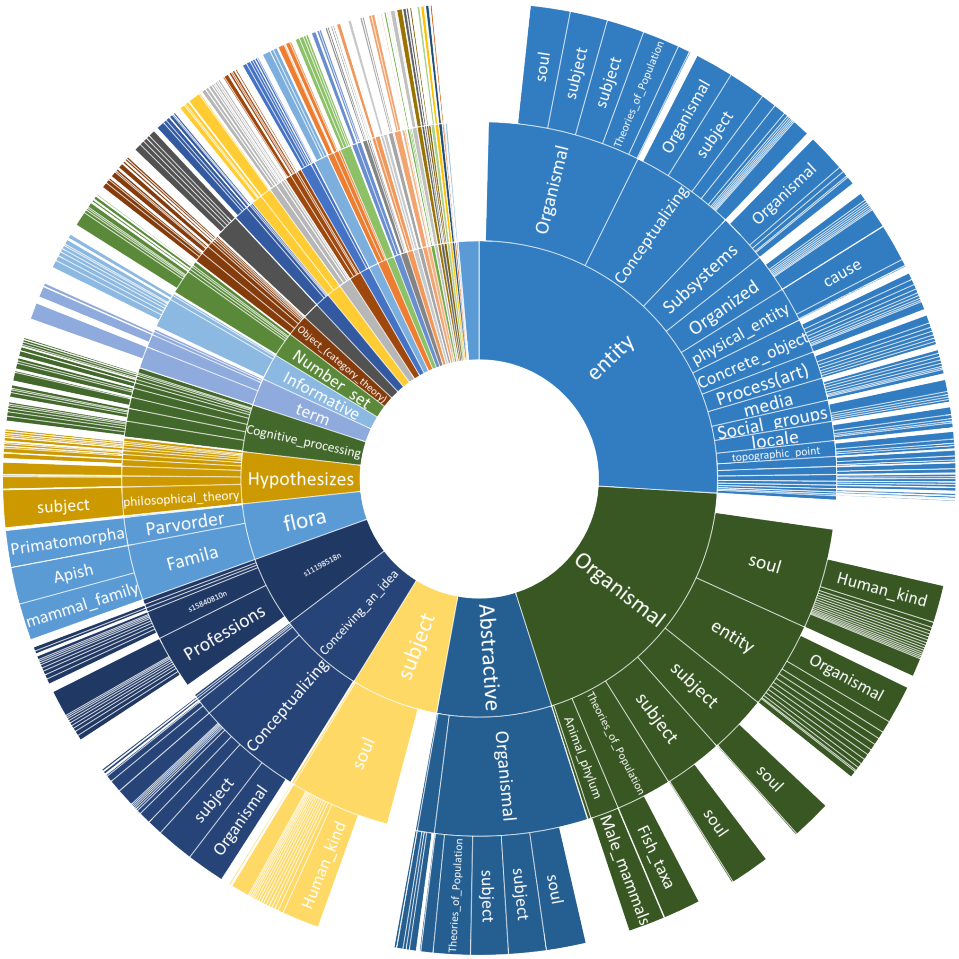


Figure 3.5: Instances in BabelNet

the same (or, more precisely, that they refer to the same real-world entity) [60]. In other cases, such links can be generated indirectly, e.g., if a KG contains links to Wikipedia pages, which can be easily mapped to entities in DBpedia and YAGO.

Even if those links provide a first hint at the overlap of KGs, and further links can be found by exploiting the transitivity of the `owl:sameAs` property [12], they do not provide a complete picture. Due to the OWA, which holds for KG interlinks as well, there might always be more links than the ones which are explicitly or implicitly provided by the KGs.

3.3.1 Method

To estimate the actual number of interlinks, we use a method first discussed in Ringler and Paulheim [165], which builds on a set of existing links and

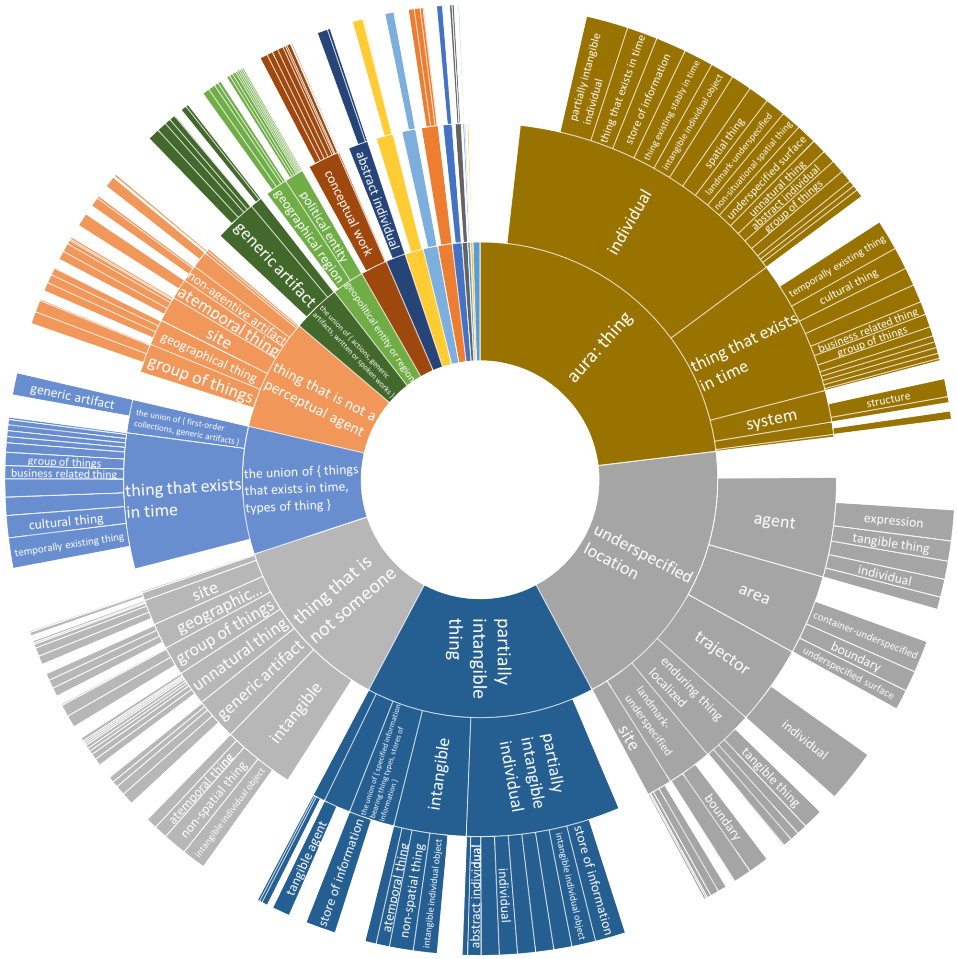


Figure 3.6: Instances in OpenCyc

heuristic link discovery:

1. We use different heuristics to discover links between two KGs automatically, e.g., different string similarity measures [45, 139].

2. Based on the existing, incomplete set of interlinks, we measure recall and precision of the individual heuristics [170].

3. With the help of those recall and precision figures, we can estimate the actual number of interlinks. After repeating the procedure with multiple heuristics, we can use the average of those estimates.

Given that the actual number of links is C (which is unknown), the number of links found by a heuristic is F, and that the number of correct links in F

Figure 3.7: Instances in NELL

is F^+, we define recall and precision as

$$R := \frac{|F^+|}{|C|} \tag{3.1}$$

$$P := \frac{|F^+|}{|F|} \tag{3.2}$$

Resolving by $|F^+|$ and combining the equations, we estimate $|C|$ as

$$|C| = |F| \cdot P \cdot \frac{1}{R} \tag{3.3}$$

Thus, we can estimate C given F, R, and P. A more intuitive interpretation of the last equation is that P is a measure of how strongly the heuristic *over*estimates the number of actual interlinks (thus, F is reduced by

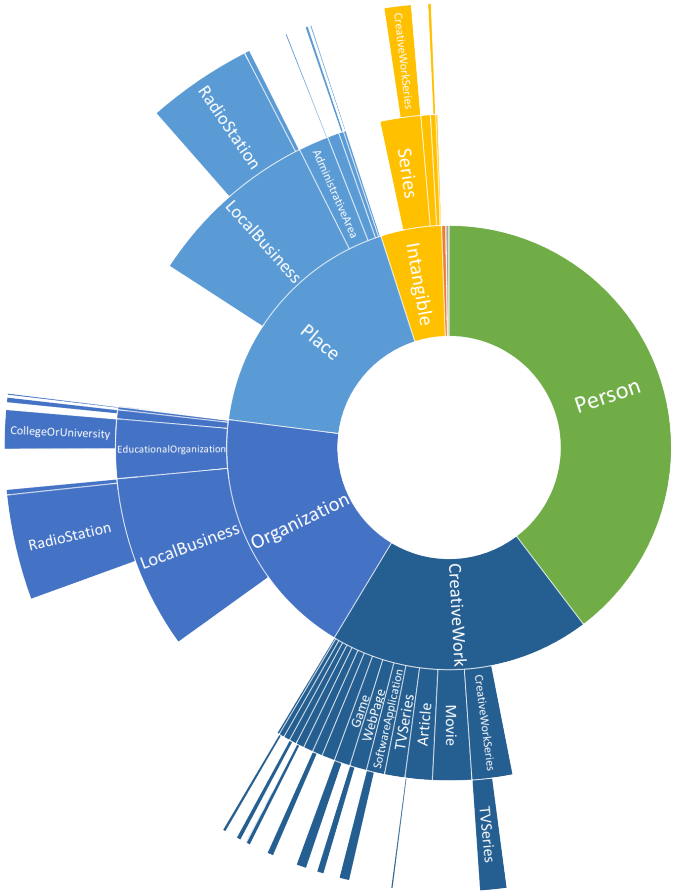


Figure 3.8: Instances in Voldemort

multiplication with P), and R is a measure of how strongly the heuristic *under*estimates the number of actual interlinks (thus, F is divided by R).

Ringler and Paulheim [165] have shown that although F varies greatly across different heuristics, the estimate C is fairly stable. For producing the estimates in this chapter, we have used the following heuristics: string equality, scaled Levenshtein (thresholds 0.8, 0.9, and 1.0), Jaccard (0.6, 0.8, and 1.0), Jaro (0.9, 0.95, and 1.0), JaroWinkler (0.9, 0.95, and 1.0), and MongeElkan (0.9, 0.95, and 1.0). We report the estimated overlap as the average of these 16 metrics.

3.3.2 Findings

To analyze the benefit of the combination of different KGs, we depict the number of estimated links both in relation to (a) the entities existing in the larger of the two KGs (Fig. 3.9) as well as (b) in relation to the links that exist explicitly or implicitly (Fig. 3.10). From (a), we can estimate the gain

in knowledge of combining two KGs (i.e., if only a small fraction of one KG is also contained in the other and vice versa, such a combination adds a lot of information). From (b), we can get insights into whether or not the set of existing links is sufficient for such a combination.

Fig. 3.9 shows that in most cases, the larger of two KGs contains most of the entities of the smaller one, i.e., the set of entities of a class in the larger KG is usually a superset of that set in the smaller one. For example, as depicted in Table 3.2, Wikidata contains about twice as many persons as DBpedia and YAGO. A value close to zero for the overlap implies that DBpedia and YAGO contain almost no persons which are not contained in Wikidata. In conclusion, combining Wikidata with DBpedia or YAGO for better coverage of the *Person* class would not be beneficial.

A notable exception is BabelNet, which often contains complementary instances. For example, DBpedia and BabelNet contain 1.2M and 2.4M instances of the class *Person*, respectively, while DBpedia and BabelNet together are estimated to 2.9M instances of the class *Person*. The reasons for the high complementarity of DBpedia/YAGO and BabelNet are their sources (only English Wikipedia vs. multiple language editions) and extraction mechanisms.

Fig. 3.10 shows that the linkage between DBpedia, YAGO and BabelNet is mostly complete (i.e., most of the common instances are also explicitly linked). This is not very surprising since they are all generated from Wikipedia with different means. On the other hand, NELL, OpenCyc, and Voldemort have a much lower degree of linkage. This shows that links between KGs are only complete where they are trivial to create, and combining different KGs otherwise requires improving the interlinking as a preliminary step.

3.4 Conclusion and Outlook

This chapter provided an overview of publicly available, cross-domain KGs on the Web. We have compared them according to different metrics, which might be helpful to implement an AI application in a given domain.

Besides the metrics used for this comparison, there are quite a few more which help in selecting and assessing a given KG. For example, data quality in KGs has not been considered in this chapter since there are already quite elaborate surveys covering this aspect [42, 214].

So far, we have measured the overlap of KGs only based on entities. Another helpful metric would be the overlap on the statement level. Even if two KGs cover the same entity, the information they contain about that entity might still be complementary. For example, for the entity *University of Mannheim*, DBpedia has the exact number of undergraduate students, PhD

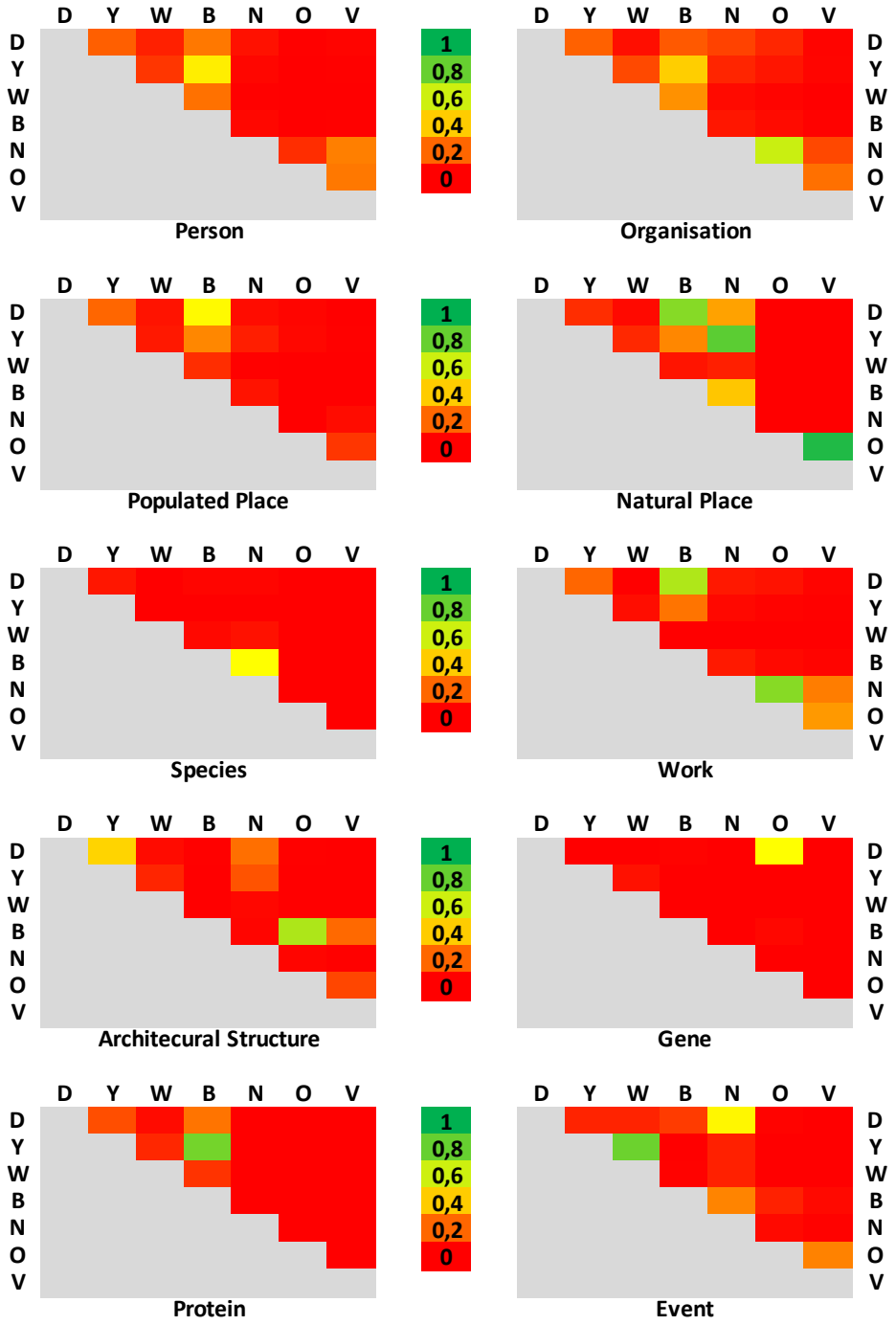


Figure 3.9: Fraction of entities in a pair of KGs which is *not* contained in the larger of the two graphs.

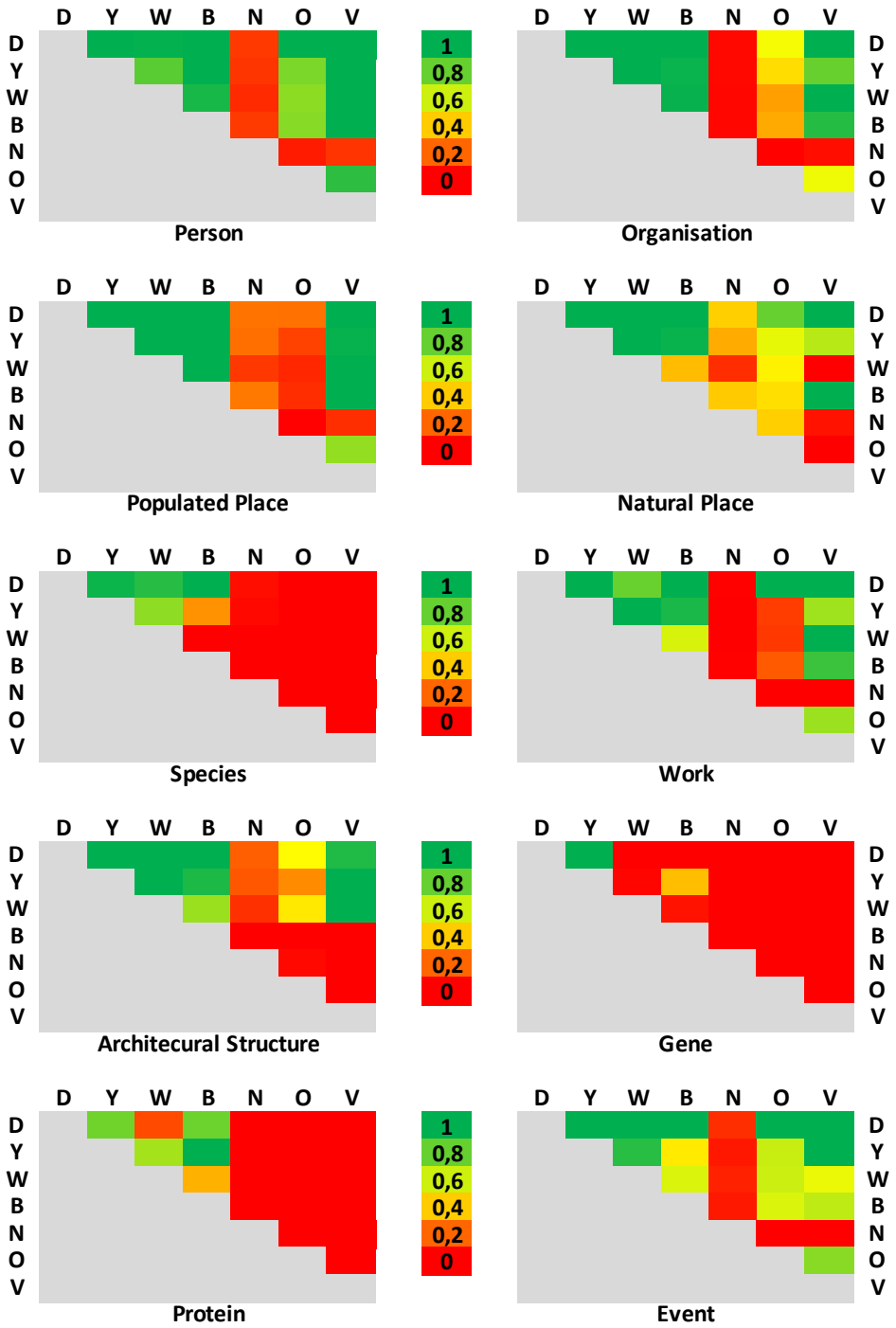


Figure 3.10: Existing entities in two KGs in relation to the number of links.

students, and so on.[10] Wikidata lists all faculties[11] and contains a list of researchers employed at the university.[12] The density of information differs as well: while YAGO lists 3 alumni of the University of Mannheim,[13] DBpedia lists 11 and Wikidata even 85 alumni.[14] Even contradicting information can be found [22]: for example, DBpedia and Wikidata provide a different number of students, and Wikidata and YAGO provide different founding dates of the University of Mannheim.

Developing cross-domain KGs is an active field of research, and new developments emerge occasionally. They differ in the data they use and/or the method of extraction:

- *DBkWik* [75, 76, 82] uses the extraction mechanism of DBpedia and applies it to a multitude of Wikis. The intermediate result is a collection of a few thousand isolated KGs, which must be integrated into a coherent joint KG [78].
- *Chaudron* [183] uses Wikipedia as a source and focuses on quantifiable values (e.g., sizes, weights, etc.). Besides the mere extraction, Chaudron uses sophisticated methods for recognizing and converting units of measurement.
- The Linked Hypernym Dataset (*LHD*) [96], like the aforementioned WebIsALOD, focuses on the extraction of a hypernym graph. It uses a deep linguistic analysis of the first paragraph in Wikipedia.
- *ClaimsKG* [186] extracts claims from fact-checking Web pages, such as politifact, and interlinks them with other KGs, such as DBpedia, which also allows for finding related claims.

The methods discussed in this chapter can be used to assess those emerging KGs and discuss their added value over existing ones. So, for example, for the above-mentioned DBkWik, we found that it is highly complimentary to DBpedia: 95% of all entities in DBkWik are not contained in DBpedia and vice versa.

[10] `http://dbpedia.org/page/University_of_Mannheim`
[11] `https://www.wikidata.org/wiki/Q317070`
[12] `https://w.wiki/7UU`
[13] `https://bit.ly/2U4wL0A`
[14] `https://w.wiki/7UV`

CHAPTER 4

Automated Knowledge Graph Construction

Having reviewed the content of all kinds of general-purpose open KGs in the previous chapter, we focus on the construction mechanisms of these KGs in this chapter. We are interested in AKGC, so we only consider automatically extracted KGs. Apart from manually curated KGs, this excludes KGs relying on human-in-the-loop mechanisms [157] or dataset-dependent mappings (e.g., via RDF Mapping Language (RML) [8, 86]) to extract instance data.

We define a pipeline for AKGC, which we then use to compare the extraction mechanisms of the KGs DBpedia, YAGO, NELL, BabelNet, and DBkWik. We formulate advantages and disadvantages to compile them into a list of open challenges in AKGC.

The work presented in this chapter is based on the following submission:

Nicolas Heist and Heiko Paulheim. CaLiGraph: A Knowledge Graph from Wikipedia Categories and Lists. In *Semantic Web Journal (SWJ)*, 2024. [under review]

4.1 A Pipeline for Automated Knowledge Graph Construction

AKGC is typically not an end-to-end ML task but consists of multiple steps, each with unique requirements and challenges [201]. Fig. 4.1 lists the steps in the order they will be addressed in the CaLiGraph extraction framework, together with actual examples. The pipeline consists of the two high-level blocks of *Ontology Construction (OC)* and *Knowledge Graph Population (KGP)*, with the former being responsible for the definition of the ontology necessary to describe the domain (TBox) and the latter being responsible for populating

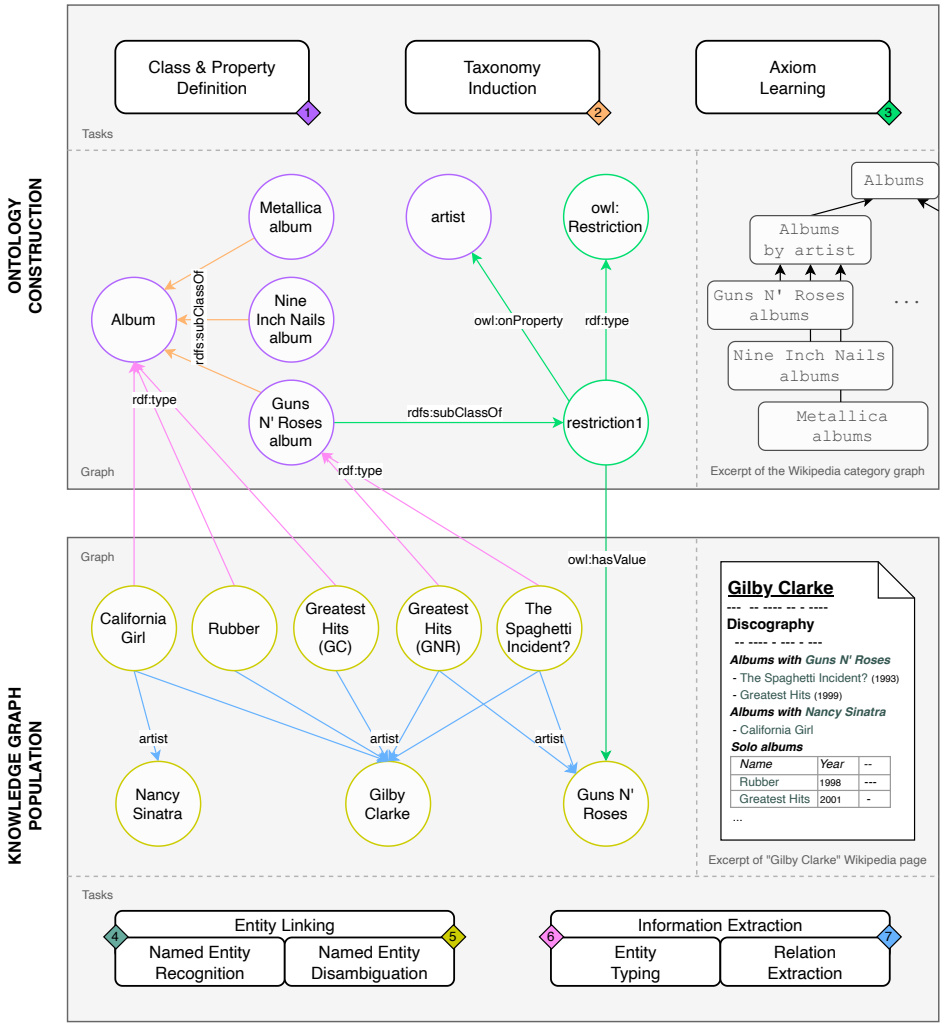


Figure 4.1: A pipeline for AKGC (taken from Fig. 1.1 on page 6).

the graph with data using concepts of the ontology (ABox). Whether the steps are executed once or iteratively, in this sequence or another, depends on the KG to be extracted.
OC steps:

1. **Class & Property Definition** Define relevant classes and properties of the domain

2. **Taxonomy Induction** Discover hierarchical relationships among classes and properties

3. **Axiom Learning** Formulate constraints for classes (e.g., disjointnesses) and properties (e.g., domains/ranges)

Knowledge Graph Population (KGP) steps:

4. **Named Entity Recognition** Identify mentions of named entities in a given data corpus

5. **Named Entity Disambiguation** Add the mentions to the KG by creating new or updating existing entities

6. **Entity Typing** Discover type assertions for the entities in the KG using the data corpus

7. **Relation Extraction** Discover relation assertions for the entities in the KG using the data corpus

While the steps in the OC block may be conducted manually for a sufficiently small domain, the steps in the KGP block are always automated processes using a pre-defined data corpus.

4.2 Construction of General-Purpose Knowledge Graphs

Given the pipeline above, we discuss the construction processes of automatically extracted general-purpose KGs. We only consider publicly accessible KGs and disregard closed-source industry-created KGs like those from Microsoft, Facebook, Amazon or ebay [142]. Fig. 4.2 shows a timeline with the major milestones and data sources of the public KGs discussed in the following.

4.2.1 DBpedia

OC DBpedia provides a Mappings Wiki[1] where the community defines classes, properties, datatypes and restrictions. Further, they map infoboxes to types in the schema and infobox keys to properties. The extraction is conducted automatically based on these definitions.

KGP DBpedia mints one entity per article in Wikipedia. A disambiguation of entities is unnecessary as they are annotated with links in the markup. Type assertions are derived from infobox types, and relation assertions are derived from infobox keys.

[1] `https://mappings.dbpedia.org`

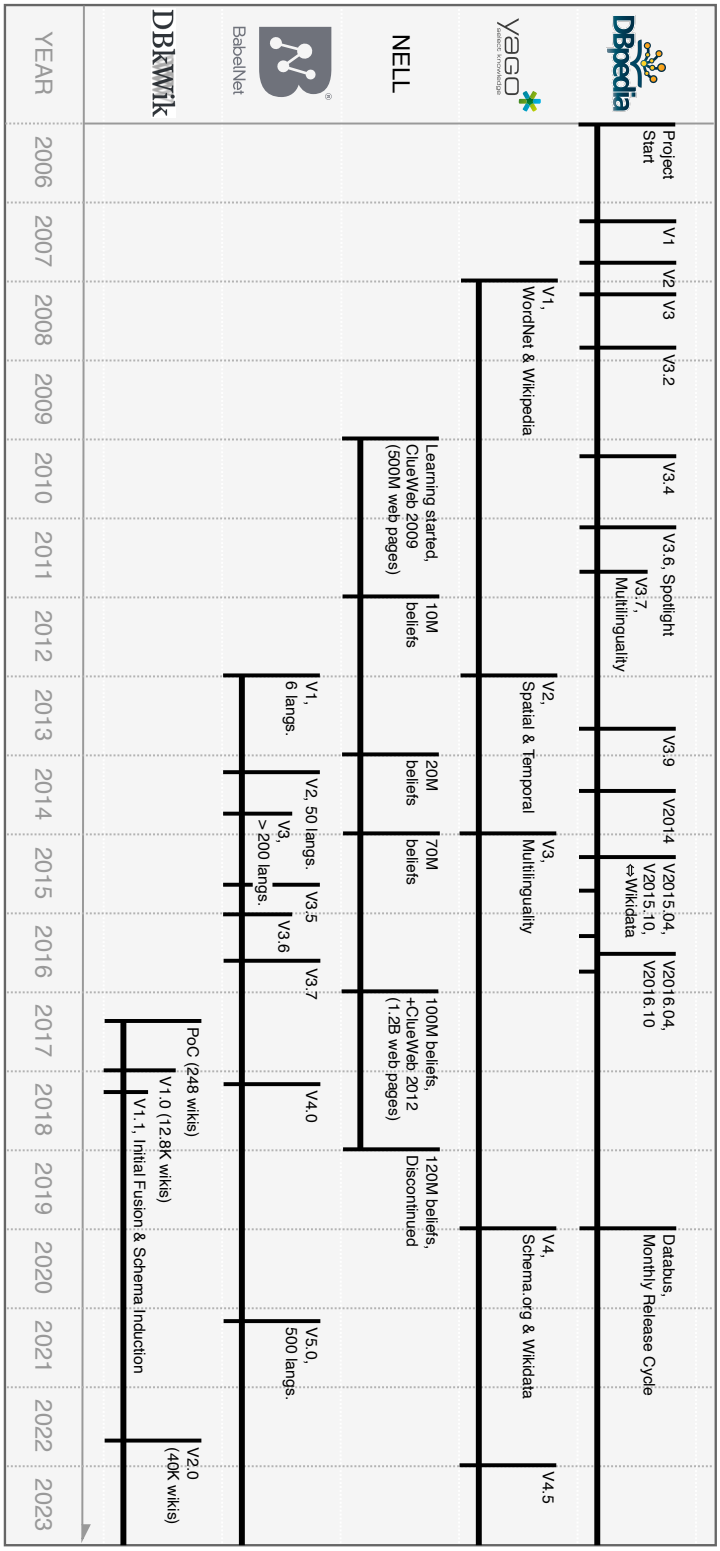


Figure 4.2: A timeline with major milestones of popular public KGs.

4.2.2 YAGO

OC Up to version 3 [120], YAGO automatically combines WordNet [128] with the WCG to create a large ontology. They add axioms for some classes derived from the WCG using hand-crafted rules. In version 4 [152], they fundamentally change the KG by combining the ontology from Schema.org with the one from Wikidata to create a cleaned, "reason-able" version of Wikidata. They define manual mappings between Schema.org and Wikidata classes to create the combined ontology and add simple SHACL constraints to ensure data validity.

KGP Up to version 3, YAGO performs KGP similarly to DBpedia, using articles as entities and extracting assertions from infoboxes. Additionally, they define an enhancement process where additional entities may be added from any external sources or tools. In version 2 [81], temporal and geospatial data is integrated; in version 3 [120], multilingual data from other Wikipedia language chapters is added. In version 4, entities and assertions are taken from Wikidata.

4.2.3 NELL

OC NELL started with an initial ontology defining hundreds of concepts and binary relations. During runtime, the ontology is extended with additional concepts and relations. Extraction in NELL is based on two components: *Learning Tasks* and *Coupling Constraints* [131]. The former are functions that process inputs like text and images to generate an output (e.g., an embedding of a concept). The latter couples multiple Learning Tasks to introduce logical constraints (e.g., entities of a subclass are also entities of a superclass).

KGP NELL is bootstrapped with a dozen examples for each concept and relation. New entities and assertions are added with each iteration through outputs from the Learning Tasks. The content of the KG is refined through iterative application of these tasks and the feedback from Coupling Constraints and occasional user input.

4.2.4 BabelNet

OC The ontology consists of concepts derived from senses in WordNet as well as articles and categories in Wikipedia [47]. They connect the two resources by mapping senses to articles automatically. In early versions, only lexical properties are used. In the recent version, they integrate related KGs like Wikidata and YAGO, taking over their semantic properties as well.

KGP Initially, the graph was populated with entities from Wikipedia articles. From WordNet, lexical and semantic pointers between synsets are extracted as relations. Relations between Wikipedia articles were initially extracted as unlabeled relations. In the recent version, efforts have been made to extract the semantics of the relations. Further, assertions from related KGs like Wikidata and YAGO are included [137].

4.2.5 DBkWik

OC DBkWik uses a variation of the DBpedia extraction framework to extract data from Wikis. Contrary to DBpedia, DBkWik has no community-defined mappings. Instead, they generate a shallow schema from the infoboxes of each Wiki and fuse these schemas afterwards. Then, they enrich the unified schema with subclass relations and restrictions for domains and ranges [77].

KGP Entities are derived from articles in the Wikis, and assertions are derived from infoboxes. Similar to the schema, entities must also be matched to avoid duplicates from overlapping Wikis [78].

4.3 Limitations and Challenges

In Table 4.1, we list the advantages and limitations of the previously discussed KGs. Following, we distil these into a (incomplete) list of challenges (mostly complementary to challenges mentioned by Weikum [201]):
(C1) Ontology with expressive, fine-grained types. An expressive and detailed ontology is a prerequisite for comprehensive data modelling. While some KGs have very detailed taxonomies already, there is a lack of axioms that explicitly describe the intent of their types.
(C2) Coverage of long-tail entities. Identifying and disambiguating long-tail entities is difficult as the data source contains, by definition, only limited information about them; most KGs choose to use a fixed set of entities like the set of Wikipedia articles.
(C3) Maintain high data quality. The quality of automatically constructed KGs will never be perfect, but KGs should have a certain quality to be useful in downstream tasks (typically, a correctness of 95% is desired [201]). Apart from the methods used, quality depends on the data sources targeted during extraction.
(C4) Coverage of unknown properties. Many KGs use a fixed schema with pre-defined properties to model knowledge. Extending this schema is a challenging task; potential errors greatly impact the KG quality.

Without a doubt, there are many more challenges to address. Weikum [201], for example, mentions additional challenges like the support of analyt-

KG	Advantages	Limitations
DBpedia	The ontology is hand-curated and, hence, of high quality. Entities and assertions are extracted from highly structured data and are of high quality as well. Due to its pioneering role of representing Wikipedia and its good accessibility, DBpedia serves as a central hub of the linked data web (cf. Section 2.1.2).	The manually-defined schema has limited expressiveness and flexibility. DBpedia is biased towards popular entities as Wikipedia allows articles only if the subject is of certain notability (cf. Section 2.3.3). Further, the information in the KG is limited to the content of the infoboxes in Wikipedia.
YAGO	Both YAGO3 and YAGO4 have very expressive and fine-grained ontologies. Due to many external sources and the connection to Wikidata, YAGO has a high density of assertions per entity.	YAGO3 suffers from the same problem of representing tail entities as DBpedia. In YAGO4, many more entities are ingested through the switch to Wikidata, which again introduces the limitation of manual curation. What happens to the manually defined mappings if Schema.org and/or the Wikidata taxonomy change is unclear.
NELL	The coverage of schema, entities, and assertions in NELL is limited only by the available information in the data source, which consists of a Web crawl.	The quality is comparably low as NELL initially starts with very little knowledge and uses only web resources with occasional human feedback during knowledge acquisition. In a link prediction evaluation, NELL scored a MAP of 0.35 in 2010 and 0.55 in 2017 [131]. Refinement iterations are run on a fixed Web crawl, i.e., recent knowledge is not considered.
BabelNet	BabelNet emphasizes the lexicographic perspective, describing entities and the senses of the words that entities are referenced with.	While a large variety of resources is included through the exploitation of mappings to other KGs, long-tail entities and new properties are not explicitly addressed.
DBkWik	DBkWik taps into additional data sources by targeting thousands of Wikis from a Wikifarm and can integrate specialized knowledge from many domains.	Creating a comprehensive schema from thousands of Wikis is difficult, especially when little information about the individual concepts and entities is available. DBkWik is restricted to entities defined in the ingested Wikis.

Table 4.1: Advantages and Limitations of public general-purpose KGs.

ical tasks by focusing on adding quantitative facts for entities or the insertion of common-sense knowledge into KGs.

There is a clear trade-off between challenges (C1) and (C2) on the one side and challenge (C3) on the other side. The more expressive and fine-grained an ontology, and the more entities covered, the likelier errors are introduced into the graph. This is especially true when the information used to extract concepts and entities is vague (as is the case for very specific concepts and long-tail entities). By relying on a combination of Wikipedia and Wikidata as data sources, YAGO and BabelNet find a good balance of all three challenges with their most recent versions.

NELL and DBkWik, on the other hand, use data sources where information is more difficult to extract. Naturally, the average quality is lower, but the KGs may contain valuable information that goes beyond what is contained in structured elements of Wikipedia and Wikidata. NELL is also the only KG equipped with an approach that tackles challenge (C4) to some extent, as additional properties can be derived from the data source continuously.

Part II

Ontology Construction

CHAPTER 5

Deriving a Fine-Grained Ontology from Wikipedia Categories and Lists

The WCG is a large network of categories structuring the articles in Wikipedia. Hence, it can serve as an excellent foundation for a general-purpose taxonomy. This chapter presents an approach that combines the WCG with Wikipedia list pages to form a large-scale taxonomy. The taxonomy has the advantage of being deeply interlinked with categories, list pages and the DBpedia ontology. These links bear the potential to enrich the taxonomy with additional information derived from these sources at a later stage. To demonstrate the potential of this taxonomy, we show in this chapter how it can be used to derive a large number of new entities and assertions from Wikipedia. The contributions of this chapter are:

- an approach for constructing a combined taxonomy of Wikipedia categories, lists, and DBpedia types,
- a distantly supervised ML approach for extracting entities from Wikipedia list pages, and
- an initial version of CaLiGraph with 800K classes and, beyond DBpedia, 700K entities, 7.5M type assertions, and 3.8M relation assertions.

The work presented in this chapter is based on the following publication:

Nicolas Heist and Heiko Paulheim. Entity Extraction from Wikipedia List Pages. In *The Semantic Web - ESWC 2020. Lecture Notes in Computer Science*, vol. 12123, pp. 327-342, Virtual Event, May 2020, Springer, Cham. [68]

List of Japanese speculative fiction writers

From Wikipedia, the free encyclopedia

This is a **list of Japanese speculative fiction writers**. Writers are sorted alphabetically by surname.

A [edit]

- Kimifusa Abe (安部公房, real name of Kōbō Abe)
- Kōbō Abe (安部公房, 1924–1993)
- Hirotaka Adachi (安達寛高, real name of Otsuichi)
- Jirō Akagawa (赤川次郎, b. 1948)
- Mizuhito Akiyama (秋山瑞人, b. 1971)
- Motoko Arai (新井素子, b. 1960)
- Mizuhito Akiyama (秋山瑞人, b. 1971)
- Kunio Aramaki (荒巻邦夫, real name of Yoshio Aramaki)
- Yoshimasa Aramaki (荒巻義雅, real name of Yoshio Aramaki)
- Yoshio Aramaki (荒巻義雄, b. 1933)
- Hiroshi Aramata (荒俣宏, b. 1947)
- Alice Arisugawa or Arisu Arisugawa (有栖川有栖, b. 1959)
- Taku Ashibe (芦辺拓, b. 1958)
- Yukito Ayatsuji (綾辻行人, b. 1960)

B–D [edit]

- Chen Shunchen (陳舜臣, pseudonym of Chin Shunshin)
- Chen Soon Shin (陳舜臣, pseudonym of Chin Shunshin)
- Kimio Chiba (千葉喜美雄, real name of Ryu Mitsuse)
- Chihitsudō (遲筆堂, pseudonym of Hisashi Inoue)
- Chin Shunshin (陳舜臣, 1924–2015)
- Gakuto Coda, Gakuto Kōda (甲田学人, b. 1977)

Figure 5.1: Excerpt of the Wikipedia page *List of Japanese speculative fiction writers* displaying the subjects in an *enumeration* layout.

5.1 Motivation

While Wikipedia's infoboxes and categories have been the subject of many information extraction efforts of KGs, list pages have received very little attention despite their apparent wealth of information. For entities of the page *List of Japanese speculative fiction writers* (shown in Fig. 5.1), we can derive several bits of information: *(type, Writer)*, *(nationality, Japan)*, and *(genre, Speculative Fiction)*. To include this information into a KG, we must first construct a taxonomy containing the concepts we want to describe.

In contrast to finding entities of a category, finding such entities among all the entities mentioned on a list page is a non-trivial problem. We will refer to these entities, being instances of the concept expressed by the list page, as *Subject Entities (SEs)*. Unlike categories, list pages are an informal construct in Wikipedia. Hence, the identification of their SEs brings up several challenges: While list pages are usually formatted as enumerations or tables, they have no convention of how the information in them is structured. For example, SEs can be listed somewhere in the middle of a table (instead of in the first column), and enumerations can have multiple levels. Furthermore, context information may not be available (it is difficult to find *Japanese speculative fiction writers* in a list if one doesn't know to look for *writers*).

The rest of this chapter is structured as follows. Section 5.2 discusses exist-

ing approaches concerned with extracting knowledge from (semi-)structured elements in Wikipedia. Section 5.3 introduces the idea of entity extraction from list pages, followed by a description of our approach in Section 5.4. In Section 5.5, we discuss results and present an empirical evaluation of our approach.

5.2 Related Work

The extraction of knowledge from structured elements in Wikipedia is mostly focused on two fields: Firstly, the field of taxonomy induction, where most of the approaches use the WCG to derive a taxonomy, and secondly, the application of IE methods to derive facts from various (semi-)structured sources like infoboxes, tables, lists, or abstracts of Wikipedia pages.

The approach of Ponzetto and Navigli [153] was one of the first to derive a large taxonomy from Wikipedia categories by focusing on a category's lexical head. They exploit the fact that almost exclusively categories with plural lexical heads are useful taxonomy elements. Hence, they can clean the WCG from non-taxonomic categories and relationships. Several other approaches create a combined taxonomy of the WCG and additional resources like WordNet (YAGO [120]) or Wikipedia pages (WiBi [47]).

As mentioned in Section 2.2.3, the distant supervision paradigm [130] is used extensively for IE in Wikipedia as it provides an easy way to automatically gather large amounts of training data with a low error rate. In the original work, Mintz et al. use Freebase as background knowledge to extract information from Wikipedia. Aprosio et al. [7] extend this approach using DBpedia as background knowledge.

Regarding list pages, Paulheim and Ponzetto [150] frame their general potential as a source of knowledge in Wikipedia. They propose to use a combination of statistical and NLP methods to extract knowledge and show that, by applying them to a single list page, they can extract a thousand new statements. Kuhn et al. [100] infer types for entities on list pages and are thus most closely related to our approach. To identify SEs of the list pages, they rely on information from DBpedia (e.g., how many relations exist between entities on the list page). Consequently, they can only infer new types for existing DBpedia entities. They use a score inspired by TF-IDF to find the type of a list page and can extract 303,934 types from 2,000 list pages with an estimated precision of 86.19%.

5.3 Categories and List Pages in Wikipedia

As described in Section 2.3.3, the WCG has a subgraph consisting of list categories organizing many of the list pages in Wikipedia. The *List of Japanese*

speculative fiction writers from Fig. 5.1, for example, is a member of the list category *Lists of Japanese writers*, which in turn has the parent *Lists of writers by nationality*, and so on.

As this subgraph is part of the WCG, we can use the list categories as a natural extension of a taxonomy induced by the WCG (e.g., by linking *Lists of Japanese writers* to the respective category *Japanese writers*). This comes with the benefit of including list pages into the taxonomy (i.e., we can infer that *List of Japanese speculative fiction writers* is a subconcept of the category *Japanese writers*).

In each list page, some links point to entities in the category the list page reflects, and others do not. In the list page *List of Japanese speculative fiction writers*, for example, some links point to pages about such writers (i.e., to its SEs), while others point to specific works by those writers. To distinguish those two cases, the unifying taxonomy is of immense value. Through the hierarchical relationships between categories and list pages, we can infer that if an entity is mentioned in both a list page *and* a related category, it is very likely a SE of the list page. Consequently, if an entity is mentioned in the list page *List of Japanese speculative fiction writers* and is contained in the category *Japanese writers*, it is almost certainly a Japanese speculative fiction writer.

In the remainder of this section, we provide additional background information and statistics for the resources used in our approach. The analyses and experiments in this chapter are based on *Wikipedia2016* (cf. Section 2.3.4). Further, we use the compatible DBpedia version from October 2016, which we employ as background KG $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{E}, \mathcal{L}, \mathcal{A})$.

The Wikipedia Category Graph

In the version from 2016, the WCG consists of 1,475,015 categories. Following Ponzetto and Navigli [154], we use only transitive subcategories of the category *Main topic classifications* while also getting rid of categories having one of the following keywords in their name: *wikipedia, lists, template, stub*.

The resulting filtered set of categories $\mathcal{C}^F$ contains 1,091,405 categories connected by 2,635,718 subcategory edges. We denote the set of entities in a category $c \in \mathcal{C}^F$ with $\mathcal{E}_c \subseteq \mathcal{E}$.[1]

The Wikipedia List Graph

The set of list categories $\mathcal{C}^L$ consists of 7,297 categories (e.g., *Lists of People*), connected by 10,245 subcategory edges (e.g., *Lists of Celebrities* being a subcategory of *Lists of People*). The set of list pages $\mathcal{L}$ contains 94,562 pages. Out of those, 75,690 are contained in at least one category in $\mathcal{C}^F$ (e.g., *List of*

[1] All articles in Wikipedia are DBpedia entities; hence, all articles in categories must be contained in DBpedia.

List of Cuban-American writers

From Wikipedia, the free encyclopedia
(Redirected from List of Cuban American writers)

This is a list of the most notable Cuban-American writers.

Name	Year of birth/death	Portrait	Notes
Alex Abella	1950–		Mystery/crime novelist, non-fiction writer, and journalist
Iván Acosta			Playwright; works include *El Super* (movie version 1979) and *Un cubiche en la luna* (1989)[1]
Mercedes de Acosta	1893–1968		
Robert Arellano	1969–		Novelist; works include *Havana Lunar* (2010 Edgar Award finalist) and *Havana Libre* (2017).
Reinaldo Arenas[2]	1943–1990		
René Ariza	1940–1994		
Octavio Armand [es]	1946–		Poet[1]
Joaquín Badajoz	1972–		Poet, author, essayist (North American Academy of the Spanish Language, fellow member)
Jesús J. Barquet[3]			
José Barreiro[4]	1948–		
Ruth Behar[5]	1956–		

Figure 5.2: Excerpt of the Wikipedia page *List of Cuban-American writers* displaying the subjects in a table layout.

Internet Pioneers is contained in the category *History of the Internet*), 70,099 are contained in at least one category in $\mathcal{C}^L$ (e.g., *List of Internet Pioneers* is contained in the category *Lists of Computer Scientists*), and 90,430 are contained in at least one of the two.[2]

The Anatomy of List Pages.

List pages can be categorised into one of three possible layout types [100]: 44,288 pages list entities in a bullet point-like *enumeration*. The list page *List of Japanese speculative fiction writers* in Fig. 5.1 lists the SEs in an enumeration layout. In this case, the SEs are mostly mentioned at the beginning of an enumeration entry. However, as some exceptions on the page show, this is not always the case.

46,160 pages list entities in a table layout. An example of this layout is given in Fig. 5.2, where an excerpt of the page *List of Cuban-American writers* is shown. The respective subjects of the rows are listed in the first column, but this can also vary between list pages.

The remaining 4,114 pages do not have a consistent layout. They are thus categorised as *undefined*.[3] As our approach significantly relies on the structured nature of a list page, we exclude list pages with an undefined layout from our extraction pipeline.

For a list page l, we define the task of identifying its SEs $\mathcal{E}_l$ among all the

[2]Note that $\mathcal{C}^F$ and $\mathcal{C}^L$ are disjoint as we exclude categories with the word *lists* in $\mathcal{C}^F$.

[3]We heuristically label a list page as having one of the three layout types by looking for the most frequent elements: enumeration entries, table rows, or none of them.

mentioned entities $\widehat{\mathcal{E}_l}$ in l as a binary classification problem.[4] A mentioned entity is classified as a SE of l or not. If not, it is usually mentioned in the context of an entity in $\mathcal{E}_l$ or for organisational purposes (e.g., in a *See also* section). In Figs. 5.1 and 5.2, mentioned entities are marked in blue (indicating that they have their own Wikipedia page and are thus contained in DBpedia) and in red (indicating that they do not have a Wikipedia page and are consequently no entities in DBpedia). The case of entities that are not tagged as such in Wikipedia (e.g., *Jesús J. Barquet* in the first column of Fig. 5.2) is not considered here as it introduces additional complexity to the task. Of the three types of possible entities, the latter two are the most interesting as they would add the most information to DBpedia. But it is also beneficial to identify entities already contained in DBpedia because we can derive additional information about them through the list page they are mentioned in.

Note that for both layout types, *enumeration* and *table*, we find at most one SE per enumeration entry or table row. We inspected a subset of $\mathcal{L}$ and found this pattern to occur in every one of them.

Learning Category Axioms with Cat2Ax

The approach presented in this chapter uses axioms over categories to derive a taxonomy from the WCG. Cat2Ax [67] is an approach that derives two kinds of axioms from Wikipedia categories: type axioms (e.g., for the category *Japanese writers* it learns that all entities in this category are of the type *Writer*), and relation axioms (e.g., for the same category it learns that all entities have the relation *(nationality, Japan)*). The approach uses statistical and linguistic signals to derive the axioms and achieves a correctness of 96% for the derived axioms. Chapter 6 describes this approach in detail.

5.4 Distantly Supervised Entity Extraction from List Pages

The processing pipeline for constructing a taxonomy from the WCG and retrieving SEs from list pages in $\mathcal{L}$ is summarized in Fig. 5.3. The pipeline consists of two main components: During *Training Data Generation*, we create a unified taxonomy of categories, lists, and DBpedia types. With distant supervision, we induce positive and negative labels from the taxonomy for some of the mentioned list page entities.

The resulting training data is passed to the *Entity Classification* component. We enrich it with features extracted from the list pages and learn classification models to identify the SEs.

[4] Neither $\mathcal{E}_l$ nor $\widehat{\mathcal{E}_l}$ are necessarily a subset of $\mathcal{E}$ as some entities in l may have no corresponding article in Wikipedia.

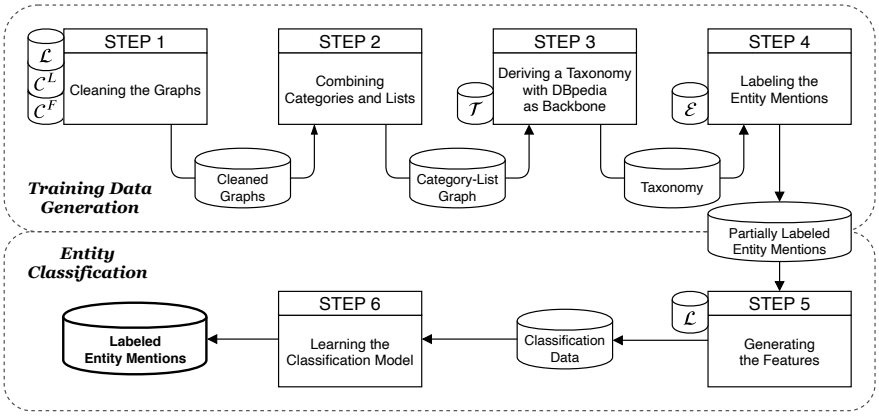


Figure 5.3: Overview of the pipeline for constructing a taxonomy from the WCG and retrieving SEs from list pages. Small cylindrical shapes next to a step indicate the use of external data, and large cylindrical shapes contain data passed between pipeline steps.

5.4.1 Training Data Generation

Step 1: Cleaning the Graphs

The initial category graph ($\mathcal{C}^F$ as nodes, subcategory relations as edges) and the initial list graph ($\mathcal{C}^L$ and $\mathcal{L}$ as nodes, subcategory relations and category membership as edges) both contain nodes and edges that have to be removed to convert them into valid taxonomies. Potential problems are shown in an abstract form in Fig. 5.4 and on an example in Fig. 5.5. In particular, we have to remove nodes that do not represent proper taxonomic types (e.g., *London* in Fig. 5.5). Additionally, we have to remove edges that either do not express a valid subtype relation (e.g., the edge from *Songs* to *Song awards* in Fig. 5.5) or create cycles (e.g., the self-references in Fig. 5.4).

To remove non-taxonomic nodes, we rely on the observation made by Ponzetto and Navigli [153] that a Wikipedia category is a valid type in a taxonomy if its head noun is in the plural. Consequently, we identify the head nouns of the nodes in the graph and remove all nodes with singular head nouns.[5]

To remove invalid edges, we first apply a domain-specific heuristic to eliminate non-taxonomic edges and subsequently apply a graph-based heuristic that removes cycles in the graphs. An edge is removed if the head noun of the parent is not a synonym or a hypernym of the child's head noun [153]. In Fig. 5.5, the head nouns of nodes are underlined; for example, we remove the edge from *Songs* to *Song awards* as the word *songs* is neither a synonym nor a hypernym of *awards*.

[5]We use spaCy (`http://spacy.io`) for head noun tagging.

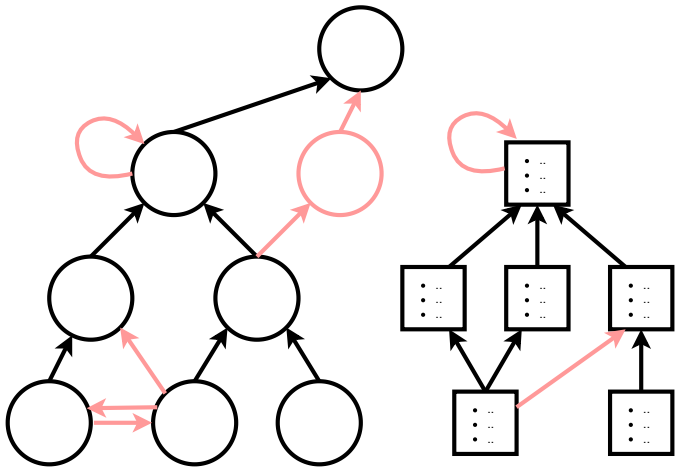


Figure 5.4: Possible invalid nodes and edges (marked in red) in the category graph (circles) and list graph (rectangles).

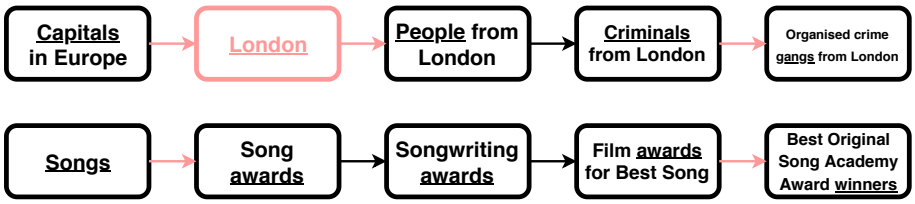


Figure 5.5: Examples of non-taxonomic nodes and edges (marked in red) that must be removed from the respective category graph or list graph.

We base our decision of synonym and hypernym relationships on a majority vote from three sources: (1) We parse the corpus of Wikipedia for Hearst patterns [63].[6] (2) We extract them from WebIsALOD [74], a large database of hypernyms crawled from the Web. (3) We extract them directly from categories in Wikipedia. To that end, we apply the Cat2Ax approach [67] to compute robust type and relation axioms for Wikipedia categories from linguistic and statistical signals. For every edge in the category graph, we extract a hypernym relationship between the head noun of the parent and the head noun of the child if we find matching axioms for both parent and child. For, if we find the axiom that every entity in the category *People from London* has the DBpedia type *Person* and we find the same axiom for *Criminals from London*, then we extract a hypernym relation between *People* and *Criminals*.

As a graph-based heuristic to resolve cycles, we detect edges that are part of a cycle and remove the ones pointing from a deeper node to a higher node

[6]Patterns that indicate a taxonomic relationship between two words like "X is a Y".

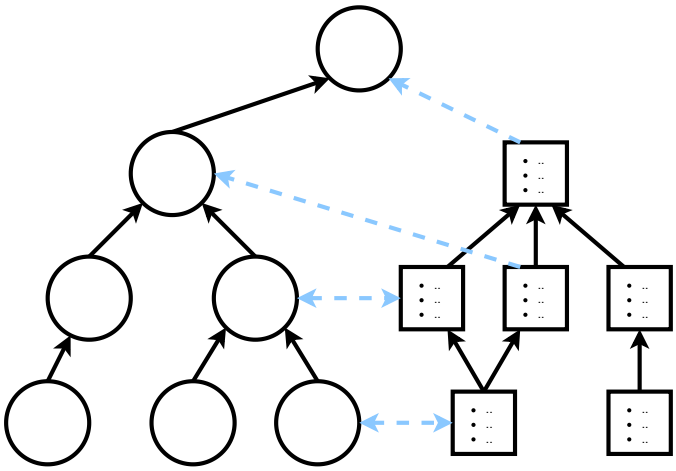


Figure 5.6: Possible connections between the category graph and the list graph

in the graph.[7] If cycles can not be resolved because edges point between nodes on the same depth level, those are removed as well.

Through the cleaning procedure, we reduce the size of the category graph from 1,091,405 nodes and 2,635,718 edges to 738,011 nodes and 1,324,894 edges. We reduce the size of the list graph from 77,396 nodes and 105,761 edges to 77,396 nodes and 95,985 edges.

Step 2: Combining Categories and Lists

For a combined taxonomy of categories and lists, we find links between them through linguistic similarity and existing connections in Wikipedia. As Fig. 5.6 shows, we find two types of links: equivalence and hypernymy. We identify the former by looking for category-list pairs that are either named similar (e.g., *Japanese writers* and *Lists of Japanese writers*) or are synonyms (e.g., *Media in Kuwait* and *Lists of Kuwaiti media*). With this method, we find 24,383 links.

We extract a hypernym link (similar to the method applied in Step 1) if the head noun of a category is a synonym or hypernym of a list's head noun. However, to avoid false positives, we limit the candidate links to existing edges in Wikipedia (i.e., the subcategory relation between a list category and a category or the membership relation between a list page and a category). With this method, we find 19,015 hypernym links. By integrating the extracted links into the two graphs, we create a category-list graph with 815,543 nodes (738,011 categories, 7,416 list categories, 70,116 list pages) and 1,463,423 edges.

[7]We define the depth of a node in the graph as the length of its shortest path to the root node *Main topic classifications*.

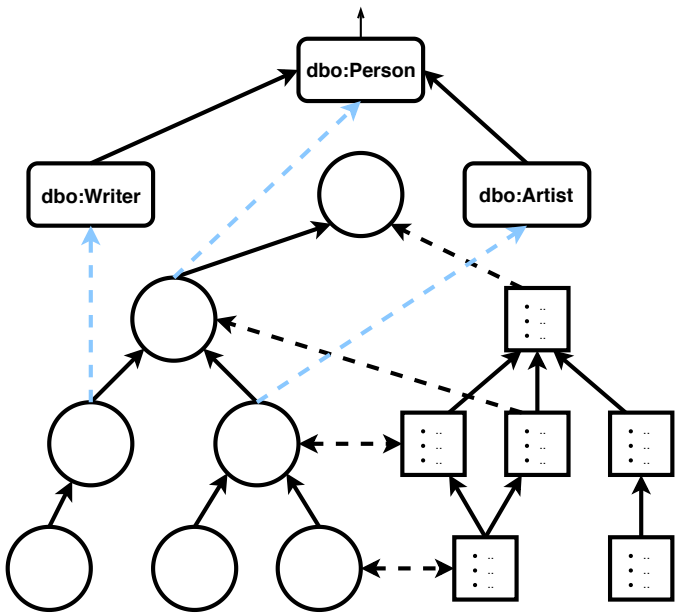


Figure 5.7: Extension of the category-list taxonomy with DBpedia mappings.

Step 3: Deriving a Taxonomy with DBpedia Mappings

As a final step, we map the category-list graph to the DBpedia taxonomy (as depicted in Fig. 5.7). To achieve that, we again apply the Cat2Ax approach to our current graph to produce type axioms for the graph nodes. For example, we discover the axiom that every entity in the category *Japanese writers* has the DBpedia type *Writer*. Thus, we use the type as a parent of *Japanese writers*. Taking the transitivity of the taxonomy into account, we find a DBpedia supertype for 88% of the graph's nodes.

Step 4: Labeling the Entity Mentions

We parse all entity mentions in list pages directly from the Wiki markup using DBpedia dumps and WikiTextParser[8] as markup parser.

We compute the training data for mentioned entities $\widehat{\mathcal{E}_l}$ of a list page l directly from the taxonomy. To that end, we define two mapping functions:

$$related : \mathcal{L} \rightarrow P(\mathcal{C}^F) \tag{5.1}$$

$$types : \mathcal{L} \rightarrow P(\mathcal{T}) \tag{5.2}$$

The function *related(l)* in Eq. (5.1) returns the subset of $\mathcal{C}^F$ that contains the taxonomically equivalent or most closely related categories for l. For example, *related(List of Japanese speculative fiction writers)* returns the category

[8] https://github.com/5j9/wikitextparser

Japanese writers and all its transitive subcategories (e.g., *Japanese women writers*). To find *related(l)* of a list page l, we traverse the taxonomy upwards starting from l until we find a category c that is contained in $\mathcal{C}^F$. We return c and all of its children.

With this mapping, we assign positive labels to entity mentions in l if they are contained in a category in *related(l)*:

$$\widehat{\mathcal{E}}_l^+ = \left\{ e | e \in \widehat{\mathcal{E}}_l \wedge \exists c \in related(l) : e \in \mathcal{E}_c \right\} \tag{5.3}$$

In the case of *List of Japanese speculative fiction writers*, $\widehat{\mathcal{E}}_l^+$ contains all entities that are mentioned on the list page *and* are members of the category *Japanese writers* or one of its subcategories.

The function *types(l)* from Eq. (5.2) returns the subset of the DBpedia types $\mathcal{T}$ that best describes entities in l. For example, *types(List of Japanese speculative fiction writers)* returns the DBpedia types *Agent*, *Person*, and *Writer*. To find *types(l)*, we retrieve all ancestors of l in the taxonomy and return those contained in $\mathcal{T}$.

With this mapping, we assign a negative label to an entity e mentioned in l if there are types in $\mathcal{T}_e$ that are disjoint with types in *types(l)*:

$$\widehat{\mathcal{E}}_l^- = \left\{ e | e \in \widehat{\mathcal{E}}_l \wedge \exists t_e \in \mathcal{T}_e, \exists t_l \in types(l) : disjoint(t_e, t_l) \right\} \tag{5.4}$$

To identify disjointnesses in Eq. (5.4), we use the disjointness axioms provided by DBpedia as well as additional ones that are computed by the methods described in Töpper et al. [189]. DBpedia declares, for example, the types *Person* and *Building* as disjoint, and the type *Person* is contained in *types(List of Japanese speculative fiction writers)*. Consequently, we label any mentions of buildings on the list page as negative examples.

In addition to the negative entity mentions that we retrieve via Eq. (5.4), we label entities as negative using the observation we have made in Section 5.3: As soon as we find a positive entity mention in an enumeration entry or table row, we label all the remaining entity mentions in that entry or row as negative.

For enumeration list pages, we find a total of 9.6M entity mentions. Of those, we label 1.4M as positive and 1.4M as negative. For table list pages, we find a total of 11M entity mentions. Of those, we label 850k as positive and 3M as negative.

5.4.2 Entity Classification

Step 5: Generating the Features

For a single data point (i.e., the mention of an entity in a specific list page), we generate a set of features that is shown in Table 5.1. Shared features are

	Type	Features
Shared	Page	#sections
	Positional	Position of section in LP
	Linguistic	Section title, POS/NE tag of entity and its direct context
Enum	Page	#entries, Avg. entry indentation level, Avg. entities/words/characters per entry, Avg. position of first entity
	Positional	Position of entry in enumeration, Indentation level of entry, #sub-entries of entry, Position of entity in entry
	Custom	#entities in current entry, #mentions of entity in same/other enumeration of LP
Table	Page	#tables, #rows, #columns, Avg. rows/columns per table, Avg. entities/words/characters per row/column, Avg. first column with entity
	Positional	Position of table in LP, Position of row/column in table, Position of entity in row
	Linguistic	Column header is synonym/hyponym of word in LP title
	Custom	#entities in current row, #mentions of current entity in same/other table of LP

Table 5.1: Features of the ML model grouped by list page type and feature type. Page features are computed for the complete list page (LP) and do not vary between entity mentions. We include standard deviations as features in addition to averages for page features.

created for entity mentions of enumeration and table list pages.

Features of the type *Page* encode the list page's characteristics and are similar for all entity mentions of the particular page. Features of the type *Positional, Linguistic, Custom* describe the characteristics of a single entity mention and its immediate context.

Step 6: Learning the Classification Model

To find a suitable classification model, we conduct an initial experiment on six classifiers (shown in Table 5.2) and compare them with the apparent baseline of always picking the first entity mention in an enumeration entry or table row. We compute the performance using 10-fold cross-validation while ensuring all entity mentions of a list page are in the same fold. In each fold, we use 80% of the data for training and 20% for validation. We report all the classifiers' performances after tuning their most important parameters

	Enum			Table		
Algorithm	P	R	F1	P	R	F1
Baseline (pick first entity)	74	96	84	64	53	58
Naive Bayes	80	90	84	34	**91**	50
Decision Tree	82	78	80	67	66	67
Random Forest	85	**90**	**87**	85	71	**77**
XG-Boost	**90**	83	86	**90**	53	67
Neural Network (MLP)	86	84	85	78	72	75
SVM	86	60	71	73	33	45

Table 5.2: Precision (P), Recall (R), and F-measure (F1) in per cent for the positive class (i.e. true SEs in a list page) of various classification models.

with a coarse-grained grid search.

Table 5.2 shows that all applied approaches outperform the baseline regarding precision. The XG-Boost algorithm scores the highest in terms of precision while maintaining rather high levels of recall. Since we want to identify entities in list pages with the highest possible precision, we use the XG-Boost model. After fine-grained parameter tuning, we train models with a precision of 91% and 90%, and a recall of 82% and 55% for enumeration and table list pages, respectively.[9] Here, we split the dataset into 60% training, 20% validation, and 20% test data.

5.5 Results and Discussion

5.5.1 List Page Extraction

Entities

We extracted 1,549,893 SEs that exist in DBpedia already. On average, an entity is extracted from 1.86 different list pages. Furthermore, we extracted 754,115 SEs that are new to DBpedia (from 1.07 list pages on average). Based on the list pages they have been extracted from, we assign them DBpedia types (i.e., the supertypes of the list page in the derived taxonomy). Fig. 5.8 shows the distribution of new entities over various high-level types.

Entity Types

Overall, we generated 7.5M new type assertions for DBpedia: 4.9M for entities in DBpedia (we assign a type to 2M previously untyped entities) and 2.6M for new entities (we find an average of 3.5 types per new entity). This is an increase of 51.15% in DBpedia's total type statements. We especially

[9]The models are trained using the scikit-learn library: `https://scikit-learn.org/`.

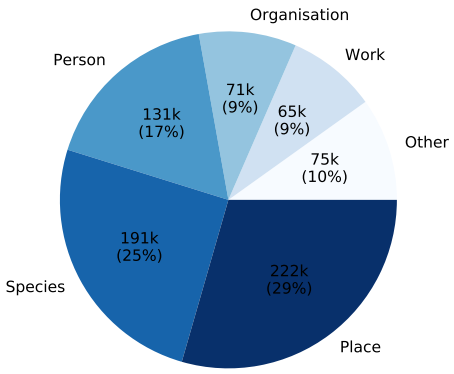


Figure 5.8: Distribution of entities that are new to DBpedia based on high-level types.

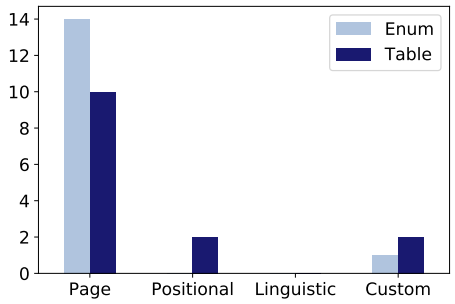


Figure 5.9: The 15 most important features used by XG-Boost grouped by respective feature type.

generate statements for rather specific types, i.e., types deep in the ontology.[10] Adding all the generated type statements to DBpedia, the average type depth would increase from 2.9 to 2.93. For new entities, we have an average type depth of 3.06. Fig. 5.10 shows the increase of type statements for the subtypes of the DBpedia type *Building*. We would increase the number of type statements by several orders of magnitude for almost all of them.

Entity Facts

Besides type assertions, we also infer relation assertions using the relation axioms that we generated via Cat2Ax. We generated 3.8M relation assertions: 3.3M for existing entities in DBpedia and 0.5M for new entities. For some previously unknown entities, we discovered large numbers of facts. For *Dan Stulbach*,[11] for example, we discover the type *Person* and information about his *birth place, occupation*, and *alma mater*.

5.5.2 Evaluation

We evaluate the correctness of both the constructed taxonomy and the inferred assertions.

[10] We define the depth of a type in DBpedia as the length of its shortest path to the root type *owl:Thing*.

[11] `http://caligraph.org/resource/Dan_Stulbach`

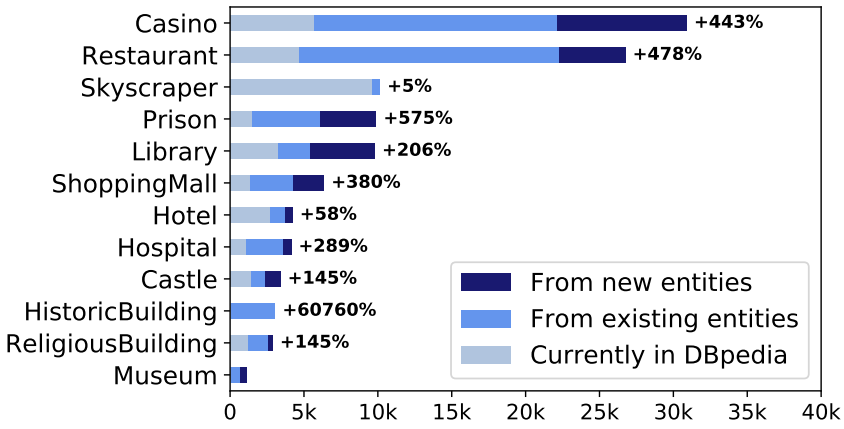


Figure 5.10: Comparison of the number of type statements that are currently in DBpedia with additional statements found by our approach for all subtypes of the DBpedia type *Building*.

Taxonomy

To validate the taxonomy, we conducted an evaluation using the crowdsourcing platform Amazon MTurk.[12] We randomly sampled 2,000 edges of the taxonomy graph and asked three annotators each whether the edge is taxonomically correct. The edges have been evaluated as correct in 96.25% (±0.86%) of the cases using majority vote (with an inter-annotator agreement of 0.66 according to Fleiss' kappa [48]).

Assertions

The correctness of the inferred type and relation assertions strongly depends on the Cat2Ax approach as we use its axioms to generate the assertions. Cat2Ax has a correctness of 96.8% for type axioms and 95.6% for relation axioms. For the resulting type and relation assertions (after applying the axioms to the entities of the categories), Cat2Ax achieves a correctness of 90.8% and 87.2%, respectively. However, the original Cat2Ax approach does not rely on a complete taxonomy of categories but computes axioms for individual categories without considering hierarchical relationships between them. In contrast, we include information about the subcategories of a given category while generating the axioms. We inspected 1,000 assertions[13] and find a correctness of 99% (±1.2%) for existing and 98% (±1.7%) for new type assertions and 95% (±2.7%) for existing and 97% (±2.1%) for new relation assertions.

[12] `https://mturk.com`

[13] We inspect 250 type and relation assertions for both existing and new entities.

Classification Models

With values of 91% and 90%, the precision of the classification models is significantly lower than the correctness of the extracted type and relation assertions. At first glance, this is a contradiction because, although the models extract entities and not assertions, a statement is incorrect if it has been created for the wrong entity. However, we must consider that the training data used to train and evaluate the models has been created using distant supervision. Hence, it will likely contain errors (e.g., due to wrong inheritance relations in the taxonomy). The fact that the final output of the processing pipeline has a higher correctness than the evaluation results of the models indicates that the models can learn meaningful patterns from the training data.

Fig. 5.9 shows the feature types of the 15 features with the highest influence on the decision of the final XG-Boost models. Almost all of them are features of the type *Page*, i.e., features that describe the general shape of the list page the entities are extracted from. Features of the other types describing the immediate context of an entity are used only very sparsely. This indicates that, to bridge the gap in recall between the classification models and the baseline, we have to develop models that can better use the structure of a list page. Accordingly, we see the most significant potential in an adapted ML approach that, instead of classifying every entity mention in isolation, uses a holistic perspective and identifies the set of mentions that fit the list page best, given its structure.

5.6 Conclusion

This chapter presented an approach for generating a large-scale taxonomy from the WCG, list pages, and Wikipedia. We demonstrated the potential of the taxonomy by extracting entities and assertions from Wikipedia that go beyond what is contained in DBpedia. The implemented approach is published as part of the CaLiGraph extraction framework.[14] The taxonomy and the extracted information are published as the first version of CaLiGraph.

In Chapters 7 and 8, we significantly extend the presented approach for entity extraction from list pages by expanding it to all listings in Wikipedia and considering unlinked mentions as well. However, the approaches presented there still rely on this chapter's taxonomy and training data generation approach.

[14] `https://github.com/nheist/CaLiGraph`

CHAPTER 6

Learning Defining Axioms for Wikipedia Categories

In the previous chapter, we have already mentioned the *Cat2Ax* approach as it plays an important role in the construction of a taxonomy from the WCG. This chapter formally introduces Cat2Ax, enriching Wikipedia-based KGs by explicating the semantics in category names. We combine the category graph structure, lexical patterns in category names, and instance information from a KG to learn patterns in category names and map these patterns to type and relation axioms. The approach's potential has already been demonstrated in the previous chapter, where the generated axioms were used to derive hypernyms and assertions. The contributions of this chapter are as follows:

- We introduce an approach that extracts axioms for Wikipedia categories using features derived from the instances in a category and their lexicalisations.
- We extract more than 700K axioms for explicating the semantics of category names at a precision of more than 95%. The axioms are integrated into the CaLiGraph ontology.
- Using those axioms, we generate more than 7.7M assertions unknown to DBpedia at a precision of more than 87%.

The work presented in this chapter is based on the following publication:

Nicolas Heist and Heiko Paulheim. Uncovering the Semantics of Wikipedia Categories. In *The Semantic Web – ISWC 2019. Lecture Notes in Computer Science*, vol. 11778, pp. 219–236, Auckland, New Zealand, October 2019, Springer, Cham. [67]

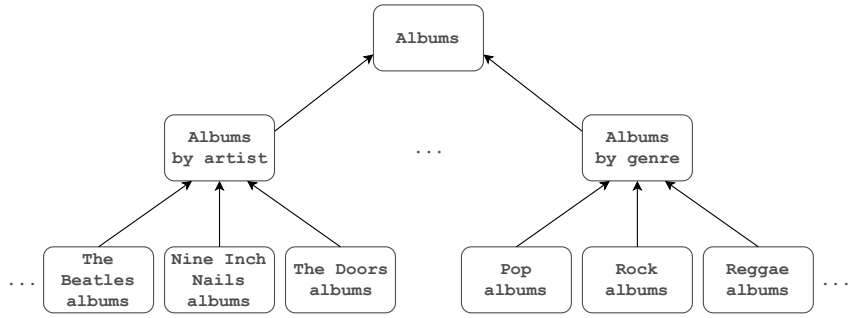


Figure 6.1: Excerpt of the WCG showing the category *Albums* and some of its subcategories.

6.1 Motivation

Missing knowledge in Wikipedia-based KGs can be attributed to absent information in Wikipedia, but also to the extraction procedures of KGs. DBpedia uses infobox mappings to extract assertions for individual instances but does not explicate any information implicitly encoded in categories. YAGO uses manually defined patterns to derive assertions for entities of matching categories. For example, they extract a person's birth year by exploiting categories ending with *births*. Consequently, all persons contained in the category *1879 births* are attributed with *1879* as *year of birth* [184]. Likewise, most existing works, like from Liu et al. [113] and Xu et al. [211], leverage textual patterns in the category names.

There are some limitations to such approaches since, in many cases, very specific patterns are necessary (e.g., *(county,Chester County)* for the category *Townships in Chester County, Pennsylvania*), or the information is only indirectly encoded in the category (e.g., *(timeZone,Eastern_Time_Zone)* for the same category). To capture as much knowledge as possible from categories, we propose an approach that learns patterns not only from the category names but exploits the underlying KG as well.

While category names are plain strings, we aim at uncovering the semantics of those category names. To that end, we want to extract both type and relation information from categories. In the example in Fig. 6.1, we would, e.g., learn type (Eq. (6.1)) as well as relation (Eqs. (6.2) and (6.3)) axioms.

$$\exists category.\{Reggae_albums\} \sqsubseteq Album \tag{6.1}$$

$$\exists category.\{Reggae_albums\} \sqsubseteq \exists genre.\{Reggae\} \tag{6.2}$$

$$\exists category.\{Nine_Inch_Nails_albums\} \sqsubseteq \exists artist.\{Nine_Inch_Nails\} \tag{6.3}$$

Once those axioms are defined, they can be used to fill in the missing type and relation assertions for all instances for which those categories have been assigned.

The rest of this chapter is structured as follows. Section 6.2 frames the approach described in this chapter in related works. Section 6.3 lays out the preliminaries of our work, followed by an introduction of our approach in Section 6.4. In Section 6.5, we discuss an empirical evaluation of our approach.

6.2 Related Work

With the wider adoption of general-purpose KGs such as DBpedia [104], YAGO [120], or Wikidata [195], their quality has come into the focus of recent research [42, 214]. The systematic analysis of KG quality has inspired a lot of research around an automatic or semi-automatic improvement or refinement [147].

Generally, KG refinements can be distinguished along various dimensions: the goal (filling missing knowledge or identifying erroneous axioms), the target (e.g., schema or instance level, type or relation assertions, etc.), and the knowledge used (using only the KG as such or also external sources of knowledge). The approach discussed in this chapter extracts axioms on the schema level and assertions on the instance level using Wikipedia categories as an external source of knowledge.

Two signals can be exploited to extract information from categories: (1) lexical information from the category's name and (2) statistical information of the instances belonging to the category. YAGO, as discussed above, uses the first signal. A similar approach is *Catriple* [113], which exploits manually defined textual patterns (such as *X by Y*) to identify parent categories which organize instances by objects of a given relation: for example, the category *Albums by genre* has child categories whose instances share the same object for the relation *genre*, and can thus be used to generate axioms such as the one in Eq. (6.2) above. The Catriple approach does not explicitly extract category axioms but finds 1.27M relation assertions. A similar approach is taken in Nastase et al. [136], utilizing POS tagging to extract patterns from category names without deriving any KG axioms.

In the area of taxonomy induction, many approaches make use of lexical information when extracting hierarchies of terms. Using Hearst patterns [63] is one of the most well-known methods to extract hypernymy relations from text. It has been extended multiple times, e.g., by [99] who enhance their precision by starting with pre-defined terms and post-filtering the final results. [193] use an optimal branching algorithm to induce a taxonomy from definitions and hypernym relations extracted from text.

The *C-DF* approach [211] is an approach of the second category, i.e., it

relies on statistical signals. It uses probabilistic methods on the category entities to identify an initial set of axioms. From that, it mines the extraction patterns for category names automatically. The authors find axioms for more than 60K categories and extract around 700K relation assertions and 200K type assertions.

Exploiting statistical information from category instances is a setting similar to ontology learning [164]. For example, approaches such as DL-Learner [103] find description logic patterns from a set of instances. These approaches are very productive when there is enough training data, and they provide exact results, especially when both positive and negative examples are given. Both conditions are not trivially fulfilled for the problem setting in this chapter: many categories are rather small (75% of categories have fewer than ten members), and, due to the OWA, negative examples for category membership are not given. Therefore, we postulate that both statistical and lexical information must be combined to derive high-quality axioms from categories.

With Catriple and C-DF, we compare our approach to the two closest approaches in the literature. While Catriple relies solely on lexical information in the category names, and C-DF relies solely on statistical information from the instances assigned to categories, we propose a *hybrid* approach combining the lexical and statistical signals. Moreover, despite exploiting category names, we do not use language-specific techniques, so our approach is language-agnostic.

6.3 Preliminaries

The Cat2Ax approach uses three kinds of sources: the WCG, background knowledge from a KG, and lexicalisations of entities and types in the KG. In this section, we provide relevant definitions and give background information about the respective sources.

Like in the previous chapter, the analyses and experiments are based on *Wikipedia2016* (cf. Section 2.3.4). We use the compatible DBpedia version from October 2016, which we employ as background KG $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{E}, \mathcal{L}, \mathcal{A})$. Further, we reuse the filtered set of categories $\mathcal{C}^F$ defined in Section 5.3 and denote the set of entities in a category $c \in \mathcal{C}^F$ with $\mathcal{E}_c \subseteq \mathcal{E}$.

Background Knowledge

To get an estimate of how likely the combination of a property p and a value v occurs within the entities of a category c, we calculate their frequencies using background knowledge from $\mathcal{K}$:

$$\mathit{freq}(c, p, v) = \frac{|\{e | e \in \mathcal{E}_c \wedge (e, p, v) \in \mathcal{A}\}|}{|\mathcal{E}_c|} \tag{6.4}$$

As shown in Example 6.3.1, we compute type frequencies if the predicate is *rdf:type*.

Example 6.3.1. Type frequency calculation

The category *The Beatles albums* has 24 entities, with 22 having the type *dbo:Album*, resulting in a type frequency *freq(*The Beatles albums, rdf:type, dbo:Album*)* of 0.92.[1]

For any other property, we compute relation frequencies:

Example 6.3.2. Relation frequency calculation

The Beatles albums has 24 entities, with 11 having *dbr:Rock_and_roll* as *dbo:genre*. This results in a relation frequency *freq(*The Beatles albums, dbo:genre, dbr:Rock_and_roll*)* of 0.46.

Resource and Type Lexicalisations

A lexicalisation is a word or phrase used in natural language text that refers to an entity or type in $\mathcal{K}$. For an entity *e*, *lex(e)* contains all its lexicalisations, and *lexCount(e,l)* is the count of how often a lexicalisation *l* has been found for *e*. When the count of a lexicalisation *l* is divided by the sum of all counts of lexicalisations for an entity *e*, we have an estimate of how likely *e* will be expressed by *l*.

We are, however, interested in the inverse problem: Given a lexicalisation *l*, we want the probability of it expressing an entity *e*. We define $lex^{-1}(l)$ as the set of entities having *l* as lexicalisation. The lexicalisation score, i.e., the probability of an entity *e* being expressed by the lexicalisation *l*, is then computed by the fraction of how often *l* expresses *e* compared to all other entities:

$$lexScore(e, l) := \frac{lexCount(e, l)}{\sum_{e' \in lex^{-1}(l)} lexCount(e', l)} \tag{6.5}$$

Example 6.3.3. Entity lexicalisation score calculation

We encounter the word *lennon* in Wikipedia and want to find out how likely it is that the word refers to the resource *dbr:John_Lennon*. We find 357 occurrences of the word for which we know the resource it refers to. 137 of them actually refer to *dbr:John_Lennon*, while others refer, e.g., to the soccer player *dbr:Aaron_Lennon* (54 times) or *dbr:Lennon,_Michigan* (14 times). We use the occurrence counts to compute a *lexScore(*dbr:John_Lennon, "lennon"*)* of 0.42.

[1]When referring to DBpedia in our examples and experiments, we use the prefix *dbr:* for entities and *dbo:* for properties and types.

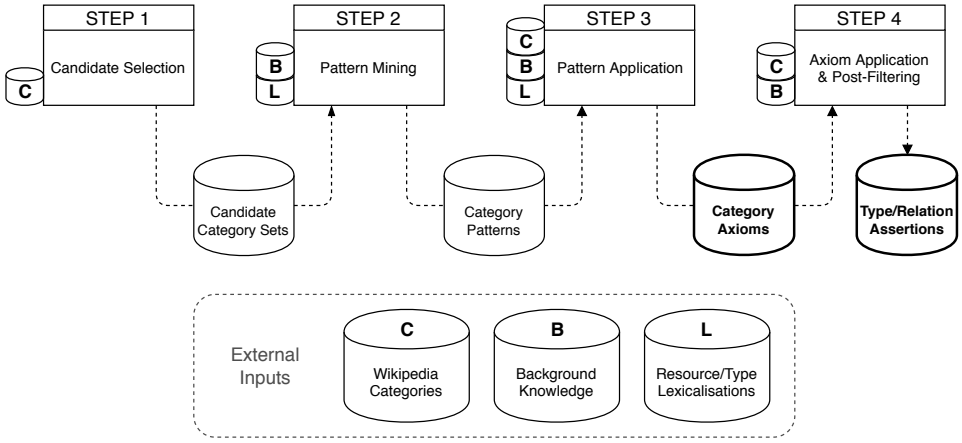


Figure 6.2: Overview of the Cat2Ax approach.

DBpedia already provides the lexicalisations of entities [23]. They are gathered from the anchor texts of links between Wikipedia articles. For types, however, there is no such data set provided.

To gather type lexicalisations from Wikipedia, we apply the following methodology: For every type $t \in \mathcal{T}$, we crawl the articles of all entities having type t and extract hypernymy relationships using Hearst patterns [63]. To ensure that we only extract relationships for the correct type, we exclusively use the ones with a lexicalisation of the article's entity as their subject. To increase the coverage of type lexicalisations, we intentionally count individual words instead of complete phrases of the extracted lexicalisation. To calculate the lexicalisation scores of a phrase, we sum up the counts of the phrase's words.

Example 6.3.4. Type lexicalisation score calculation

We are extracting type lexicalisations for the type *dbo:Band*. The entity *dbr:Nine_Inch_Nails* has the appropriate type. We extract hypernymy relationships in its article text. In the sentence *"Nine Inch Nails is an American industrial rock band [..]"*, we find the subject *Nine Inch Nails* and the object *American industrial rock band*. As the subject is in *lex(*dbr:Nine_Inch_Nails*)*, we accept the object as lexicalisation of *dbo:Band*. The lexicalisation count of the words *American, industrial, rock, band* is incremented, and, for each of those words encountered, the lexicalisation score for the class *dbo:Band* increases.

6.4 Approach

The overall approach of Cat2Ax is shown in Fig. 6.2. The external inputs have already been introduced in Section 6.3. The outputs of the approach (marked in bold font) are twofold: A set of axioms defining restrictions for entities in a category and a set of assertions which are novel facts about resources in the graph.

The approach has four steps: The *Candidate Selection* uses hierarchical relationships in the WCG to form sets of categories sharing a property that a textual pattern can describe.

In the *Pattern Mining* step, we identify patterns in the names of categories that are characteristic of a property or type. To achieve that, we use background knowledge about resources in the respective categories and lexicalisations. We consider a pattern only if it applies to a majority of the categories in a candidate category set.

In the *Pattern Application* step, we apply the extracted patterns to all categories to find category axioms. Here, we again rely on background knowledge and lexicalisations to decide whether a pattern applies to the category.

Finally, we generate assertions by applying the axioms of a category to its resources and subsequently use post-filtering to remove assertions that would create contradictions in the KG.

6.4.1 Candidate Selection

In this first step, we extract sets of categories with names that indicate a shared relation or type. We base the extraction of such candidate category sets on two observations:

The first one is inspired by the Catriple approach [113]. They observed that parents often organize their children according to a certain property in a parent-child relationship of categories. Contrary to Catriple, we do not use the parent category to identify this property, but we use the complete set of children to find their similarities and differences.

As we now know from the first observation, the children of a category can have certain similarities (which are the reason that they have the same parent category) and differences (which are the reason that the parent was split up into child categories). As a second observation, we discovered that when a certain property organizes the children of a category, their names have a shared part (i.e., a common prefix and/or postfix) and a part that differs for each category. We found that the shared part is often an indicator of the type of resources contained in the category, while the differing part describes the value of the property by which the categories are organized.

Using these observations, we produce the candidate category sets by looking at the children of each Wikipedia category and forming groups out

of children that share a prefix and/or postfix.

Example 6.4.1. Album candidate sets

In Fig. 6.1, we see parts of two candidate category sets with the postfix *albums*. The first one contains 143 children of the category *Albums by artist*. The second one contains 45 children of the category *Albums by genre*.

Note that we sometimes form multiple candidate category sets from the category's children as there might be more than one shared pre- or postfix.

Example 6.4.2. Reality TV candidate sets

The children of the category *Reality TV participants* yield three candidate sets ending on *participants*, *contestants*, and *members*.

6.4.2 Pattern Mining

We want to discover each candidate category set's characteristic property and type. Therefore, we identify patterns to be used in the subsequent steps to extract category axioms. Each pattern consists of a textual indicator (i.e., the shared part in the names of categories) and the implication (i.e., the shared property or type).

To determine the characteristic property, we inspect every category in the candidate set and compute a score for every possible relation in the category. As mentioned in Section 6.4.1, the value of a relation differs for the categories in a set. We thus focus on finding the property with the highest score and disregard relation values. To that end, we aggregate the scores from all categories and choose the property that performs best over the complete category set. We learn a pattern for the property that covers the complete candidate category set.

The score of a relation *(p, v)* for a category *c* consists of two parts, with one being based on background knowledge and the other on lexical information. The latter uses the part c_{var} of a category's name that differs between categories in the set to estimate how likely c_{var} expresses the value of the relation. The score is computed as follows:

$$score_{rel}(c, p, v) = freq(c, p, v) * lexScore(v, c_{var}) \tag{6.6}$$

Note that *freq(c,p,v)* is only greater than zero for relations of the resources in *resources(c)*, which drastically reduces the number of scores that have to be computed.

Example 6.4.3. Relation score calculation

For the category c=*The Beatles albums*, we compute an individual relation

score for each property-value pair in $\mathcal{A}$ having an entity in $\mathcal{E}_c$ as their subject. To compute $score_{rel}$(c, dbo:artist, dbr:The_Beatles), we multiply the frequency *freq(*c, dbo:artist, dbr:The_Beatles*)* with the lexicalisation score *lexScore(*dbr:The_Beatles, "The Beatles"*)*.

As an aggregation function for the scores, we use the median. Heuristically, we found that the property with the highest median of scores is suited to be the characteristic property for a category set. To avoid learning incorrect patterns, we discard the property if it cannot be found in at least half of the categories in the set, i.e., if the median of scores is zero.

Example 6.4.4. Relation score aggregation

After computing all the relation scores for all categories in the category set formed by the 143 children of *Albums by artist*, we aggregate the computed scores by their property and find *dbo:artist* to have the highest median score.

The support of a pattern is the count of how often a pattern has been learned for a category. If we discover a valid property for a category set, the support of the respective property pattern is increased by the number of categories in the set. We assume hereby that if this property is characteristic for the majority of categories, then it is characteristic for all categories in the set.

For the extraction of characteristic types, we apply the same methodology except for the calculation of the score of a type. We compute the score of a type t in the category c using its frequency in c and a lexicalisation score derived from the shared part c_{fix} in a category's name:

$$score_{type}(c, t) := freq(c, \texttt{rdf:type}, t) * lexScore(t, c_{fix}) \tag{6.7}$$

Example 6.4.5. Property and type patterns

For the category sets formed by the children of *Albums by artist* and *Album by genre* in Fig. 6.1, we find the following property patterns to have the highest median scores:
PP_1 = "<lex(*dbr:res*)> albums" $\sqsubseteq$ ∃dbo:artist.{*dbr:res*}
PP_2 = "<lex(*dbr:res*)> albums" $\sqsubseteq$ ∃dbo:genre.{*dbr:res*}
We increase the support of PP_1 by 143 and PP_2 by 45. For both sets, we extract the type pattern
TP_1 = "<lex(*dbr:res*)> albums" $\sqsubseteq$ *dbo:Album*
and increase its support by 188 (respectively using the counts from Example 6.4.1).

6.4.3 Pattern Application

Before we apply the patterns to the categories to identify axioms, we need to define a measure for the confidence of a pattern. This is especially necessary because we can find multiple implications for the same textual pattern as shown in Example 6.4.5. We define the confidence *conf(P)* of a pattern *P* as the quotient of the support of *P* and the sum of supports of all the patterns matching the same textual pattern as *P*.

Example 6.4.6. Pattern confidence computation

Assuming PP_1 and PP_2 in Example 6.4.5 are the only property patterns we found, we have a pattern confidence of 0.76 for PP_1 and 0.24 for PP_2.

Next, we apply all our patterns to the categories and compute the axiom confidence by calculating the fit between the category and the pattern. Therefore, we reuse the scores from Eqs. (6.6) and (6.7) and combine them with the confidence of the pattern. As a relation pattern only specifies the property of the axiom, we compute the axiom confidence for every possible value of the axiom's property to have a ranking criterion. For a category *c*, a property pattern *PP* with property p_{PP} and a possible value *v*, we define the confidence as:

$$conf(c, PP, v) := conf(PP) * score_{rel}(c, p_{PP}, v). \tag{6.8}$$

And similarly, we define the confidence for a type pattern *TP* with type t_{TP} as:

$$conf(c, TP) := conf(TP) * score_{type}(c, t_{TP}). \tag{6.9}$$

Using the confidence scores, we can control the quality of extracted axioms by only accepting those with a confidence greater than a threshold τ. We inspect and evaluate the generated axioms in our experiments to find a reasonable threshold.

Example 6.4.7. Axiom confidence computation

Both patterns PP_1 and PP_2 from Example 6.4.5 match the category *Reggae albums*. Using PP_1, we can't find an axiom for the category as there is no evidence in $\mathcal{K}$ for the property *dbo:artist* together with any resources that have the lexicalisation *Reggae* (i.e., $score_{rel}$ is equal to 0). For PP_2, however, we find the axiom *(Reggae albums, dbo:genre, dbr:Reggae)* with a confidence of 0.18.

Multiple property or type patterns can have a confidence greater than τ for a single category. The most precise variant for property and type patterns is to accept only the pattern with the highest confidence and discard all the

others. However, we found that multiple patterns can imply valid axioms for a category and thus follow a more differentiated selection strategy.

For relation axioms, we accept multiple axioms as long as they have different properties. We accept only the axiom with higher confidence when the properties are equal.

Example 6.4.8. Accepting axioms with different properties

For the category *Manufacturing companies established in 1912* (short: c_1), we find the relation axioms (c_1, *dbo:foundingYear*, 1912) and (c_1, *dbo:industry*, *dbr:Manufacturing*). As they have different properties, we accept both.

Example 6.4.9. Discarding axioms with similar properties

For the category *People from Nynäshamn Municipality* (short: c_2), we find the relation axioms (c_2, *dbo:birthPlace*, *dbr:Nynäshamn_Municipality*) and (c_2, *dbo:birthPlace*, *dbr:Nynäshamn*). As they have the same property, we only accept the former as its confidence is higher.

For type axioms, we accept the axioms with the highest confidence and any axioms with lower confidence that imply sub-types of the already accepted types.

Example 6.4.10. Accepting transitive type axioms

For the category *Missouri State Bears baseball coaches* (short: c_3), we find the axioms (c_3, *rdf:type*, *dbo:Person*) and (c_3, *rdf:type*, *dbo:CollegeCoach*). Despite the latter having lower confidence than the former, we accept both because *dbo:CollegeCoach* is a sub-type of *dbo:Person*.

6.4.4 Axiom Application and Post-Filtering

With the category axioms from the previous step, we generate new assertions by applying the axiom to every category resource.

Example 6.4.11. Applying relation axioms

We apply the axiom (*Reggae albums,dbo:genre,dbr:Reggae*) to all resources of *Reggae albums* and generate 50 relation assertions, 13 of which are not yet present in $\mathcal{K}$.

Categories can contain special resources that do not belong to the category itself but, for example, describe the category's topic. For example, the category *Landforms of India* contains several actual landforms but also the resource *Landforms of India*. To avoid generating wrong assertions for such special resources, we filter all generated assertions using the existing knowledge in the knowledge base.

For relation assertions, we use the functionality of its property to filter invalid assertions. Accordingly, we remove a relation assertion *(s, p, o)* if the property *p* is functional[2] and there is an assertion *(s, p, o')* with $o \neq o'$ already in $\mathcal{A}$.

Example 6.4.12. Accepting assertions for non-functional properties

Out of the 13 new *dbo:genre* assertions generated for the category *Reggae albums* in the previous example, nine refer to resources which do not have a *dbo:genre* at all. Four add a genre to a resource with one or more values for *dbo:genre*. The latter is possible since *dbo:genre* is not functional.

Example 6.4.13. Discarding assertions for functional properties

The relation assertion (*dbr:Bryan_Fisher, dbo:birthYear*, 1982) is removed because the assertion (*dbr:Bryan_Fisher, dbo:birthYear*, 1980) is contained in $\mathcal{A}$, and *dbo:birthYear* is functional.

To identify invalid type assertions, we use disjointness axioms in $\mathcal{K}$ and remove any type assertion that, if added to $\mathcal{A}$, would lead to a disjointness violation.

Example 6.4.14. Discarding assertions for disjoint types

The assertion (*dbr:Air_de_Paris, rdf:type, dbo:Person*) is discarded because the subject already has the type *dbo:Place*, which is disjoint with *dbo:Person*.

6.5 Experiments

In this section, we first provide details about applying the Cat2Ax approach with DBpedia as background knowledge. Subsequently, we discuss the evaluation of Cat2Ax and compare it to related approaches. For the implementation of the approaches, we used the Python libraries spaCy[3] and nltk[4]. The code of Cat2Ax[5] and all data[6] of the experiments are openly available. The Cat2Ax approach is also integrated in the CaLiGraph extraction framework.[7]

[2] Since the DBpedia ontology does not define any functional object properties, we use a heuristic approach and treat all properties which are used with multiple objects on the same subject in less than 5% of the subjects as functional. This heuristic marks 710 out of 1,355 object properties as functional.

[3] `https://spacy.io/`

[4] `https://www.nltk.org/`

[5] `https://github.com/nheist/Cat2Ax`

[6] `http://data.dws.informatik.uni-mannheim.de/Cat2Ax`

[7] `https://github.com/nheist/CaLiGraph`

	Textual pattern	Implication	Sup.	Conf.
1	Films directed by <lex(*dbr:res*)>	$\sqsubseteq$ ∃dbo:director.{*dbr:res*}	7661	1.00
2	Films directed by <lex(*dbr:res*)>	$\sqsubseteq$ dbo:Film	7683	1.00
3	<lex(*dbr:res*)> albums	$\sqsubseteq$ ∃dbo:artist.{*dbr:res*} $\sqsubseteq$ ∃dbo:genre.{*dbr:res*} $\sqsubseteq$ ∃dbo:recordLabel.{*dbr:res*}	31426 552 411	0.97 0.02 0.01
4	<lex(*dbr:res*)> albums	$\sqsubseteq$ dbo:Album	33542	1.00
5	Populated places in <lex(*dbr:res*)> district	$\sqsubseteq$ ∃dbo:isPartOf.{*dbr:res*} $\sqsubseteq$ ∃dbo:district.{*dbr:res*}	269 51	0.84 0.16
6	Populated places in <lex(*dbr:res*)> district	$\sqsubseteq$ dbo:Settlement	362	1.0

Table 6.1: Examples of discovered textual patterns and possible implications

6.5.1 Axiom Extraction using DBpedia

Candidate Selection We find 176,785 candidate category sets with an average size of eight categories per set. From those sets, 60,092 have a shared prefix, 76,791 a shared postfix, and 39,902 a shared prefix and postfix.

Pattern Mining We generate patterns matching 54,465 different textual patterns. For 24,079 of them, we imply properties; for 54,096, we imply types. On average, a property pattern implies 1.22 different properties, while a type pattern implies 1.08 different types. Table 6.1 lists exemplary patterns that match a prefix (rows 1-2), a postfix (rows 3-4), and both a prefix and a postfix (rows 5-6).

Pattern Application We determine a threshold τ for the minimum confidence of an accepted axiom. Therefore, we have sampled 50 generated axioms for ten confidence intervals each ($[0.01, 0.02)$,$[0.02, 0, 03)$, ..., $[0.09, 0.10)$ and $[0.10, 1.00]$), and manually evaluated their precision. The results are shown in Fig. 6.3. We can observe that the precision considerably drops for a threshold lower than $\tau = 0.05$, i.e., for axioms with a confidence score less than 5%. Hence, we choose $\tau = 0.05$ for a reasonable balance of axiom precision and category coverage.

With a confidence threshold τ of 0.05, we extract 272,707 relation axioms and 430,405 type axioms. In total, they cover 501,951 distinct Wikipedia categories.

Axiom Application and Post-Filtering Applying the extracted axioms to all Wikipedia categories results in 4,424,785 relation assertions and 1,444,210 type assertions not yet contained in $\mathcal{K}$. For the type assertions, we also compute the transitive closure using the *rdfs:subclassOf* statements in the ontology (e.g., also asserting *dbo:MusicalWork* and *dbo:Work* for a type axiom learned for type *dbo:Album*), and thereby end up with 3,342,057 new

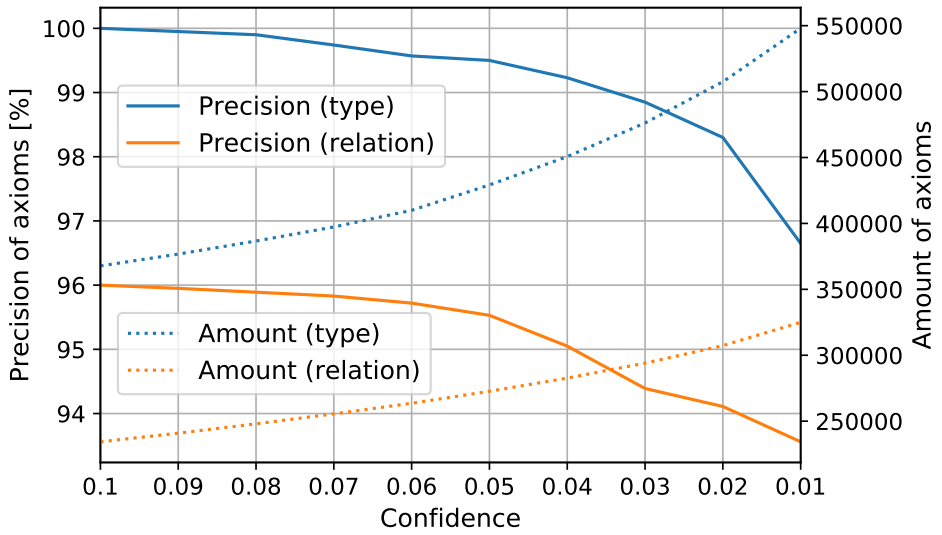


Figure 6.3: Performance of the pattern application for varying confidence intervals. We determined the precision values by manual evaluation of 50 examples per interval.

type assertions (excluding the trivial type *owl:Thing*).

Finally, we remove 72,485 relation assertions and 15,564 type assertions with our post-filtering strategy. An inspection of a small sample of the removed assertions shows that approximately half are incorrect.

6.5.2 Comparison with Related Approaches

We compare Cat2Ax with the two approaches that also use Wikipedia categories to learn axioms and/or assertions for DBpedia: Catriple [113] and C-DF [211]. As both use earlier versions of DBpedia and no code is available, we re-implemented both approaches and ran them with the current version for a fair comparison. For the implementation, we followed the algorithm descriptions in their publications and used the variant with the highest reported precision (i.e., for Catriple, we do not materialize the category hierarchy, and for C-DF, we do not apply patterns iteratively). Running Cat2Ax, Catriple, and C-DF with DBpedia takes 7, 8, and 12 hours, respectively.

Table 6.2 shows the extraction and evaluation results of the three approaches. We evaluate 250 examples per approach for both axioms and assertions. Since the Catriple approach does not produce type information, this adds up to 2,500 examples (1,250 axioms and 1,250 assertions). Each example is labelled by three annotators from the crowdsourcing platform Amazon Mechanical Turk.[8] For the labelling, the axioms and assertions are

[8] `https://www.mturk.com/`

Approach	Count	Precision [%]	Count	Precision [%]
	Relation axioms		Type axioms	
Cat2Ax	272,707	95.6	430,405	96.8
C-DF	143,850	83.6	28,247	92.0
Catriple	306,177	87.2	–	–
	Relation assertions		Type assertions	
Cat2Ax	4,424,785 (7,554,980)	87.2 (92.1)	3,342,057 (12,111,194)	90.8 (95.7)
C-DF	766,921 (2,856,592)	78.4 (93.4)	198,485 (2,352,474)	76.8 (97.1)
Catriple	6,260,972 (6,836,924)	74.4 (76.5)	–	–

Table 6.2: Total number of axioms/assertions and precision scores, based on the crowd-sourced evaluation. Numbers in parentheses denote the *total* number of assertions generated (including those already existing in DBpedia) and the precision estimation of those total numbers. The latter was derived as a weighted average from the human annotations and the overall correctness of existing assertions in DBpedia according to Färber et al. [42].

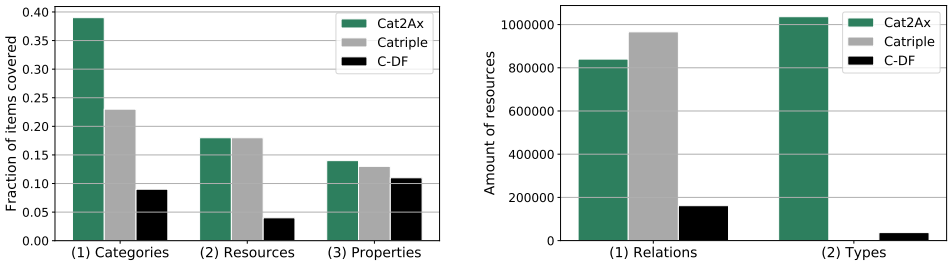
presented in natural language (using labels from DBpedia) and have to be annotated as being either correct or incorrect. The annotators evaluate batches of 50 examples selected from the complete example pool and displayed randomly. The inter-annotator agreement according to Fleiss' kappa [48] is 0.54 for axioms and 0.53 for assertions, indicating moderate agreement [102].

Compared with existing approaches, Cat2Ax outperforms C-DF in the quality and quantity of the created axioms. Catriple produces about 40% more relation assertions but at considerably lower precision and cannot generate type axioms and assertions.

Despite our efforts of post-filtering generated assertions, we observe a large gap between the precision of axioms and assertions. This is more evident when looking at *new* assertions, while the overall precision considering both kinds of assertions (in/out of $\mathcal{A}$) is typically higher. Moreover, there is a small number of axioms which are incorrect and, at the same time, very productive, i.e., they contribute a lot of new incorrect assertions. To further look into these issues, we manually inspected some of those axioms and identified three major causes of errors:

Incorrect data in DBpedia We extract the axiom (*Roads on the National Register of Historic Places in Arizona, rdf:type, dbo:Building*) because many roads in DBpedia are typed as buildings.

Correlation instead of causation We extract the axiom (*University of Tabriz alumni, dbo:birthPlace, dbr:Tabriz*) because people often study in the vicinity of their birthplace.



(a) Fraction of (1) categories with at least one axiom, (2) resources with at least one assertion, (3) properties with at least 100 assertions.

(b) Number of resources without assertions in DBpedia for which (1) a relation assertion or (2) type assertion has been found.

Figure 6.4: Comparison of the extracted results.

Incorrect generalisation We extract the axiom (*Education in Nashik district, rdf:type, dbo:University*), which holds for many instances in the category but not for all of them. This kind of error is most often observed for mixed categories – as in the example, the category contains both universities and schools.

In Fig. 6.4, we compare the results of the three approaches regarding their data source coverage. Fig. 6.4a shows the number of covered (1) categories, (2) resources and (3) properties. For (1), Cat2Ax finds an axiom for almost 40% of Wikipedia's categories. However, the difference between Cat2Ax and Catriple is no longer visible in (2). This can be traced back to Catriple not using any background knowledge during their creation of results, thus producing more productive axioms in terms of generated assertions. (3) shows that all approaches find assertions for a comparable number of properties.

Fig. 6.4b shows statistics for resources currently not described by any relation or type in DBpedia. While Cat2Ax and Catriple find relations for almost one million resources, Cat2Ax additionally finds types for more than one million untyped resources.

6.6 Conclusion

This chapter presented an approach that extracts high-quality axioms for Wikipedia categories. We used the axioms to mine new assertions for KGs. For DBpedia, we added 4.4M relation assertions at a precision of 87.2% and 3.3M type assertions at a precision of 90.8%. Our evaluation showed that our approach produces significantly better results than existing approaches.

Although we conducted experiments with DBpedia, the approach is not limited to only this KG. Any KG linked to Wikipedia (or DBpedia) can be

extended with the approach. Moreover, Cat2Ax could also be applied to KGs created from other Wikis, such as DBkWik [75], or used with different hierarchies, such as WebIsALOD [74]. Hence, Cat2Ax has general potential that goes beyond DBpedia and Wikipedia. The implemented approach is published as stand-alone version[9] and as part of the CaLiGraph extraction framework.[10]

In the previous chapter, we applied Cat2Ax to produce additional assertions that may be integrated into CaLiGraph. In addition to that, we enhance the types in the CaLiGraph ontology with the generated axioms. Fig. 1.1 on page 6 shows how such an axiom is integrated into the ontology of the KG. The complete list of triples for the restriction on *Guns N' Roses album* is given in Example 2.1.6 on page 15.

[9] `https://github.com/nheist/Cat2Ax`
[10] `https://github.com/nheist/CaLiGraph`

Part III

Knowledge Graph Population

CHAPTER 7

Subject Entity Detection in Wikipedia Listings

In Chapter 5, we implemented an initial approach for the extraction of entities from listings. This chapter presents an approach that addresses the two main drawbacks of the former: the limitation to list pages and the negligence of unlinked entity mentions. The approach uses a Transformer network to identify SEs on the token level. Due to the flexible input format of Transformers, it can be applied to any listing and is independent of entity boundaries as input. The contributions of this chapter are as follows:

- We present an approach for SE detection in listings with the following advantages:
  - The input format of the Transformer allows the application to any listing and considers dependencies between listing items.
  - The approach detects SEs end-to-end without relying on mention boundaries of the entities in the input sequence.
  - A mechanism for generating negative samples of listings and a fine-tuning with noisy listing labels lead to more accurate predictions.
- In our evaluation, we show that the performance of our approach is superior to previous work and analyse its performance in the more general scenario of arbitrary listings on Wikipedia pages.
- We run the extraction of SEs on the complete Wikipedia corpus and incorporate the results in a new version of CaLiGraph.

The work presented in this chapter is based on the following publication:

Nicolas Heist and Heiko Paulheim. Transformer-based Subject Entity Detection in Wikipedia Listings. In *Proceedings of the 6th International Workshop on Deep Learning for Knowledge Graphs* (DL4KG @ ISWC'22), vol. 3342, October 2022, CEUR Workshop Proceedings. [71]

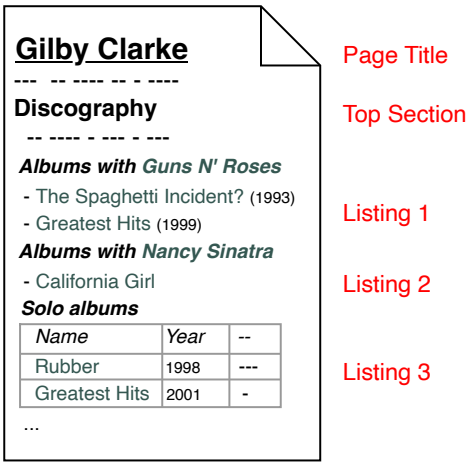


Figure 7.1: Simplified view on the listings of the Wikipedia page of Gilby Clarke. Entity mentions are highlighted in green.

7.1 Motivation

Background knowledge is essential in tasks like text summarisation or question answering. With ready-to-use EL tools like Falcon [172], entities in a text can be identified, and additional information can be drawn from background KGs like DBpedia or CaLiGraph. Of course, this is only possible if the necessary information about the entity is included in the KG [191]. Hence, it is important to equip KGs with as much entity knowledge as possible. While this is easily possible for prominent, often-mentioned entities, retrieving information about infrequently mentioned long-tail and emerging entities is tedious [44].

In this chapter, we generalize over structures like enumerations (*Listings 1 and 2*) and tables (*Listing 3* in Fig. 7.1) by simply viewing them as listings with listing items (i.e., enumeration entries or table rows). Further, we generalize the definition of SEs from Section 5.1 to arbitrary listings. We define SEs as *all entities in a listing appearing as instances to a common concept*. In the case of Fig. 7.1, the SEs are the mentioned albums (e.g., *The Spaghetti Incident?* or *California Girl*). Here, the common concept is made explicit through the section labels above the listings (*Albums with..*), but it may as well only be implicitly defined through the respective SEs. As a listing item typically mentions only one SE together with some context (in this case, the publication year of the album), we assume that at most one SE per listing item exists.

In the English Wikipedia alone, we find almost five million listings in roughly two million articles. From our estimation, about 80% of the listings are suitable for extracting SEs, bearing an immense potential for KGE (for details, see Section 7.3.1). Upon extraction, they can easily be digested

by downstream applications. Due to the semi-structured nature of listings, the extraction quality is higher than extraction from plain text, and SEs are typically extracted in groups of instances sharing a common concept (as given by the definition above). The latter point makes the subsequent disambiguation step much easier, as the group of all extracted instances provides context for every individual instance. Another example of the downstream use of SEs will be demonstrated in Chapter 9 where we use groups of SEs to learn lexical patterns that entail axioms. For example, if a listing is in a section that starts with *Albums with*, we learn that the SEs are of the type *Album*.

The combination of these two ideas, i.e., extracting novel SEs and learning defining axioms for them, can bring a big benefit. In Fig. 7.1, instead of simply discovering *California Girl* as a new entity, we additionally assign the type *Album*. Thinking further, we can learn an axiom that all albums mentioned in the discography of *Gilby Clarke* are albums that he authors. The additional information can be used to refine the description of the extracted entity in the KG.

The rest of this chapter is structured as follows. Section 7.2 frames the presented approach in related works. Section 7.3 lays out the preliminaries of our work, followed by an introduction of our approach in Section 7.4. In Section 7.5, we evaluate our approach and discuss the results of its application to the whole Wikipedia data corpus.

7.2 Related Work

With the presented approach, we detect SEs end-to-end directly from the listing text. For a given listing, we identify mentions of named entities and decide at the same time whether they are SEs of a listing or not. In the following, we describe relevant NER approaches and discuss approaches that detect SEs.

7.2.1 Named Entity Recognition

The NER task has been introduced in Section 2.2.2. Early NER systems are based on hand-crafted rules and lexicons, followed by systems using feature engineering and ML [135]. One of the first competitive NER systems that used neural networks was presented by Collobert et al. in 2011 [33]. This eventually led to more sophisticated architectures based on word embeddings and LSTMs (e.g. from Lample et al. [101]).

With the rise of Transformer networks [192] like BERT [37] in 2018, they also found their direct application in NER (e.g., by Liang et al. [109]), or as part of an end-to-end EL system like the one from Broscheit [21]. The latter uses a simple but effective prediction scheme, where entities are predicted

at the token level, and multiple subsequent tokens with the same predicted entity are collapsed into the actual entity prediction. Our work uses a similar token-level prediction scheme to detect SEs.

7.2.2 Subject Entity Detection

Although SE detection has not explicitly been addressed very frequently in the literature, some approaches deal with its related problems or subproblems. In table interpretation, an important task is the identification of the *subject column*, i.e., the column containing the entity with outgoing relations to all other columns. TAIPAN [40] is an approach that aims to recover the semantics of tables and names subject column identification as the first major task towards relation extraction in tables. To identify subject columns, they choose the columns having entities with the most outgoing edges to entities in other columns w.r.t. a background KG. While this is a viable approach for tables already annotated with entities, it is not broadly applicable to general listings that may not have many known (or even annotated) entities.

Another related approach is from Zhao et al. [219], who deal with a problem they call *key entity detection*. Primarily, they do sentiment analysis in financial texts and use the detection of key entities – which they define as subjects of events related to financial information – to attribute the positive or negative sentiment to a concrete entity. Similar to our proposed approach, they use a Transformer to detect key entities. However, they only use it to select the key entities from a predefined set of entities and ignore the NER part.

As mentioned in the introduction, the most closely related approach is the work presented in Chapter 5: using manually defined features and a binary XGBoost classifier, entities on list pages are classified into subject or non-subject entities. For the page *List of Japanese speculative fiction writers*,[1] for example, all entities in the enumerations that are *Japanese speculative fiction writers* are classified as SEs.

More concretely, the approach uses page features (e.g., the number of sections or tables on a page), positional features (e.g., indentation level of an entry in an enumeration), and linguistic features (e.g., whether the column header is a synonym of the list page title). Overall, SEs are extracted with a precision of 90% and a recall of 67%. The classifier is trained and evaluated with a set of list pages annotated through distant supervision with DBpedia as background knowledge. This part is discussed in detail in Section 7.3.2 as the approach presented here also relies on this training data generation strategy.

[1] https://en.wikipedia.org/wiki/List_of_Japanese_speculative_fiction_writers

7.3 Preliminaries

7.3.1 Listings in Wikipedia

Overall, the English Wikipedia has more than five million articles. Roughly two million contain at least one listing in the form of an enumeration or a table. All these pages contain 3.5 million enumerations and 1.4 million tables.[2] The roughly 90K list pages in Wikipedia contain the most structured and easily exploitable form of listings. Here, listings are almost exclusively used to list several entities with some common property (e.g., all Japanese speculative fiction writers).

Listings that appear on other Wikipedia pages are used for this purpose as well, but not exclusively, making detecting SEs much more complex. From a sample of Wikipedia listings, we estimate that our approach is applicable to approximately 85% of enumerations and 67% of tables. Enumerations are often used to structure content (e.g., to list the individual episodes in a biography). But even if listings are used to describe entities, they may not be usable due to various reasons:

- Entity description without explicit mention (example in Fig. 7.2a)
- Description of the properties of a single entity (example in Fig. 7.2b)
- Listing items contain groups of entities (example in Fig. 7.2c)

The first point renders many tables useless for our approach as an entity is implicitly described through entities and literals mentioned in multiple table columns (e.g., a sports match is described through date, player, opponent, and result).

7.3.2 Distantly-Supervised Training Data Generation for List Pages

We use the training data generation strategy we introduced in Section 5.4.1 to train the models in our experiments. The strategy is based on the observation that DBpedia types, Wikipedia categories, and Wikipedia list pages can be transformed into an immense taxonomy through linguistic and statistical methods. For example, the taxonomy contains the hierarchy *Person* > *Writer* > *Speculative fiction writer* > *Japanese speculative fiction writer*. The first two elements originate from DBpedia types, the third from a category, and the last from a list page.

Consequently, we can use this hierarchy to infer the DBpedia classes of SEs for many list pages. To label the list page *List of Japanese speculative fiction writers*, we assign every entity with the DBpedia class *Writer* a positive label and every entity with a class that is disjoint with *Writer* a negative label.

[2] These numbers exclude very small listings with less than three items, which we do not consider.

Awards and achievements [edit]

Awards [edit]

Man of the Match

No.	Date	Player	Opponent	Venue	Result	Contribution	Ref.
1	9 April 2018	Shikhar Dhawan	Rajasthan Royals	Hyderabad	Won by 9 wickets	77* (57)	[24]
2	12 April 2018	Rashid Khan	Mumbai Indians	Hyderabad	Won by 1 wicket	0 (1) & 1/13 (4 overs)	[27]
3	14 April 2018	Billy Stanlake	Kolkata Knight Riders	Kolkata	Won by 5 wickets	2/21 (4 overs)	[30]
4	24 April 2018	Rashid Khan	Mumbai Indians	Mumbai	Won by 31 runs	6 (9) & 2/11 (4 overs)	[39]
5	29 April 2018	Kane Williamson	Rajasthan Royals	Jaipur	Won by 11 runs	63 (43)	[43]
6	5 May 2018	Rashid Khan	Delhi Daredevils	Hyderabad	Won by 7 wickets	2/23 (4 overs)	[46]
7	7 May 2018	Kane Williamson	Royal Challengers Bangalore	Hyderabad	Won by 5 runs	56 (39)	[48]
8	10 May 2018	Shikhar Dhawan	Delhi Daredevils	New Delhi	Won by 9 wickets	92* (50)	[51]
9	25 May 2018	Rashid Khan	Kolkata Knight Riders	Kolkata	Won by 13 runs	34 (10) & 3/19 (4 overs)	[65]

(a) Listing containing no explicit mention of the entities
Source: `https://en.wikipedia.org/wiki/Sunrisers_Hyderabad_in_2018`

Uses [edit]

DHTML allows authors to add effects to their pages that are otherwise difficult to achieve, by changing the Document Object Model (DOM) and page style. The combination of HTML, CSS and JavaScript offers ways to:

- Animate text and images in their document.
- Embed a ticker or other dynamic display that automatically refreshes its content with the latest news, stock quotes, or other data.
- Use a form to capture user input, and then process, verify and respond to that data without having to send data back to the server.
- Include rollover buttons or drop-down menus.

(b) Listing describing the properties of an entity
Source: `https://en.wikipedia.org/wiki/Dynamic_HTML`

Grape varieties [edit]

- Red: Monastrell, Tempranillo, Cabernet sauvignon, Merlot and Syrah
- White: Macabeo, Parellada, Malvasía, Chardonnay and Moscatel

(c) Listing containing groups of entities
Source: `https://en.wikipedia.org/wiki/Ibiza_(Vino_de_la_Tierra)`

Figure 7.2: Examples of Wikipedia page listings with layout or content challenging for SE detection.

Then, we include all listing items into our training set that either have an entity with a positive label or only entities with negative labels. Other listing items are ignored as we cannot be certain that they may contain SEs, which we could not identify due to the incompleteness of DBpedia.

CaLiGraph uses this extended taxonomy of DBpedia types, categories, and list pages as a type hierarchy, thereby enriching the original DBpedia instances with additional, more fine-grained types. Furthermore, CaLiGraph contains more instances than DBpedia as it contains the extracted SEs from list pages.

7.3.3 Transformers for Token Classification

Pre-trained Transformer networks [192] like BERT [37] and DistilBERT [173] produced new state-of-the-art results for various NLP tasks, including NER and question answering. To a large extent, their ubiquitous application is due to the fact that only a comparably small amount of fine-tuning is necessary to fit them to various tasks. BERT, for instance, consists of 12 multi-head attention layers followed by a simple linear layer as a classification head. It is often sufficient to fine-tune the final classification head to apply a Transformer model to a token classification problem.

The input for a Transformer model can consist of plain text and must be tokenised before processing. Every word in the input sequence is transformed into one or more tokens (if the word is not contained in the vocabulary, multiple word-piece tokens are used). Further, the input sequence has to contain special tokens that indicate, for example, the sequence's start and end. Using BERT for token classification, the input sequence has a fixed length of 512 tokens, has to start with a *[CLS]* token and end with a *[SEP]* token. Additional special tokens may be introduced to provide more context information to the model.

7.4 Subject Entity Detection with Transformers

To detect SEs in listings, we phrase the problem as a token classification problem where we, similar to the work of Broscheit [21], produce a label for every token of the input sequence. In a subsequent step, we aggregate the token labels to predictions of SE mentions. We use 13 different token labels, such as *Person* or *Organisation*, to identify SEs and additionally make a prediction of their types (refer to Table 7.5 for the full list of labels). In Section 7.4.1, we explain how to create input sequences that preserve the context and the structure of a listing. In Section 7.4.2, we show our choice of labels for SE prediction, and in Section 7.4.3, we introduce a mechanism to generate negative samples of listings. Finally, Section 7.4.4 explains how to use noisy SE labels on page listings for further fine-tuning of our models.

7.4.1 Token-level Subject Entity Detection

To pass a listing for SE detection to the Transformer model, we use multiple special tokens to encode context information (page, section, potential table header) and structural information (entries, rows, columns) of the listing into the input sequence. Every sequence consists of the listing context, followed by the special token indicating the end of context *[CXE]*, and one or more listing items:

```
[CLS] <context> [CXE] <listing items> [SEP]
```

We use the special token *[CXS]* to separate context elements. Within listing items, table rows and columns are indicated with *[ROW]* and *[COL]*, respectively. For enumerations, we use the tokens *[E1]* to *[En]* to indicate the start of an entry with the indentation level *1* to *n*.

Ignoring that some words may be split into multiple tokens, the input for the first listing item of *Listing 1* in Fig. 7.1 is shown in Example 7.4.1.

Example 7.4.1. Single enumeration list item as input

[CLS] Gilby Clarke [CXS] Discography [CXS] Albums with Guns N' Roses [CXE]
[E1] The Spaghetti Incident? (1993) [SEP]

We want the model to take dependencies between listing entities into account. For example, if the SE in the first listing item is mentioned right in the beginning, this is likely the case for the remaining listing items as well. Instead of only one listing item per input sequence, we provide as many as the maximum length permits. The model can take these dependencies within the input sequence into account through the attention layers of the Transformer architecture. As shown in Example 7.4.2, we put *Listing 1* into one input sequence.

Example 7.4.2. Multiple enumeration list items as input

[CLS] Gilby Clarke [CXS] Discography [CXS] Albums with Guns N' Roses [CXE]
[E1] The Spaghetti Incident? (1993)
[E1] Greatest Hits (1999) [SEP]

Likewise, we encode *Listing 3* as one input sequence in Example 7.4.3.

Example 7.4.3. Multiple table list items as input

[CLS] Gilby Clarke [CXS] Discography [CXS] Solo albums [CXS]
[ROW] Name [COL] Year [CXE]
[ROW] Rubber [COL] 1998
[ROW] Swag [COL] 2001 [SEP]

If the listing is too long to fit into one input sequence, we split the listing items into chunks and process them one after another. Each chunk is augmented with the same context information and a different set of listing

items. Depending on the length of listing items, it is possible to fit 20 or more items into one input sequence. In our ablation study in Section 7.5.5, we show that this item chunking strategy has a strongly positive effect on the recall of the model. We also immensely reduce the model's run time for training and prediction. The number of processed input sequences is reduced by a factor roughly equivalent to the median number of items per listing.[3]

7.4.2 Coarse-grained Entity Type Prediction

The most common notation to tag tokens in NER is the BIO notation (*Begin*, *Inside*, and *Outside* of an entity) together with an entity type (e.g., *Person* or *Organisation*). We decided not to use the BIO notation as, per definition, there is at most one SE per listing item. Instead of making the task even simpler and getting rid of the entity type prediction in favour of a simple binary SE prediction task as well, we decided to stick with the coarse-grained entity type prediction. This comes with the advantage that the entity types can be used as additional information in downstream tasks – most importantly in a subsequent entity disambiguation step. We will see in our ablation study in Section 7.5.5 that the more difficult task of entity type prediction even slightly increases the precision of the model.

Context and special tokens are annotated with the *IGN* label (for *IGNORE*) to indicate to the model that we need no prediction for these tokens. SEs are annotated with the respective entity type; everything else is annotated with *NONE*. Again ignoring word-piece tokenization, the labels for *Listing 1* of Fig. 7.1 are shown in Example 7.4.4.

Example 7.4.4. Type labels as input

IGN IGN IGN IGN IGN IGN IGN IGN IGN IGN IGN IGN
IGN WORK_OF_ART WORK_OF_ART WORK_OF_ART NONE
IGN WORK_OF_ART WORK_OF_ART NONE IGN

7.4.3 Negative Sampling through Shuffled Listings

It is difficult to find negative examples of complete listings if the training data is generated heuristically and with distant supervision as described in Section 7.3.2. Positives can be found easily (i.e., there is an entity in the listing item that has the correct type), but the inverse does not always hold. If we do not find a positive, this may mean that the listing item does not contain one, but it is also possible that the annotation is missing (cf. OWA). From a logical standpoint, it is even unlikely that some items in a listing contain SEs while others do not.

[3]We deliberately use the median and not the average of items per listing as large listings will be split into multiple input sequences due to the size limitation.

To mitigate this problem, we equipped our approach with a sampling mechanism for negatives that randomly assembles them from the contexts and items of all positives in the training set. If the context and items are assembled randomly, the differences between the individual items (and the difference in the context) should be higher than in a real listing. The intention behind the mechanism is that the model learns to identify the coherence between SEs of listing items as well as between items and the context.

The mechanism is simple for enumeration listings as we pick the context from one listing and a random number of items (between three and the maximum number of items per chunk) from other listings. For table listings, we must ensure that the number of columns of an assembled listing is consistent. Hence, the positives from the training set are divided into groups of the same column size, and listings are only assembled from within a single group. A negative example produced from four different listings could look as shown in Example 7.4.5.

Example 7.4.5. Negative listing example

[CLS] Gilby Clarke [CXS] Discography [CXS] Albums with Guns N' Roses [CXE]
[E1] James Stewart as Billy Jim Hawkins
[E1] Curzon Mill Company, part of Ashton syndicate.
[E1] Brepholoxa Van Duzee, 1904 [SEP]

The mechanism has exactly one hyper-parameter: the proportion of negative listings to generate. We experiment with values between 0.0 (no negative samples) and 1.0 (as many negatives as we have positives).

7.4.4 Fine-Tuning on Noisy Page Labels

The training data generation strategy described in Section 7.3.2 lets us create labels for listings of list pages that we use for the initial training of our models. To train a model that works well on listings of any pages, additional training data of listings that are not on list pages may be beneficial (the differences in listings have been described in Section 7.3.1).

We gather this data by training a model using the heuristically labelled list pages. We apply the model to listings of all pages for noisy labels of SEs. We then filter them by discarding any listings where multiple types of SEs have been predicted (e.g., if the first SE of a listing is labelled as *PERSON* and the second is labelled as *WORK_OF_ART*).

7.5 Experiments

The goal of our experiments is to compare the performance of our approach against previous work on SE detection in list pages (Section 7.5.3) and evaluate its performance in the more general setting of Wikipedia page

listings (Section 7.5.4). Further, we analyze some of our design choices in an ablation study (Section 7.5.5). Finally, we apply our best model to the complete Wikipedia corpus and report our extraction results (Section 7.5.6).

7.5.1 Metrics

To evaluate our SE detection models, we stick to the common metrics for NER introduced in SemEval-2013 [176]. We report precision, recall, and F1 scores for the following scenarios:
Partial: Prediction matches the boundary of the true entity at least partially
Exact: Prediction matches the true entity's boundary exactly
Ent-Type: At least partial boundary match and entity type matches
Strict: Predicted boundary and type match the true entity exactly

7.5.2 Datasets

In the experiments, we primarily focus on Wikipedia as a data corpus due to its encyclopedic structure and the convenient mapping of entities to DBpedia and CaLiGraph. From the main dataset D, which consists of all Wikipedia pages that contain listings, we create the subsets $D\text{-}LP_{train}$ and $D\text{-}LP_{test}$ (from list pages) as well as $D\text{-}P_{train}$ and $D\text{-}P_{test}$ (from any pages with listings). The statistics of the datasets are shown in Table 7.1. We use *Wikipedia2020* (cf. Section 2.3.4) together with version 1.1.0 of CaLiGraph, which is extracted from this dump, to generate the datasets.

The datasets $D\text{-}LP_{train}$ and $D\text{-}LP_{test}$ are created as explained in Section 7.3.2. For the experiments, we use a part of $D\text{-}LP_{train}$ for validation to have a distribution of 60% training, 20% validation, and 20% test set (similar to the setting in Section 5.4.1).

The datasets $D\text{-}P_{train}$ and $D\text{-}P_{test}$ consist of listings from arbitrary Wikipedia pages. Hence, no type information is available to infer the SE labels through distant supervision. For $D\text{-}P_{train}$, we retrieved the labels as described in Section 7.4.4. For $D\text{-}P_{test}$, we provided the type information by manually annotating the roughly one thousand listings with coarse-grained entity types (e.g., *Person* or *Organisation*). We mapped these types to their CaLiGraph counterparts and used this information to infer the SE labels via distant supervision. This substantially reduced the annotation effort from labelling roughly 10K listing items with concrete SE labels to labelling 1K listings with coarse-grained types. This implies that this dataset is also, in part, heuristically created, and the results have to be taken with a grain of salt.

7.5.3 Evaluation on Wikipedia List Pages

The evaluation results for experiments on the dataset $D\text{-}LP_{test}$ are given in Table 7.2. We compare the existing approach from Chapter 5 (now

Dataset	#Pages	#Listings		#Items (average)		#Items (median)	
		Enums	Tables	Enums	Tables	Enums	Tables
D	2.0M	3.5M	1.4M	10.57	14.43	6	8
$D\text{-}LP_{train}$	69K	290K	117K	18.06	31.26	8	12
$D\text{-}LP_{test}$	17K	75K	29K	18.17	31.32	8	12
$D\text{-}P_{train}$	547K	664K	306K	18.72	24.53	12	13
$D\text{-}P_{test}$	502	763	265	8.42	11.25	6	7

Table 7.1: Statistics of the datasets used for the experiments. The complete corpus D contains all Wikipedia pages that have listings. $D\text{-}LP_{train}$ and $D\text{-}LP_{test}$ are extracted from all Wikipedia list pages and are labelled through distant supervision; $D\text{-}P_{train}$ contains listings from arbitrary pages and contains noisy labels from a model trained on list pages while $D\text{-}P_{test}$ is annotated manually.

Approach	Enums			Tables			Overall		
	P	R	F1	P	R	F1	P	R	F1
Old	91	82	86	**90**	55	68	90	67	77
New_{LP}	**93**	**94**	**94**	89	**87**	**88**	**92**	**91**	**92**
New_P	92	93	93	88	86	87	91	90	91

Table 7.2: Evaluation results for SE detection on Wikipedia list pages (evaluating on $D\text{-}LP_{test}$). Precision, recall and F1-score (in %) are given for the *Exact* scenario. New_{LP} is the best configuration for $D\text{-}LP_{test}$ while New_P is the best configuration for $D\text{-}P_{test}$ using $D\text{-}LP_{train}$ as training data.

called *Old*) with our model in the two configurations New_{LP}[4] and New_P.[5] Both configurations are trained with the training part of $D\text{-}LP_{train}$ and tuned using the evaluation part. The former model configuration is best w.r.t. performance on $D\text{-}LP_{test}$. The latter model configuration is best w.r.t. performance on $D\text{-}P_{test}$. To train our models, we use the Huggingface Transformer library [207].

Both model configurations significantly outperform the *Old* approach, especially regarding recall for both enumerations and tables. This shows that our model can identify substantially more entities while maintaining high precision. The precision increased slightly for enumerations, and the recall is over 10% higher. While precision is kept almost constant for tables, the recall increased by more than 30%.

[4]Config.: Model roberta-base trained for 3 epochs with batch size 64, learning rate 5e-5, no warmup or weight decay, negative sample size 0.5

[5]Config.: Model roberta-base trained for 2 epochs with batch size 64, learning rate 5e-5, no warmup or weight decay, negative sample size 0.3

Metric	Enums			Tables			Overall		
	P	R	F1	P	R	F1	P	R	F1
Partial	76	78	77	68	82	74	73	79	76
Exact	73	76	75	67	81	73	71	77	74
Ent-Type	76	78	77	65	78	71	73	78	75
Strict	71	74	73	64	77	70	69	75	72
Baseline	23	85	36	21	90	34	23	86	36

Table 7.3: Precision, recall and F1-score (in %) for SE detection on page listings (evaluated on $D\text{-}P_{test}$) using our best model configuration New_{final}.

7.5.4 Evaluation on Wikipedia Page Listings

The evaluation results for the model New_{final}[6] on $D\text{-}P_{test}$ is given in Table 7.3. Comparing the *Exact* scenario with the results on Wikipedia list pages, it becomes clear that the performance on arbitrary listings is worse. The losses in performance for tables are slightly higher than for enumerations. This aligns with the observation that our approach applies to fewer tables than enumerations. For tables, we have the advantage that mention boundaries are often indicated through column separators, but this is not reflected in the results. Training the models for over two to three epochs on $D\text{-}LP_{train}$ leads to overfitting on list page data and hence reduced performance on $D\text{-}P_{test}$.

Unfortunately, applying the *Old* approach to this dataset is impossible as it contains several features specific to list pages. As an alternative, we implemented the pick-first-entity baseline, which has already proven to be a strong baseline for the *Old* approach. In this baseline, we label the first mentioned entity in an item as SE. In Table 7.3, we see that this baseline has a very high recall (as most SEs are mentioned in the beginning) while the precision is far lower than the one of New_{final}. This shows that the model can sort out many false positives (tripling precision) by sacrificing only some correct SEs. In cases where coverage is not the only important criterion (as is usually the case in KGE), the New model should be preferred. The more important point, however, is that the New model does not depend on mention boundaries as input (which might also account for some loss in performance).

7.5.5 Ablation Study

To verify some assumptions we made during the design of the SE detection approach, we conducted an ablation study using the page listings dataset $D\text{-}P_{test}$. We investigate how the item chunking in input sequences influences the model's performance. The results in Table 7.4 show a slightly positive effect on precision (3% for enumerations, 4% for tables) and a roughly

[6]Config.: Similar to New_P with an additional fine-tuning step of one epoch on D-P_{train}.

Approach	Enums			Tables			Overall		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
New_{final}	**73**	76	**75**	**67**	81	**73**	**71**	77	**74**
.. without item chunks	70	35	47	63	40	49	68	37	48
.. without type prediction	69	**78**	73	54	**84**	66	64	**79**	71
.. without negative sampling	71	74	73	66	81	73	70	76	73
.. without fine-tuning on pages	65	48	55	67	64	66	66	52	58

Table 7.4: Evaluation results for SE detection on Wikipedia page listings (evaluated on D-P_{test}) for variations of our best model configuration New_{final}. Precision, recall and F1-score (in %) are given for the *Exact* scenario.

doubling effect on recall. The results confirm that the model can improve its predictions by considering the dependencies between the listing items.

Further, we investigate whether the additional prediction of entity types influences the performance (as opposed to a binary prediction of SEs). The results show a positive effect on precision and a slightly negative effect on recall. As the F1 measure increases slightly and the predicted types provide additional information for downstream tasks, we stick with type prediction instead of binary SE prediction.

Additionally, we see from Table 7.4 that our negative sampling mechanism slightly increases the precision and recall of our final model. Consequently, the model seems to be able to learn whether there is some consistency between the listing items in the input sequence.

Finally, the fine-tuning on pages has a very strong effect on recall as it comes with an increase of 25% and the model's precision is also increased by 5%. This result confirms that additional fine-tuning on noisy labels still yields a huge benefit.

7.5.6 Subject Entity Extraction over Wikipedia

Applying the model New_{final} to the complete dataset of Wikipedia listings D took 13 hours on a single NVIDIA RTX A6000 GPU with 48GB of RAM. We extracted 40M entity mentions from 2.7M enumerations and 1M tables on 1.7M pages. Of the 40M entity mentions, 19.5M can be traced back to 3.8M known entities (i.e., the predicted mention boundary overlapped with an existing link in Wikipedia, and hence, CaLiGraph), meaning each known entity has, on average, 5.1 mentions. If we use that same factor of 5.1 to estimate the number of entities for the remaining 20.5M entity mentions, they describe 4M additional unknown entities that could be added to the KG.

In Table 7.5, we display the number of extracted entity mentions aggregated by entity type. Unsurprisingly, the most frequently extracted entities are of the types *Person, Work of Art, Organisation, and Location*. Apart from that, the mention type distribution roughly resembles the distribution of

Entity Type	#Mentions	Entity Type	#Mentions	Entity Type	#Mentions
PERSON	13,622,704	GPE	1,519,747	NORP	230,707
OTHER	9,398,003	PRODUCT	1,000,117	LANGUAGE	86,354
WORK_OF_ART	7,148,235	SPECIES	964,922	LAW	11,490
ORG	2,916,528	FAC	893,226		
LOC	1,531,452	EVENT	370,440	**Total**	**39,693,925**

Table 7.5: Number of extracted mentions of SEs for the whole Wikipedia dataset of listings D aggregated by entity type.

entities in DBpedia (cf. Chapter 3).

7.6 Conclusion

In this chapter, we have presented a Transformer-based SE detection approach that overcomes several limitations of prior work to make it applicable to more general settings and, at the same time, improve extraction performance. An evaluation of listings of Wikipedia pages shows that the performance for such a more general setting is considerably worse than for the scenario of Wikipedia list pages. While the inferior results can partly be attributed to conceptual limitations of SE detection in arbitrary listings (c.f. Section 7.3.1), further improvement is necessary so that downstream applications can consume the results without extensive post-filtering.

The entities extracted here are integrated in CaLiGraph. The implemented approach is published as part of the CaLiGraph extraction framework.[7] However, a subsequent entity disambiguation step is necessary to match the identified SE mentions with existing entities or create new entities in the KG.

[7] `https://github.com/nheist/CaLiGraph`

CHAPTER 8

NASTyLinker: NIL-Aware Scalable Transformer-based Entity Linker

Most prevalent EL approaches assume that the reference KG with the entities to link against is complete. In practice, however, it is necessary to deal with the case of linking to an entity not contained in the KG, i.e., to a NIL entity. In the previous chapter, we have identified a large number of entity mentions in Wikipedia listings, ready to be integrated into a KG. This chapter addresses the task of disambiguating such mentions by mapping them to existing entities in the KG or creating new ones in the case of NIL entities. Our proposed approach, *NASTyLinker*, is aware of NIL entities and produces corresponding mention clusters while maintaining high linking performance for known entities. The contributions of this chapter are as follows:

- We introduce the NASTyLinker approach, which extends existing EL approaches by using a top-down clustering mechanism to link mentions to known entities consistently and produce clusters for NIL mentions.
- In our experiments, we demonstrate the presented approach's competitive linking performance and scalability through an evaluation on the NILK dataset.
- We use NASTyLinker for KGP by linking entities in Wikipedia listings. We report on the linking statistics and provide a qualitative analysis of the results.

The work presented in this chapter is based on the following publication:

Nicolas Heist and Heiko Paulheim. NASTyLinker: NIL-Aware Scalable Transformer-based Entity Linker. In *The Semantic Web - ESWC 2023. Lecture Notes in Computer Science*, vol. 13870, pp. 174-191, Hersonissos, Greece, May 2023, Springer, Cham. [72]

Name	Township(s)	Coordinates	NTS map	Status	CGNDB id
Jackpine Lake	Banting, Chambers	47°8′44″N 79°56′3″W	031M/04	Official	FBRBM
James Lake	Best	47°10′41″N 79°44′26″W	031M/04	Official	FBRHD
Jamieson Lake	Banting	47°9′22″N 79°59′37″W	031M/04	Official	FBRHY
Jessie Lake	Strathcona	47°2′28″N 79°48′14″W	031M/04	Official	FBRSJ
Jumping Cariboo Lake	Law, Olive	46°52′57″N 79°46′32″W	031L/13	Official	FBSOZ
Jumpingcat Lake	Belfast, Joan	47°1′48″N 80°9′59″W	041P/01	Official	FBSPA

(a) A lake in Ontario, CA.
Lakes of Temagami

- Elbow Lake, 46°21′53″N 113°01′31″W, el. 7,746 feet (2,361 m)[14]
- Evans Lake, 47°00′15″N 113°04′19″W, el. 4,193 feet (1,278 m)[15]
- Hagan Pond, 46°28′46″N 112°52′55″W, el. 5,000 feet (1,500 m)[16]
- James Lake, 47°04′32″N 113°12′43″W, el. 4,137 feet (1,261 m)[17]
- Jones Lake, 47°02′35″N 113°08′35″W, el. 4,088 feet (1,246 m)[18]
- Kleinschmidt Lake, 46°58′33″N 113°02′35″W, el. 4,186 feet (1,276 m)[19]

(b) A lake in Montana, US.
List of lakes of Powell County, Montana

Members [edit]

- Lionel Williams – vocals, various instruments (2007–present)

Associated musicians [edit]

- Bryan Lee – drums (2007–2010)
- Calin Stephensen – bass (2007–2008)
- James Lake – drums/synth (2011–2017)
- Ian Gibbs – various instruments (2011–2018)

(c) A musician in the band Vinyl Williams.
Vinyl Williams

Character	Actor/Actress	Duration
Joe Lacerra	Stephen Liska	1998–2005
Cindy Lake	DeAnna Robbins	1982–83
James Lake	Glenn Corbett	1983
Mary Margaret Lake	Fawne Harriman	1983
Sammy Lake	Danny McCoy Jr.	1978
Hilary Lancaster	Kelly Garrison	1991–93
Dr. Joshua Landers	Heath Kizzier	1996–98

(d) A character in a soap opera.
List of The Young and Restless characters

Figure 8.1: Listings in Wikipedia containing the mention *James Lake*. All of the mentions refer to distinct entities. A Wikipedia article exists only for the entity of the mention in (a).

8.1 Motivation

EL is crucial for many downstream tasks like question answering [36, 200], or KG population and completion [147]. One main challenge of EL is the inherent ambiguity of entities mentioned in the text. Fig. 8.1 shows four homonymous mentions of distinct entities with the name *James Lake* (a lake in Canada, a lake in the US, a musician, and a fictional character). Correctly linking the mentions in Figs. 8.1a and 8.1b is incredibly challenging as both mentions point to geographically close lakes.

In a typical EL setting, we assume the training data mentions all entities to be linked against. This assumption is dropped in Zero-Shot EL [117], where a linking decision is made based on entity information in the reference KG (e.g., textual descriptions, types, relations). In this setting, a seminal approach has been introduced with BLINK [210] (cf. Section 2.2.3).

In a practical setting, we additionally encounter the problem of mentions without a corresponding entity in the reference KG. The mention in Fig. 8.1a is the only one with a counterpart in a reference KG where entities are based on Wikipedia articles. For the other mentions, a correct prediction based on the reference KG is impossible. Instead, NIL-aware approaches could either (1) create an (intermediate) entity representation for the NIL entity to link or (2) produce clusters of NIL mentions with all mentions in a cluster referring to the same entity.

While this problem has been largely ignored by EL approaches for quite some time, recent works demonstrate that reasonable predictions for NIL

mentions can be made by clustering mentions based on inter-mention affinities [1, 94]. Both compute inter-mention and mention-entity affinities using a bi-encoder architecture based on BLINK [210]. EDIN [94] is an approach of category (1) that uses a dedicated adaptation dataset to create representations for NIL entities in an unsupervised fashion. Hence, the approach can only link to a NIL entity if there is at least one mention in the adaptation dataset.

For some EL tasks, especially as a prerequisite for KGP, creating an adaptation dataset with good coverage is not trivial because an optimal adaptation dataset has to contain mentions of all NIL entities. Agarwal et al. [1] present an approach of category (2) that creates clusters of mentions and entities in a bottom-up fashion by iteratively merging the two most similar clusters, always under the constraint that a cluster must contain at most one entity.

With NASTyLinker, we propose an EL approach that is NIL-aware in the sense of category (2), hence avoiding the need for an adaptation dataset. Like Agarwal et al., it produces clusters of mentions and entities based on inter-mention and mention-entity affinities from a bi-encoder. NASTyLinker relies on a top-down clustering approach that assigns mentions to the entity with the highest transitive affinity in case of a conflict. Contrary to Agarwal et al., who discard cross-encoders completely due to the quadratic growth in complexity when evaluating inter-mention affinities, our experiments show that applying a cross-encoder only for the refinement of mention-entity affinities can result in a considerable increase of linking performance at a reasonable computational cost.

The rest of this chapter is structured as follows. Section 8.2 gives an overview of the state of the art in NIL-aware EL. Section 8.3 formally defines the task, followed by a description of NASTyLinker in Section 8.4. We evaluate our approach in Section 8.5 and discuss its performance in a KGP task.

8.2 Related Work

Entity Linking

EL (introduced in Section 2.2.2) has been studied extensively in the last two decades [161, 178]. Initially, approaches relied on word and entity frequencies, alias tables, or neural networks for their linking decisions [34, 50, 129]. The introduction of pre-trained Transformer models [37] made it possible to create representations of mentions and entities from text without relying on other intermediate representations. Gillick et al. [53] shows how to learn dense representations for mentions and entities; Logeswaran et al. [117] extend this by introducing the zero-shot EL task and demonstrate that

reasonable entity embeddings can be derived solely from entity descriptions. Wu et al. [210] introduce BLINK, the prevalent bi-encoder and cross-encoder paradigm for zero-shot EL. Based on this paradigm, various improvements for zero-shot EL have been proposed. KG-ZESHEL [167] adds auxiliary entity information from KG embeddings into the linking process; Partalidou et al. [145] propose alternative pooling functions for the bi-encoder to increase the accuracy of the candidate generation step.

Cross-Document Coreference Resolution (CDC)

NIL-Aware EL is closely related to CDC, the task of identifying coreferent entity mentions in documents without explicitly linking them to entities in a KG [10]. Dutta and Weikum [39] explicitly tackle CDC in combination with EL by applying clustering to bag-of-words representations of entity mentions. More recently, Logan IV et al. [116] evaluate greedy nearest-neighbour and hierarchical clustering strategies for CDC, however, without explicitly evaluating them with respect to EL.

Entity Discovery and NIL-Aware EL

The majority of EL approaches may identify NIL mentions (for instance, through a binary classifier or a ranking that explicitly includes *NIL*) but does not process them in any way [178, 179]. In 2011, the TAC-KBP challenge [92] introduced a task that includes NIL clustering; in the NEEL challenge [171] based on microposts, NIL clustering was part of the task as well. Approaches that tackle these tasks typically apply clustering based on similarity measures over the entity mentions in the text [18, 41, 55, 132, 160]. More recently, Angell et al. [6] train two separate bi-encoders and cross-encoders to compute inter-mention and mention-entity affinities. Subsequently, they apply a bottom-up clustering for refined linking predictions within single biomedical documents. Agarwal et al. [1] extend the approach to cross-document linking through a clustering based on minimum spanning trees for all mentions in the corpus. Clusters are formed by successively adding edges to a graph as long as the constraint that a cluster can contain at most one entity is not violated. Instead, they omit the cross-encoder and employ a custom training procedure for the bi-encoder. They explicitly evaluate their approach w.r.t. NIL entity discovery by removing some entities from zero-shot EL benchmark datasets in the training set. Our approach employs a similar method for computing affinities but with a top-down clustering approach to better identify clusters of NIL mentions. The EDIN pipeline [94] also applies clustering w.r.t. inter-mention and mention-entity affinities, but only to identify NIL mention clusters on a dedicated adaptation dataset. Subsequently, the entity index is enhanced with pooled representations of these clusters to predict NIL entities. In their clustering phase, they first produce groups

of mentions and then identify NIL mention clusters by checking whether less than 70% of the mentions refer to the same entity. As we aim to apply NIL-aware EL for KGP, relying on an adaptation dataset is impossible. Still, we include the clustering method of the EDIN pipeline in our experiments to compare how well the approaches detect NIL mention clusters.

8.3 Task Formulation

A document corpus $\mathcal{D}$ contains a set of textual entity mentions $\mathcal{M}$. Each mention $m \in \mathcal{M}$ refers to an entity e in the set of all entities $\mathcal{E}^*$. Given a KG $\mathcal{K}$ with known entities $\mathcal{E}^k$,[1] the task in standard EL is to assign an entity $\hat{e} \in \mathcal{E}^k$ to every mention in $\mathcal{M}$. In this setting, we assume that $\mathcal{E}^* = \mathcal{E}^k$, i.e., all entities are contained in $\mathcal{K}$.

In NIL-aware EL, we drop the assumption that every mention links to an entity contained in $\mathcal{K}$. Instead there is a set of NIL entities $\mathcal{E}^n$ with $\mathcal{E}^k \cup \mathcal{E}^n = \mathcal{E}^*$ and $\mathcal{E}^k \cap \mathcal{E}^n = \emptyset$. For mentions $\mathcal{M}^k$ that refer to entities in $\mathcal{K}$, the task is still to predict an entity $\hat{e} \in \mathcal{E}^k$. For mentions $\mathcal{M}^n$ referring to entities not contained in $\mathcal{K}$, the task is to predict a cluster identifier $c \in \mathcal{C}$ so that the clustering $\mathcal{C}$ resembles the distribution of mentions in $\mathcal{M}^n$ to entities in $\mathcal{E}^n$ as closely as possible. We assume that we are additionally operating in a zero-shot setting, i.e., the training portion $\mathcal{D}_{train}$ of the document corpus may not contain mentions for all entities in $\mathcal{E}^*$.

Similar to related works [1, 117], we assume that the textual entity mentions are already given. Further, we only investigate the relevant steps for KGP, i.e., detection and disambiguation of NIL entities. While we discard the indexing aspect, an EL model, which includes the entities in $\mathcal{E}^n$, can still be created in a subsequent step by training a new model on the enhanced KG.

8.4 NASTyLinker: An Approach for NIL-Aware and Scalable Entity Linking

This section describes our proposed approach for making NIL-aware EL predictions. Fig. 8.2 depicts the three main phases of the NASTyLinker approach. In the *Linking Phase*, we first retrieve inter-mention and mention-entity affinities from an underlying EL model for the subsequent clustering. We define constraints for such a model and describe the one used in our experiments in Section 8.4.1. During the *Clustering Phase*, clusters of mentions and entity candidates are created using greedy nearest-neighbour clustering (Section 8.4.2). Finally, we retrieve entity candidates for every cluster. In

[1] In our original definition in Section 2.2, we use $\mathcal{E}$ for all entities in the KG. Here, we use $\mathcal{E}^k$ to highlight the difference to $\mathcal{E}^n$.

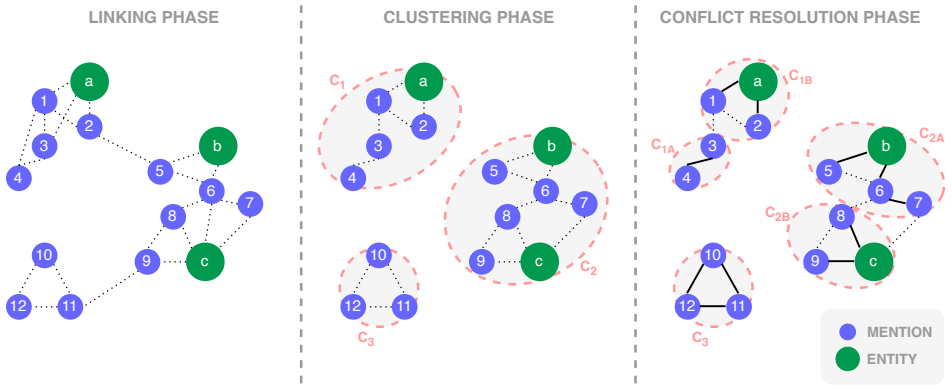


Figure 8.2: Main phases of the NASTyLinker approach. Dotted lines show top-k affinity scores; solid lines indicate the highest transitive affinity scores.

the *Conflict Resolution Phase*, clusters are split based on transitive mention-entity affinities to ensure that a cluster contains at most one known entity (Section 8.4.3).

8.4.1 Entity Linking Model

In the *Linking Phase*, we compute the k most similar mentions and entities for every mention in $\mathcal{M}$ (dotted lines in Fig. 8.2). The underlying EL model has to provide a function ϕ with $\phi(m, e) \in [0; 1]$ for the similarity between mention m and entity e as well as $\phi(m, m') \in [0; 1]$ for the similarity between mentions m and m'. In addition, efficiently retrieving the top k mention and entity candidates for a given mention must be possible.

For our experiments with NASTyLinker, we choose the BLINK architecture [210] as the underlying EL model as it provides the foundation for many state-of-the-art EL models. Furthermore, as the bi-encoder creates embeddings for mentions and entities alike, methods for an approximate nearest-neighbour search like FAISS [93] can retrieve linking candidates efficiently. As the application of the cross-encoder is the most time-consuming part of this model, we explore in our experiments the trade-off between linking performance and runtime when reranking only inter-mention affinities, mention-entity affinities, or both.

Partalidou et al. [145] propose several layouts for structuring the input sequence of mentions and entities for the Transformer model. We achieved the best results with the mention layout

```
[CLS] [<type>] <mention label> [CTX] <mention context> [SEP]
```

and the entity layout

```
[CLS] [<type>] <entity title> [CTX] <entity description> [SEP]
```

where [CTX] is a special delimiter token and [<type>] is a placeholder for a special token of the mention type (POS-tag) or entity type (top-level type in the KG). We stick to Wu et al. [210] for optimisation and use in-batch (hard) negatives for the bi-encoder and bi-encoder-generated negatives for the cross-encoder.

8.4.2 Cluster Initialization

We follow Logan IV et al. [116] and use a greedy nearest-neighbour clustering to produce an initial mention clustering. Given the mention affinity threshold τ_m, the mentions $\mathcal{M}$ are grouped into clusters $\mathcal{C}$ so that two mentions $m, m' \in \mathcal{M}$ belong to the same cluster if $\phi(m, m') > \tau_m$.

Further, we assign entity candidates to the clusters using a threshold for entity affinity τ_e. For a cluster $C \in \mathcal{C}$ with mentions M_c, we select the known entities with the highest affinity to each cluster mention:

$$E_c^k = \bigcup_{m \in M_c} \{ \underset{e \in \mathcal{E}^k}{argmax}\, \phi(m, e) : \phi(m, e) > \tau_e \}. \tag{8.1}$$

In Fig. 8.2, the dotted lines represent affinities greater than the thresholds τ_m and τ_e, respectively. Cluster C_1 contains four loosely connected mentions with m_1 and m_2 directly connected to the entity candidate e_a. Either all four mentions refer to e_a as they are transitively connected, or some mentions refer to an entity in $\mathcal{E}^n$ (e.g., a situation like in Figs. 8.1a and 8.1b). Cluster C_2 contains several mentions with two known entity candidates e_b and e_c, making a trivial assignment of mentions to entities impossible. Finally, cluster C_3 contains three connected mentions without assigned entity candidates, most likely representing a NIL entity. Conflicts like the ones occurring in the former two clusters are resolved in the subsequent resolution phase.

8.4.3 Cluster Conflict Resolution

The objectives of the *Conflict Resolution Phase* are twofold: For every cluster $C \in \mathcal{C}$ we (1) find sub-clusters with $|E_c^k| = 1$ (c.f. C_{1B}, C_{2A}, and C_{2B} in Fig. 8.2), and (2) identify mentions in M_c that do not refer to any entity in E_c^k. For these, we create one or more sub-clusters representing the NIL entities E_c^n of C (c.f. C_{1A} and C_3 in Fig. 8.2).

For conflict resolution, we view a cluster $C \in \mathcal{C}$ as a graph G_c with $M_c \cup E_c^k$ as nodes and affinities above threshold as edges. To ensure objective (1), we assign every mention in a cluster to the candidate entity with the highest transitive affinity, defined as follows:

$$\phi^*(m, e) = \underset{m \sim e \in G_c}{max} \prod_{u,v}^{m \sim e} \phi(u, v) \tag{8.2}$$

with $m \sim e$ denoting a path from a mention m to an entity e in G_c and (u, v) a single edge. The rationale for this metric is to favour strong contextual similarity between mentions over the mediocre similarity between a mention and an entity. As the entity context comes from a different data corpus (i.e., information from a KG) than the mention context, it is more likely that the contexts for a mention and its linked entity are dissimilar than the contexts of two mentions linking to the same entity.

Example 8.4.1. Assignment of mentions to known entities

With affinities $\phi(m_6, e_b) = 0.9$, $\phi(m_6, m_7) = 0.9$, $\phi(m_7, e_c) = 0.8$, and paths $m_7 - m_6 - e_b$, $m_7 - e_c$ from Fig. 8.2, we find that $\phi^*(m_7, e_b) = 0.81 > \phi^*(m_7, e_c) = 0.8$, resulting in the assignment of m_7 to the cluster of e_b in spite of e_c being the most likely entity for m_7 w.r.t. ϕ.

To ensure objective (2), we introduce a threshold τ_a as a lower limit for the transitive affinity between a mention and an entity. We label mentions as NIL mentions if they do not have a transitive affinity higher than the threshold to any entity in E_c^k:

$$M_c^n = \{m \in M_c | \nexists e \in E_c^k : \phi^*(m, e) > \tau_a\} \tag{8.3}$$

From M_c^n, we produce one or more mention clusters similar to the initialization step in Section 8.4.2.

Example 8.4.2. Mention clusters for NIL entities

With $\tau_a = 0.75$, affinities $\phi(m_1, e_a) = 0.9$, $\phi(m_1, m_3) = 0.8$, $\phi(m_3, m_4) = 0.9$, and path $m_4 - m_3 - m_1 - e_a$ from Fig. 8.2, we find that $\phi^*(m_3, e_a) = 0.72 < \tau_a$ and $\phi^*(m_4, e_a) = 0.648 < \tau_a$. m_3 and m_4 are labelled as NIL mentions and, due to their direct connection, form the single cluster C_{1B}.

The function ϕ^* can be computed efficiently on a graph using Dijkstra's algorithm with $-log\phi$ as a function for edge weights. Edges are only inserted in the graph for $\phi > \tau_a$, avoiding undefined edge weights for $\phi = 0$.

8.5 Experiments

We first describe the datasets and experimental setup used to evaluate NASTyLinker. Then, we compare the performance of our approach with related NIL-aware clustering approaches on the NILK dataset [89] and analyze its potential to scale. Finally, we report on the application of NASTyLinker for KGP by linking entities in Wikipedia listings.

Dataset		$\lvert\mathcal{M}^k\rvert$	$\lvert\mathcal{M}^n\rvert$	$\lvert\mathcal{E}^k\rvert$	$\lvert\mathcal{E}^n\rvert$
NILK	Training ($\mathcal{D}^N_{train}$)	85,052,764	1,327,039	3,382,497	282,210
	Validation ($\mathcal{D}^N_{val}$)	10,525,107	162,948	422,812	35,276
	Test ($\mathcal{D}^N_{test}$)	10,451,126	162,497	422,815	35,279
LISTING	Training ($\mathcal{D}^L_{train}$)	11,690,019	6,760,273	3,073,238	?
	Validation ($\mathcal{D}^L_{val}$)	3,882,641	2,272,941	1,695,156	?
	Test ($\mathcal{D}^L_{test}$)	3,884,066	2,259,072	1,701,015	?
	Prediction ($\mathcal{D}^L_{pred}$)	18,658,271		?	?

Table 8.1: Mention and entity occurrences in the partitions of the datasets. NIL mention counts for $\mathcal{D}^L$ are estimated w.r.t. LCWA. The number of NIL entities $\mathcal{E}^n$ in the listings dataset is unknown. For $\mathcal{D}^L_{pred}$ a single mention count is displayed as we can't know whether a mention in $\mathcal{M}$ links to an entity in $\mathcal{E}^k$ or $\mathcal{E}^n$.

8.5.1 Datasets

NILK

NILK is a dataset explicitly created to evaluate EL for known and NIL entities. It uses Wikipedia as a text corpus and Wikidata [195] as reference KG. All entities in Wikidata up to 2017 are labelled as known entities, and entities added to Wikidata between 2017 and 2021 are labelled as NIL entities. Mention and entity counts of NILK are displayed in Table 8.1. About 1% of mentions in NILK are NIL mentions, and about 6% of entities are NIL entities. NIL entities are potentially slightly biased towards more popular entities, as the fact that they are present in Wikidata hints at a certain popularity, which may be higher than the popularity of an average NIL entity. Hence, the average number of mentions per NIL entity is quite high in this dataset: half of the entities are mentioned more than once, and more than 15% are even mentioned more than five times. Mention boundaries are already given, and the authors define partitions for training, validation, and testing, which are split in a zero-shot manner w.r.t. NIL entities. As mention context, the authors provide 500 characters before and after the actual mention occurrence in a Wikipedia page. As entity descriptions, we use Wikipedia abstracts.[2]

Wikipedia Listings

The LISTING dataset consists of the mentions of SEs extracted from enumerations and tables of Wikipedia as described in Chapter 7. Likewise, we use CaLiGraph as reference KG. Mention and entity statistics are given in Table 8.1. We partition the data into train, validation, and test while ensuring

[2]While there are entities in Wikidata which do not have a Wikipedia page, this case does not occur in NILK by construction.

that listings on a page are all in the same split. Unlike NILK, the LISTING dataset does not contain explicit labels for NIL entities. Instead, we define NIL entities using the LCWA. Given a listing with multiple mentions, we only incorporate them into training or test data if at least one mention is linked to a known entity. Then, by LCWA, we assume that all mentions that can be linked are linked. All other mentions are assigned a new unique entity identifier. The prediction partition $\mathcal{D}^L_{pred}$, however, contains all mentions without a linked entity (i.e., they may link to a known or a NIL entity). We use the text of the listing item as mention context for the dataset, and we use Wikipedia abstracts as entity descriptions.

We have considered further datasets that were used for evaluation of NIL-aware approaches for evaluation (e.g., from challenges like TAC-KBP or Microposts [35]) but discarded them due to their small size or not being free to use.

8.5.2 Metrics

Classification Metrics

We compute precision, recall, and F1 score as well as aggregations of the metrics on the instance level (micro average). As the evaluated approaches are unaware of the true NIL entities, they assign cluster identifiers to (what they assume to be) NIL mentions. To compute the classification metrics, it is necessary to map the cluster identifiers to actual NIL entities. Kassner et al. [94] allows the assignment of multiple cluster identifiers to the same NIL entity. This assumption would yield overly optimistic results. Instead, we only allow one-to-one mappings between cluster identifiers and NIL entities. Finding an optimal assignment for this scenario is equivalent to solving the linear sum assignment problem [3], for which efficient algorithms exist.

Clustering Metrics

Following related approaches [1, 94], we additionally provide Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) as clustering metrics for the comparison of the approaches to settings where no gold labels of NIL entities may be available.[3] For known entities, however, the classification metrics will most likely be more expressive than the clustering metrics as the latter treat multiple clusters with the same known entity as their label still as separate clusters.

[3]We implement further clustering metrics (B-Cubed+, CEAF, MUC) but do not list them as they are similar to or adaptations of the classification metrics.

8.5.3 Evaluated Approaches

EL Model

We compute inter-mention and mention-entity affinities with a bi-encoder similar to BLINK [210]. As the reranking of bi-encoder results with a cross-encoder is costly, we evaluate different scenarios where the cross-encoder is omitted (*No Reranking*), applied to inter-mention affinities only (*Mention Reranking*), applied to mention-entity affinities only (*Entity Reranking*), or applied to both (*Full Reranking*). We use the Sentence-BERT implementation of the bi-encoder and cross-encoder [162] with *all-MiniLM-L12-v2* and *distilbert-base-cased* as respective base models. The base models are fine-tuned for at most one million steps on the training partitions of the datasets. Longer fine-tuning did not yield substantial improvements. We use a batch size of 256 for the bi-encoder and 128 for the cross-encoder. For efficient retrieval of candidates from the bi-encoder, we apply approximate nearest-neighbour search with hnswlib [121].

We use the plain bi-encoder and cross-encoder predictions of the EL model as baselines. Additionally, we evaluate a trivial *Exact Match* approach, where we link a mention to an entity if their textual representations match exactly.[4] The more popular entity (w.r.t. ingoing and outgoing links in the KG) is selected in case of multiple matches. Naturally, this approach cannot handle NIL entities.

Clustering Approaches

Apart from the NASTyLinker clustering as described in Section 8.4, we apply the clustering approaches of Kassner et al. [94] and Agarwal et al. [1] for comparison.[5] The clustering approach of Kassner et al., which we call *Majority Clustering*, applies a greedy clustering and assigns a known entity e to a cluster if at least 70% of mentions in the cluster have the highest affinity to e. Similarly to NASTyLinker, they use hyperparameters as thresholds for minimum inter-mention and mention-entity affinities.

The clustering approach of Agarwal et al., which we call *Bottom-Up Clustering*, starts with an empty graph and iteratively adds the edge with the highest affinity, as long as it does not violate the constraint of a cluster having at most one entity. They use a single hyperparameter as a threshold for the minimum affinity of an edge, be it inter-mention or mention-entity.

[4] We apply simple preprocessing like lower-casing and removing special characters.

[5] We tried to compare with the full approach of Agarwal et al., but they do not provide any code, and our efforts to re-implement it did not yield improved results.

Approach		Known			NIL			Micro		
		F1	NMI	ARI	F1	NMI	ARI	F1	NMI	ARI
No Clustering	Exact Match	79.5	—	—	0.0	—	—	78.1	—	—
	Bi-Encoder	80.8	—	—	0.0	—	—	79.1	—	—
	Cross-Encoder	89.0	—	—	0.0	—	—	87.1	—	—
Clustering & No Reranking	Bottom-Up	64.6	99.0	97.5	41.6	94.8	81.8	64.1	96.8	93.5
	Majority	59.4	99.3	98.0	49.8	92.7	82.7	59.2	96.6	**94.6**
	NASTyLinker	76.8	98.6	95.3	40.8	**95.2**	76.8	76.0	97.3	90.3
Clustering & Mention Reranking	Bottom-Up	65.7	97.1	98.9	41.5	94.6	10.0	65.1	96.0	66.0
	Majority	66.6	92.4	74.8	44.0	94.4	73.2	66.1	92.4	70.4
	NASTyLinker	74.2	99.0	96.6	39.2	85.6	16.5	73.5	95.5	81.6
Clustering & Entity Reranking	Bottom-Up	89.0	99.3	96.2	41.6	94.1	58.0	87.9	98.2	92.6
	Majority	74.2	99.1	**99.3**	**54.1**	89.3	**92.5**	73.7	96.6	**94.6**
	NASTyLinker	**90.4**	99.3	95.5	43.7	94.6	85.3	**89.4**	**98.5**	84.1
Clustering & Full Reranking	Bottom-Up	84.2	**99.6**	98.9	41.8	84.6	3.2	83.3	96.2	65.5
	Majority	80.3	95.1	95.9	51.7	90.0	39.2	79.6	93.9	70.4
	NASTyLinker	87.9	99.5	99.2	42.5	87.6	33.6	86.9	97.4	71.7

Table 8.2: Results for the test partition $\mathcal{D}^N_{test}$ of the NILK dataset.

Hyperparameter Tuning

We select the hyperparameters of the EL model (*k, learning rate, warmup steps*) and the thresholds of all three clustering approaches w.r.t. micro F1 score on the validation partition of the datasets. For a fair comparison, we also test multiple values for the threshold for entity assignment of Majority Clustering, which in the original paper was fixed at 0.7.

Our experiments run on a machine with 96 CPUs, 1TB of RAM, and an NVIDIA RTX A6000 GPU with 48GB of RAM.

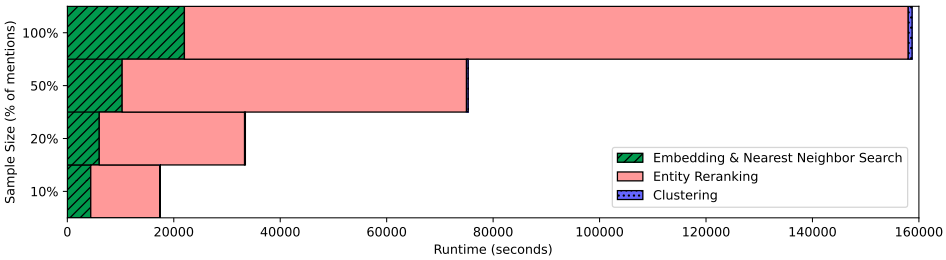
8.5.4 Entity Linking Performance

We tune hyperparameters by evaluating on $\mathcal{D}^N_{val}$. For the EL model, we use a *k* of 4, a learning rate of 2e-5, and no warmup steps. For τ_m, a value between 0.8 and 0.9 works best for all approaches. For τ_e, the best values revolve around 0.9 for NASTyLinker and Bottom-Up Clustering and around 0.8 for Majority Clustering. We use an affinity threshold τ_a of 0.75 for NASTyLinker and find that the 0.7 threshold of Majority Clustering produces the best results.

NILK Results

As shown in Table 8.2, we evaluate all clustering approaches on $\mathcal{D}^N_{test}$ in different reranking scenarios. We find Exact Match already to be a strong baseline for known entities with an F1 of 79.5%, which the Cross-Encoder outperforms by approximately 10%. Even without reranking, the three clustering approaches can achieve an F1-score between 40% and 50% for NIL entities. Overall, Majority Clustering is best suited to identify NIL entities. It

Figure 8.3: Runtime of NASTyLinker components for predictions on samples of $\mathcal{D}^N_{test}$.



is the only one to substantially benefit from reranking, increasing the F1-score by 10% when applying entity reranking. Especially for linking known entities, applying only entity reranking is the most favourable scenario, leading even to slight improvements over the baseline approaches that focus only on known entities.

As the reranking of mentions tends to decrease results while considerably increasing runtime, we omit mention reranking (and hence, full reranking) in experiments with Wikipedia listings. In the remaining scenarios, NASTyLinker finds the best balance between linking known entities and identifying NIL entities w.r.t. F1 score and NMI.

Runtime and Scalability

The fine-tuning of the bi-encoder and cross-encoder models took 2 and 12 hours, respectively. For prediction with a k of 4 on $\mathcal{D}^N_{test}$, the bi-encoder needed 6 hours. Reranking entity affinities with the cross-encoder took 38 hours. Clustering the results with any of the three approaches took 8 to 12 minutes.

In Fig. 8.3, we give an overview of the runtime of NASTyLinker components, compared over various sample sizes of $\mathcal{D}^N_{test}$. Overall, we can see that the total runtime scales linearly. With a smaller sample size, the computation of embeddings and nearest-neighbour search with the bi-encoder is responsible for a larger fraction of the total runtime. This is due to the large overhead of creating the index for the approximate nearest-neighbour search. Increasing sample size makes this factor less important for the overall runtime. In general, entity reranking is responsible for most of the total runtime.

The runtime of the clustering itself is responsible for approximately 1% of total runtime, and we do not expect it to increase substantially, as Dijkstra's algorithm has log-linear complexity and the size of mention clusters can be controlled by the threshold τ_m. Hence, the runtime of NASTyLinker is

expected to grow proportionally to the runtime of BLINK [210] for increasing sizes of datasets. If runtime is important, one should consider skipping entity reranking, as NASTyLinker still produces reasonable results when relying only on bi-encoder affinities.

8.5.5 Linking Entities in Wikipedia Listings

As the average mention context length in the LISTING dataset is lower than in NILK, fine-tuning the EL models took only 8 hours. Most of the hyperparameters chosen for NILK are also a reasonable choice for this dataset. For entity reranking, however, the approaches produce better results when the thresholds τ_m and τ_a slightly increase to 0.9 and 0.85.

Results on Test Partition

Linking results for $\mathcal{D}^L_{test}$ are provided in Table 8.3. As we rely on LCWA to label NIL mentions, we only know whether a mention is a NIL mention without knowing which ones refer to the same entity. Hence, we can only compute results for known entities and overall predictions. We assume that any prediction made for a NIL mention is incorrect for the latter. With this assumption, we are unable to produce realistic performance estimates. Still, we can see the impact of being NIL-aware (and, hence, make no prediction for NIL mentions) on the overall linking performance.

Due to their majority mechanism, Majority Clustering identifies known entities with high precision but at the cost of a reduced recall. Bottom-Up Clustering and NASTyLinker scores are comparable when considering known entities but diverge w.r.t. the micro average. NASTyLinker achieves the best micro F1 score with 86.7% in the entity reranking scenario. However, this has to be taken with a grain of salt as we do not know how many of the heuristically labelled NIL mentions refer to NIL entities and how many refer to known entities.

Knowledge Graph Population Statistics

The partition $\mathcal{D}^L_{pred}$ of the LISTING dataset contains only mentions for which we don't know whether they link to a known or a NIL entity. To make predictions for these mentions, we run the NASTyLinker approach on the whole LISTING corpus, i.e., on a total of 38 million mentions, as we need representations of all known entities for the clustering step. These mentions were extracted from 2.9 million listings on 1.4 million Wikipedia pages. As reference KG, we again use CaLiGraph so that the discovered entities can be integrated into the KG.

The total runtime was 62 hours, with 14 hours for the bi-encoder, 47 hours for the cross-encoder, and 45 minutes for the clustering. We find 13.4

Approach		Known			Micro		
		P	R	F1	P	R	F1
No Clustering	Exact Match	91.4	73.5	81.5	81.1	73.5	77.1
	Bi-Encoder	88.6	88.6	88.6	62.6	88.6	73.4
	Cross-Encoder	93.7	**93.8**	**93.8**	66.2	**93.8**	77.6
Clustering & No Reranking	Bottom-Up	89.7	84.9	87.2	63.9	84.9	72.9
	Majority	95.2	67.9	79.2	78.1	67.9	72.6
	NASTyLinker	90.6	78.5	84.1	70.7	78.5	74.4
Clustering & Entity Reranking	Bottom-Up	94.2	90.8	92.5	75.3	90.8	82.3
	Majority	**98.8**	76.2	86.0	**93.4**	76.2	83.9
	NASTyLinker	97.0	87.0	91.8	88.5	87.0	**87.7**

Table 8.3: Results for the test partition $\mathcal{D}^L_{test}$ of the LISTING dataset. No results for *NIL* are given because the real NIL entities $\mathcal{E}^n$ are unavailable in this dataset. For the micro average, we label every prediction made for a mention linked to an entity in $\mathcal{E}^n$ as incorrect.

million mentions (i.e., 70%) to be NIL mentions, which refer to 7.6 million NIL entities. The remaining 5.2 million mentions refer to 1.4 million entities already in CaLiGraph. Integrating the discovered NIL entities into CaLiGraph, we increase its entity count by 130%. Further, the discovered mentions for known entities can be used to enrich the representations of the entities in the KG through various KG completion methods (addressed in the subsequent chapter).

Qualitative Analysis

We manually inspected the results to evaluate the actual linking performance on the set of unlabeled mentions $\mathcal{D}^L_{pred}$. We randomly picked 100 mentions and 100 clusters[6] and identified, if incorrect, the type of error.[7] The results of this evaluation are given in Table 8.4. Overall, we find the outcome to agree with the results of NASTyLinker on $\mathcal{D}^L_{test}$. Hence, the approach produces highly accurate results, which we observed even for difficult cases. For example, the approach correctly created NIL entity clusters for the mention *North Course* referring to a racing horse (in pages *Appleton Stakes* and *Oceanport Stakes*), a golf course in Ontario, CA (in page *Tournament Players Club*), and a golf course in Florida, US (in page *Pete Dye*).

While the linking performance is quite consistent for mentions, the correctness of clusters for known entities is significantly lower than for NIL

[6]The sampling of clusters was stratified w.r.t. cluster size.

[7]We evaluated the linking and clustering decision w.r.t. the top-4 mention and entity candidates produced by the bi-encoder. Although recall@4 for the bi-encoder is 97%, some relevant candidates might have been missed.

Prediction	Mentions			Clusters		
	$\mathcal{E}^k$	$\mathcal{E}^n$	$\mathcal{E}$	$\mathcal{E}^k$	$\mathcal{E}^n$	$\mathcal{E}$
Correct	20	64	84	8	71	79
Incorrectly linked to NIL entity	3	—	3	1	—	1
Incorrectly linked to known entity	—	7	7	—	3	3
Not all mentions of entity in cluster	—	—	—	8	0	8
Mentions from multiple entities in cluster	—	—	—	1	4	5
Ignored (mention extracted incorrectly)	—	—	6	—	—	4
Total Count	23	71	94	18	78	96
Accuracy (%)	87.0	90.1	89.4	44.4	91.0	82.3

Table 8.4: Results of the manual evaluation of 100 clusters and 100 mentions. Columns group the results by actual entity type (known, NIL, overall), and rows group by prediction outcome. Accuracy values may deviate by $\pm 9.6\%$ for mentions and by $\pm 7.0\%$ for clusters (95% confidence).

entities.[8] This drop in performance is not due to NASTyLinker's inability to link to known entities correctly (as the accuracy of 87% on the mention-level shows). Instead, it can be attributed to the fact that clusters of known entities contain 3.8 mentions on average, while clusters of NIL entities contain 1.7 mentions on average. Hence, the likelihood of missing at least one mention is much higher, which is also the main error for known clusters.

Compared to the results on NILK, the linking accuracy for NIL mentions is much higher. We explain this with the different kinds of NIL entities in the two datasets. While an average NIL entity is mentioned 4.6 times in NILK, our results indicate that this number is approximately 1.7 for the LISTING dataset. The latter dataset may, hence, contain a lot of easy-to-link mentions by assigning them their own cluster.

8.6 Conclusion

With NASTyLinker, we introduced a NIL-aware EL approach capable of making high-quality predictions for known and NIL entities. In the practical setting of EL in Wikipedia listings, we show that our approach can populate a KG with many additional entities and enrich representations of existing entities. NASTyLinker is published as stand-alone version[9] and as part of the CaLiGraph extraction framework.[10] CaLiGraph version 3 contains all entities extracted from Wikipedia listings by NASTyLinker, increasing its entity count to 13.7 million.

[8] For the evaluation to be significant, we treat all clusters referring to the same known entity as a single cluster.

[9] `https://github.com/nheist/eswc2023-nastylinker`

[10] `https://github.com/nheist/CaLiGraph`

CHAPTER 9

Information Extraction from Co-Occurring Similar Entities

In this chapter, we explore how information extracted from similar entities that co-occur in structures like tables or enumerations can help to increase the coverage of KGs. In contrast to prevalent approaches, we do not focus on relationships within a listing (e.g., between two entities in a table row) but on the relationship between a listing's SEs and the context of the listing. To that end, we propose a descriptive rule-mining approach that uses distant supervision to derive rules for these relationships based on a listing's context. Extracted from a suitable data corpus, the rules can be used to extend a KG with novel entities and assertions. The contributions of this chapter are as follows:

- We formulate the task of IE from co-occurring similar entities in listings and show how to derive descriptive rules for listing characteristics based on the listing context.
- We present an approach that learns descriptive rules for listings in Wikipedia and is capable of extracting several millions of novel assertions for Wikipedia-based KGs.
- In our evaluation, we demonstrate the high quality of the extracted information and analyze the approach's shortcomings.

The work presented in this chapter is based on the following publication:

Nicolas Heist and Heiko Paulheim. Information Extraction From Co-Occurring Similar Entities. In *Proceedings of the Web Conference 2021* (WWW'21), pp. 3999-4009, Ljubljana, Slovenia, April 2021, ACM Press. [70]

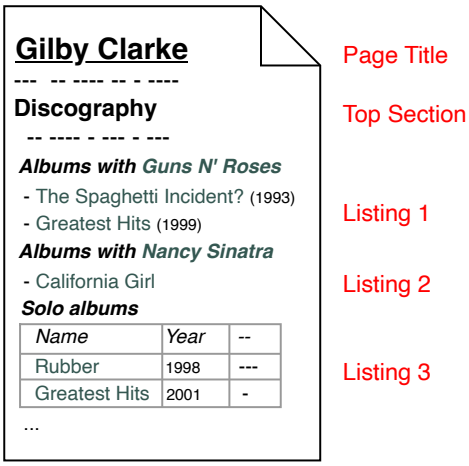


Figure 9.1: Simplified view on the listings of the Wikipedia page of Gilby Clarke. Entity mentions are highlighted in green.

9.1 Motivation

We again consider the example from Chapter 7, here shown in Fig. 9.1. The table below the *Solo albums* section (*Listing 3*) may be used to discover the publication years of albums (relation extraction) or discover additional unknown albums that are listed in further rows below *Rubber* and *Greatest Hits* (entity and type detection).

The focus of this chapter is broader with respect to two dimensions: First, we extract information from any kind of structure where similar entities co-occur. In Fig. 9.1, we would consider both tables and enumerations (e.g., *Listings 1* and *2*). Second, we consider only the SEs of listings. We aim to exploit the relationship between the SEs of a listing and the listing context. For Fig. 9.1, this means we extract that all SEs on the page's listings are albums with the artist *Gilby Clarke*, that *The Spaghetti Incident?* is an album by *Guns N' Roses*, and so on.

To that end, we propose to learn these characteristics of a listing with respect to the types and contextual relations of its SEs. In an ideal setting, we know the SEs of a listing, and we can retrieve all information about them from a KG – the characteristics of a listing are simply the types and relations that all SEs share. But uncertainty is introduced by several factors:

- SEs can only be determined heuristically. In Chapter 7, we achieved a precision of 70% to 80% for recognising SEs in Wikipedia listings.
- Cross-domain KGs are incomplete. According to the OWA, the absence of a fact in a KG does not imply its incorrectness.
- Web tables have a median of 6 rows,[1] and Wikipedia listings have

[1]According to WDC Table Corpus 2015: `http://webdatacommons.org/webtables/`.

a median of 8 rows. Consequently, many listings only have a small number of SEs from which the characteristics can be inferred.

As a result, considering each listing in isolation either leads to a substantial loss of information (as listings with insufficient background information are disregarded) or to a high generalization error (as decisions are made based on insufficient background information).

We observe that the context of a listing is often a strong indicator of its characteristics. In Fig. 9.1, the title of the top section *Discography* indicates that its listings contain some musical works, and the section title *Albums with Guns N' Roses* provides more detailed information. Our second observation is that these patterns repeat when looking at a coherent data corpus. The Wikipedia page of *Axl Rose*,[2] for example, contains the same constellation of sections.

Considering listing characteristics with respect to their context can thus yield more general insights than considering every listing in isolation. For example, the musical works of many artists in Wikipedia are listed under the top section *Discography*. Hence, we could learn the axioms

$$\exists topSection.\{\text{"Discography"}\} \sqsubseteq \texttt{MusicalWork} \tag{9.1}$$

and

$$\exists topSection.\{\text{"Discography"}\} \sqsubseteq \exists artist.\{<PageEntity>\} \tag{9.2}$$

which are then applicable to any listing with the top section *Discography* in Wikipedia.

The rest of this chapter is structured as follows. Section 9.2 gives an overview of the state of the art. Section 9.3 formally defines the task and explains our rule-learning approach. In Section 9.4, we present a use case for our approach by applying it to Wikipedia listings. Then, we evaluate individual parts of our approach in Section 9.5.

9.2 Related Work

The work presented in this chapter is a flavour of KGE, more precisely, of adding new assertions to a KG [147]. We use rules based on page context to infer facts about co-occurring entities. In particular, we focus on the co-occurrence of entities within document listings, where co-occurrence refers to proximity in page layout. Hence, in this section, we discuss related works w.r.t. KG completion from listings, exploitation of listing context, as well as rule learning for KGs.

[2] `https://en.wikipedia.org/wiki/Axl_Rose`

9.2.1 Knowledge Graph Completion from Listings

KG completion using web table information has been an active research area in the last several years. Muñoz et al. [133] described the potential of Wikipedia's tables in 2013 and presented an approach for extracting 8 million assertions from one million Wikipedia tables in 2014 [134]. In 2016, Ritze et al. [169] profiled the potential of web tables in the WDC Web Table Corpus. Using the T2K Match framework, they match web tables to DBpedia and find that the best results for the extraction of new facts can be achieved using knowledge-based trust [38] (i.e., judging the quality of a set of extracted triples by their overlap with the KG). Zhang et al. [217] present an approach for the detection of novel entities in tables. They first exploit lexical and semantic similarity for entity linking and column heading property matching. In the second step, they use the output to detect novel entities in table columns. Oulabi and Bizer [143] tackle the same problem for Wikipedia tables with a bootstrapping approach based on expert-defined rules. Macdonald and Barbosa [118] extract new facts from Wikipedia tables to extend Freebase. With an LSTM that uses contextual information from the table, they extract new facts for 28 relations.

Lists have only very sparsely been used for KG completion. Paulheim and Ponzetto [150] frame the general potential of list pages as a source of knowledge in Wikipedia. They propose to use a combination of statistical and NLP methods to extract knowledge and show that, by applying them to a single list page, they are able to extract a thousand new statements.

Compared to all previously mentioned approaches, we take an abstract view of listings by considering only their SEs. This provides the advantage that rules can be learned from and applied to arbitrary listings. In addition to that, we not only discover novel entities but also discover relations between those entities and the page subject.

In Chapter 5, we have already presented an approach for retrieving novel entities and extracting facts from Wikipedia list pages. Here, we reuse this approach for identifying SEs and use it as a baseline in our experiments. As it only works for list pages in Wikipedia, we extend it to arbitrary pages with a simple frequency-based approach.[3]

9.2.2 Exploiting the Context of Listings

As tables are the more actively researched type of listings, we focus here on the types of context used when working with tables. The most obvious source of context is found directly on the page where the table is located. This page context is, for example, used by InfoGather [212] to detect possible synonyms in table headers for means of table matching.

[3]The approaches from Chapters 7 and 8 are not considered here as the original work presented in this chapter was conducted earlier.

Zhang [218] distinguishes between "in-table" features like the table header and "out-table" features like captions, page titles, and text of surrounding paragraphs. With both kinds of features, they perform entity disambiguation against Freebase.

The previously mentioned approach of Macdonald and Barbosa [118] focuses on tables in Wikipedia and hence uses specific context features like section titles, table headers and captions, and the text in the first paragraph of the table's section. Interestingly, they do not only discover relations between entities in the table but also between a table entity and the page subject.

MENTOR [26] leverages patterns occurring in headers of Wikipedia tables to consistently discover DBpedia relations. Lehmberg et al. [106] tackle the problem of small web tables with table stitching, i.e., they combine several small tables with a similar context (e.g., same page or domain and a matching schema) into one large table, making it easier to extract facts from it.

Apart from page context, many approaches use the context of entities in tables to improve extraction results. Zhang et al. [216] generate new sub-classes to a taxonomy for a set of entities. Therefore, they find the best-describing class using the context of the entities. In particular, they use the categories of the entities as well as the immediate context around the entities on the page. Another approach that uses entity categories as context is TableNet [46]. They leverage the context to find schematically similar or related tables for a given table in Wikipedia.

In our experiments with Wikipedia, we use section headers as page context and types in the KG as entity context. However, the definition of context in our approach is kept very generic on purpose. By doing that, we can incorporate additional context sources like section text or entity categories to improve extraction results. This, however, also comes with an increase in rule complexity and, consequently, run time.

9.2.3 Rule-based Knowledge Graph Completion

Rule-based KG completion approaches typically generate rules on the instance level (rules that add new assertions for individual instances) or on the schema level (rules that add additional schematic constraints).

AMIE+ [49] and AnyBURL [123] are instance-level rule learners inspired by Inductive Logic Programming (ILP). The former uses top-down, the latter bottom-up rule learning to generate rules in the fashion of $born(X, A) \wedge capital(A, Y) \implies citizen(X, Y)$.

DL-Learner [103] is an ILP-based approach at the schema level, finding description logic patterns for a set of instances. A related approach [194] uses statistical schema induction to derive additional schema constraints (e.g., range restrictions for predicates).

The above-mentioned approaches are merely *link prediction* approaches, i.e., they predict new relations between entities already contained in the

KG. The same holds for the omnipresent KG embedding approaches [196]. Such approaches are very productive when enough training data is available, and they provide exact results, especially when both positive and negative examples are given. In the setting of this chapter, we are working with (more or less) noisy external data.

With regard to instance versus schema level, our approach can be regarded as a hybrid approach that generates rules for sets of entities, which are, in turn, used to generate facts on an instance level. In this respect, our approach is similar to C-DF [211], which uses Wikipedia categories as an external data source to derive the characteristics of categories. To that end, they derive lexical patterns from category names and contained entities.

In this chapter, we apply rule learning to co-occurring entities in Wikipedia. While existing approaches have only considered explicit co-occurrence, i.e., categories or list pages, we go beyond the state of the art by learning rules for *arbitrary* listings in Wikipedia.

9.3 Information Extraction From Co-Occurrences

Given a data corpus $\mathcal{D}$ containing structures with co-occurring entities (e.g., listings in Wikipedia or a collection of spreadsheets) and a KG $\mathcal{K}$ containing a subset of these entities, the objective is to extract additional entities and assertions.

9.3.1 Task Formulation

For the formulation of the task, we use the KG $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{E}, \mathcal{L}, \mathcal{A})$ and its complete version $\mathcal{K}^*$ as defined in Section 2.2. The data corpus $\mathcal{D}$ contains a set of listings Φ, where each listing $\phi \in \Phi$ contains a number of SEs E_ϕ. Our task is to identify assertions that hold for all entities in E_ϕ. We distinguish between taxonomic and relational information expressed in $\mathcal{K}$.

The taxonomic information is a set of types that is shared by all SEs of a listing:

$$\mathcal{T}_\phi = \{t | t \in \mathcal{T}, \forall e \in E_\phi : (e, \texttt{rdf:type}, t) \in TA^*\}, \tag{9.3}$$

and the relational information is a set of relations to other entities shared by all SEs of a listing:[4]

$$\mathcal{R}_\phi = \{(p, o) | p \in \mathcal{P} \cup \mathcal{P}^{-1}, o \in \mathcal{E}, \forall e \in E_\phi : (e, p, o) \in RA^*\}. \tag{9.4}$$

[4]Here, the entities in E_ϕ may occur both in the subject and the object position. But for a more concise notation, we use only (p,o)-tuples and introduce the set of inverse predicates $\mathcal{P}^{-1}$ to express that SEs may also occur in object position. Note that this is only a notation – the inverse predicates do not have to exist in the schema.

From these characteristics of listings, we can derive all the *additional* type assertions

$$TA^+ = \bigcup_{\phi \in \Phi} \{(e, \texttt{rdf:type}, t) | e \in E_\phi, t \in \mathcal{T}_\phi\} \setminus TA \tag{9.5}$$

and *additional* relation assertions

$$RA^+ = \bigcup_{\phi \in \Phi} \{(e, p, o) | e \in E_\phi, (p, o) \in \mathcal{R}_\phi\} \setminus RA \tag{9.6}$$

that are encoded in Φ and missing in $\mathcal{K}$. Furthermore, TA^+ and RA^+ can contain additional entities that are not yet contained in $\mathcal{K}$, as there is no restriction for SEs of Φ to be part of $\mathcal{K}$.

For the sake of readability, we will only describe the case of $\mathcal{R}_\phi$ for the remainder of this section as $\mathcal{T}_\phi$ is, notation-wise, a special case of $\mathcal{R}_\phi$ with $p = \texttt{rdf:type}$ and $o \in \mathcal{T}$.

9.3.2 Learning Descriptive Rules for Listings

Due to the incompleteness of $\mathcal{K}$, it is impossible to derive the exact set of relations $\mathcal{R}_\phi$ for every listing in Φ. Hence, our goal is to derive an approximate version $\hat{\mathcal{R}}_\phi$ by using ϕ and the knowledge about E_ϕ in $\mathcal{K}$.

Similar to the rule learner AMIE+ [49], we use the LCWA to generate negative evidence. Following the LCWA, we use the *count* of entities with a specific predicate-object combination in a set of entities E

$$count(E, p, o) = |\{e | e \in E, \exists o : (e, p, o) \in \mathcal{A}\}| \tag{9.7}$$

and the *count* of entities having predicate p with an arbitrary object

$$count(E, p) = |\{e | e \in E, \exists o' : (e, p, o') \in \mathcal{A}\}| \tag{9.8}$$

to compute a maximum-likelihood-based frequency of a specific predicate-object combination occurring in E:

$$freq(E, p, o) = \frac{count(E, p, o)}{count(E, p)}. \tag{9.9}$$

From Eq. (9.9), we first derive a naive approximation of a listing's relations by including all relations with a frequency above a defined threshold τ_{freq}:

$$\hat{\mathcal{R}}_\phi^{freq} = \{(p, o) | (p, o) \in \mathcal{R}, freq(E_\phi, p, o) > \tau_{freq}\}. \tag{9.10}$$

As argued in Section 9.1, we improve this naive frequency-based approximation by learning more general patterns that describe the characteristics

Listing	ζ	T^F	R^F
ϕ_1	(1 0 1 ... 1)	(0.2 0.9 0.0 ... 0.1)	(0.9 0.1 0.0 ... 0.1)
ϕ_2	(0 1 1 ... 0)	(0.0 0.2 0.0 ... 0.9)	(0.0 0.0 0.0 ... 0.2)
ϕ_3	(0 0 0 ... 0)	(0.7 0.7 0.0 ... 0.0)	(0.0 0.0 0.0 ... 0.4)
		...	
ϕ_{n-1}	(1 0 0 ... 1)	(0.8 0.9 0.0 ... 0.0)	(0.0 0.9 0.0 ... 0.0)
ϕ_n	(1 0 0 ... 1)	(0.7 1.0 0.0 ... 0.3)	(0.0 0.0 0.8 ... 0.0)

Table 9.1: Exemplary context (ζ), type frequency (T^F), and relation frequency (R^F) vectors for a set of listings extracted from $\mathcal{D}$. While ζ is extracted directly from $\mathcal{D}$, T^F and R^F are retrieved via distant supervision from $\mathcal{K}$.

of listings using their context. The underlying hypothesis is as follows: The context ζ_ϕ of a listing ϕ in $\mathcal{D}$ contains such information about $\mathcal{R}_\phi$ that it can be used to find subsets of Φ with similar $\mathcal{R}$.

Let Table 9.1 contain the information about all listings in $\mathcal{D}$. A listing ϕ is defined by its context ζ_ϕ (which can, in theory, contain any information about ϕ, from the title of its section to an actual image of the listing), the type frequencies $(t_1, t_2, .., t_x) \in T^F_\phi$, and the relation frequencies $(r_1, r_2, .., r_y) \in R^F_\phi$. Listings ϕ_1, ϕ_{n-1}, and ϕ_n have overlapping context vectors. t_2 has a consistently high frequency over all three listings. It is thus a potential type characteristic for this kind of listing context. Furthermore, r_1 has a high frequency in ϕ_1, r_2 in ϕ_{n-1}, and r_3 in ϕ_n – if the three relations share the same predicate, they may all express a similar relation to an entity in their context (e.g., to the subject of the page).

In a concrete scenario, the context vector *(1 0 0 ... 1)* might indicate that the listing is located on the page of a musician under the section *Solo albums*. t_2 holds the frequency of the type *Album* in this listing and r_1 to r_3 describe the frequencies of the relations (*artist*, Gilby Clarke), (*artist*, Axl Rose), and (*artist*, Slash).

We formulate the task of discovering frequent co-occurrences of context elements and taxonomic and relational patterns as an association rule mining task for all listings in $\mathcal{D}$. Association rules, as introduced by Agrawal et al. [2], are simple implication patterns originally developed for large and sparse datasets like transaction databases of supermarket chains. To discover items that are frequently bought together, rules of the form $X \implies Y$ are produced, with X and Y being itemsets. In the KG context, they have been used, e.g., for enriching the schema of a KG [149, 194].

For our scenario, we need a mapping from a context vector $\zeta \in Z$ to a predicate-object tuple. Hence, we define a rule r, its antecedent r_a, and its

consequent r_c as follows:

$$r : r_a \in Z \implies r_c \in (\mathcal{P} \cup \mathcal{P}^{-1}) \times (\mathcal{T} \cup \mathcal{E} \cup \mathcal{X}). \tag{9.11}$$

As a rule should be able to imply relations to entities that vary with the context of a listing (e.g., to *Gilby Clarke* as the page's subject in Fig. 9.1), we introduce $\mathcal{X}$ as the set of placeholders for context entities (instead of *Gilby Clarke*, the object of the rule's consequent would be `<PageEntity>`).

We say a rule antecedent r_a matches a listing context ζ_ϕ (short: $r_a \simeq \zeta_\phi$) if the vector of ζ_ϕ is 1 when the vector of r_a is 1. In essence, ζ_ϕ must comprise r_a. Accordingly, we need to find a set of rules R so that for every listing ϕ, the set of approximate listing relations

$$\hat{\mathcal{R}}_\phi^{rule} = \bigcup_{r \in R} \{r_c | r_a \simeq \zeta_\phi\} \tag{9.12}$$

resembles the true relations $\mathcal{R}_\phi$ as closely as possible.

Considering all the listings in Fig. 9.1, their $\hat{\mathcal{R}}_\phi^{rule}$ should, among others, contain the rules[5,6]

$$topSection(\text{"Discography"}) \implies (type, \texttt{MusicalWork}) \tag{9.13}$$

and

$$topSection(\text{"Discography"}) \implies (artist, <PageEntity>). \tag{9.14}$$

It is important to note that these rules can be derived from listings with differing context vectors. All listings only have to have in common that their top section has the title *Discography* and that the contained entities are of the type `MusicalWork` with the page entity as an artist. Still, the individual listings may, for example, occur in sections with different titles.

9.3.3 Quality Metrics for Rules

In plain association rule mining, two metrics are typically considered for judging the quality of a rule $X \implies Y$: the support of the rule antecedent (how often does X occur in the dataset) and the confidence of the rule (how often does $X \cup Y$ occur in relation to X).

Transferring the support metric to our task, we count the absolute frequency of a particular context occurring in Φ. Let $\Phi_{r_a} = \{\phi | \phi \in \Phi, r_a \simeq \zeta_\phi\}$, then we define the support of the rule antecedent r_a as

$$supp(r_a) = |\Phi_{r_a}|. \tag{9.15}$$

[5]Note that Eqs. (9.1) and (9.2) are the axiom equivalents of Eqs. (9.13) and (9.14). For better readability, we use the description logics notation of Eqs. (9.1) and (9.2) from here on.

[6]Instead of a binary vector, we use a more expressive notation for the listing context in our examples. The notations are trivially convertible by one-hot-encoding.

Due to the incompleteness of $\mathcal{K}$, the values of Y are, in our case, no definitive items but maximum-likelihood estimates of types and relations. With respect to these estimates, a good rule has to fulfil two criteria: it has to be correct (i.e., frequent with respect to all SEs of the covered listings), and it has to be consistent (i.e., consistently correct over all the covered listings).

We define the correctness, or confidence, of a rule as the frequency of the rule consequent for all SEs of a rule's covered listings:

$$conf(r) = \frac{\sum_{\phi \in \Phi_{r_a}} count(E_\phi, p_{r_c}, o_{r_c})}{\sum_{\phi \in \Phi_{r_a}} count(E_\phi, p_{r_c})}, \tag{9.16}$$

and we define the consistency of a rule using the mean absolute deviation of an individual listing's confidence to the overall confidence of the rule:

$$cons(r) = 1 - \frac{\sum_{\phi \in \Phi_{r_a}} |freq(E_\phi, p_{r_c}, o_{r_c}) - conf(r)|}{supp(r_a)}. \tag{9.17}$$

While a high confidence ensures that the overall assertions generated by the rule are correct, a high consistency ensures that a few listings with many SEs do not outvote the remaining covered listings.

To select an appropriate set of rules R from all the candidate rules R^* in the search space, we have to pick reasonable thresholds for the minimum support (τ_{supp}), the minimum confidence (τ_{conf}), and the minimum consistency (τ_{cons}). By applying these thresholds, we find our final set of descriptive rules

$$R = \{r | r \in R^*, supp(r_a) > \tau_{supp} \wedge conf(r) > \tau_{conf} \wedge cons(r) > \tau_{cons}\}. \tag{9.18}$$

Typically, the choice of these thresholds is strongly influenced by the nature of the dataset $\mathcal{D}$ and the extraction goal (correctness versus coverage).

9.4 Exploiting Co-Occurrences in Wikipedia

Wikipedia is a rich source of listings, both in dedicated list pages and in sections of article pages. Hence, we use it as a data corpus for our experiments. In Section 9.6, we discuss other appropriate corpora for our approach.

Due to its structured and encyclopedic nature, Wikipedia is a perfect application scenario for our approach. We can exploit the structure by building very expressive context vectors. This positively influences the quality of extraction results. Still, the definition of the context vector is kept abstract to make the approach applicable to other web resources as well. However, an empirical evaluation of the practicability or performance of the approach for resources outside of the encyclopedic domain is out of the scope of this chapter.

9.4.1 Approach Overview

Fig. 9.2 gives an overview of our extraction approach. The input of the approach is a dump of Wikipedia together with an associated KG. In the *Subject Entity Discovery* phase, listings and their context are extracted from the Wikipedia dump and SEs are identified (Section 9.4.3). Subsequently, the existing information in the KG is used to mine descriptive rules from the extracted listings (Section 9.4.4). Finally, the rules are applied to all the listings in Wikipedia in order to extract new type and relation assertions (Section 9.4.5).

9.4.2 Data Corpus

The analyses and experiments in this chapter are based on *Wikipedia2016* (cf. Section 2.3.4), and we use versions of DBpedia and CaLiGraph that are based on this dump. In this version, Wikipedia contains 6.9M articles, 2.4M of which contain listings with at least two rows.[7] In total, there are 5.1M listings with a row count median of 8, mean of 21.9, and standard deviation of 76.8. Of these listings, 1.1M are tables, and 4.0M are enumerations.

9.4.3 Subject Entity Discovery

Entity Tagging

Apart from the already tagged entities via blue and red links, we have to ensure that any other named entity in listings and their context is also identified. This is done in two steps:[8]

In the first step, we expand an article's blue and red links. If a piece of text is linked to another article, we ensure that every occurrence of that piece of text is linked to the other article. This is necessary as, by convention, other articles are only linked at their first occurrence in the text.[9]

In the second step, we use a named entity tagger to identify additional named entities in listings. To that end, we use a state-of-the-art entity tagger from spaCy.[10] This tagger is trained on the OntoNotes5[11] corpus, and thus not specifically trained to identify named entities in short text snippets as they occur in listings. Therefore, we specialize the tagger by providing it with Wikipedia listings as additional training data with blue links as positive examples. In detail, the tagger is specialized as follows:

[7] Markup is parsed with WikiTextParser: `https://github.com/5j9/wikitextparser`.

[8] As mentioned before, the work in this chapter has been conducted before Chapters 7 and 8. Hence, we use a comparably naive method to identify SEs here. However, the final version of CaLiGraph presented in Chapter 11 uses the new approaches.

[9] `https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking#Duplicate_and_repeat_links`

[10] `https://spacy.io`

[11] `https://catalog.ldc.upenn.edu/LDC2013T19`

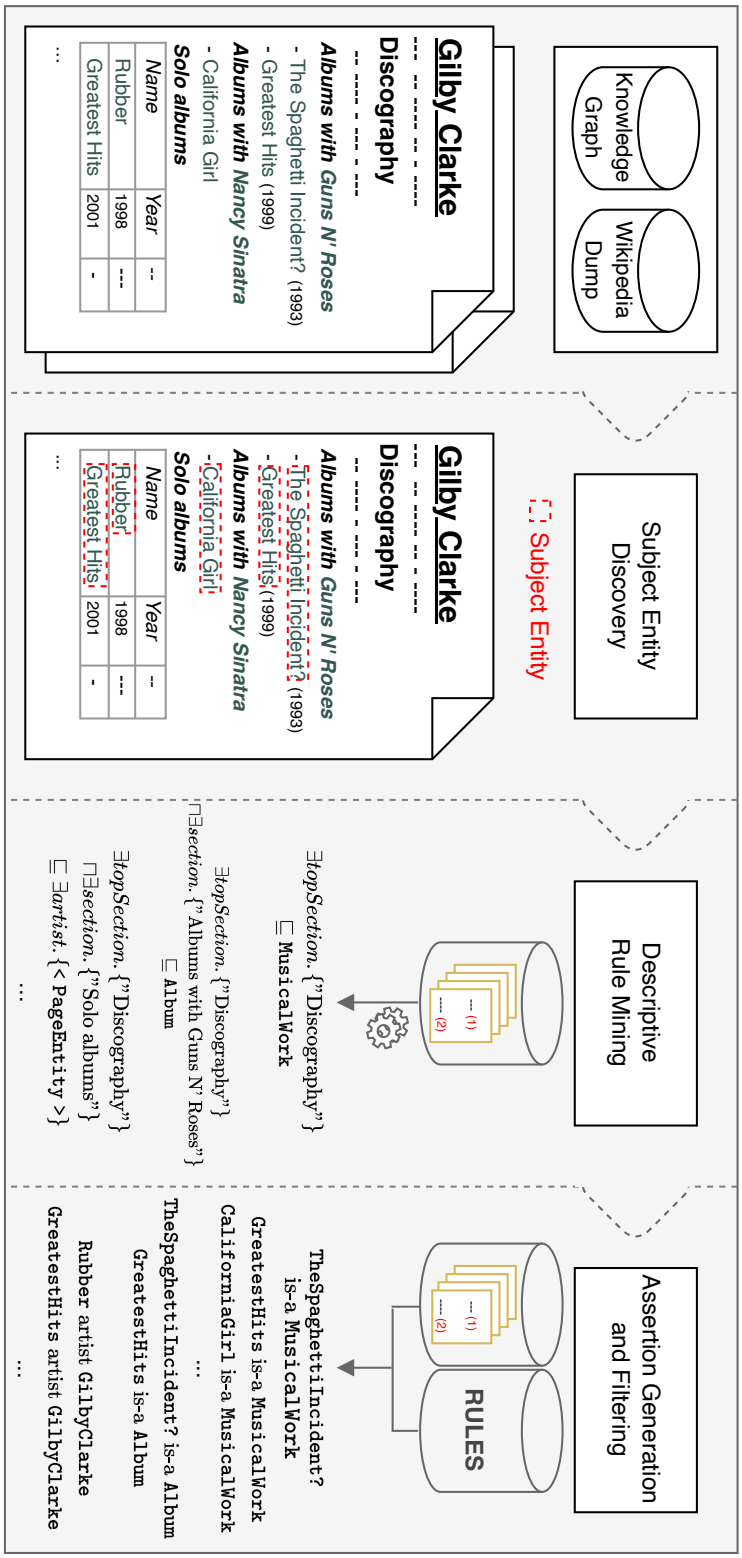


Figure 9.2: An overview of the approach with exemplary outputs of the individual phases.

- We retrieve all listings in Wikipedia list pages as training data.
- We apply the plain spaCy entity tagger to the listings to get named entity tags for all mentioned entities.
- To make these tags more consistent, we use information from DBpedia about the tagged entities: We look at the distribution of named entity tags over entities with respect to their DBpedia types and take the majority vote. For example, if 80% of entities with the DBpedia type `Person` are annotated with the tag *PERSON*, we use *PERSON* as the label for all these entities.
- Using these consistently named entity tags for blue-link entities, we specialize the spaCy tagger.

Subject Entity Classification

We apply the approach from Chapter 5 for identifying SEs in listings. In short, we use lexical, positional, and statistical features to classify entities as subject or non-subject entities (see Section 9.2.1 for more details). Despite being developed only for listings in list pages, the classifier is applicable to any kind of listing in Wikipedia. A disadvantage of this broader application is that the classifier is not trained to ignore listings used for organisational or design purposes (e.g., summaries or timelines). These have to be filtered out in the subsequent stages.

Results

After expanding all the blue and red links on the pages, the dataset contains 5.1M listings with 60.1M entity mentions. The named entity tagger identifies 51.6M additional entity mentions.

Of all the entity mentions, we classify 25.8M as SEs. Those occur in 2.5M listings of 1.3M pages. This results in a mean of 10.5 and a median of 4 SEs per listing with a standard deviation of 49.8.

9.4.4 Descriptive Rule Mining

Describing Listings

The listing context defines the search space for rule candidates. Thus, we choose the context in such a way that it is expressive enough to be an appropriate indicator for $\mathcal{T}_\phi$ and $\mathcal{R}_\phi$ and concise enough to explore the complete search space without any additional heuristics.

We exploit the fact that Wikipedia articles of a certain type (e.g., musicians) mostly follow naming conventions for the sections of their articles (e.g., albums and songs are listed under the top section *Discography*). Further, we exploit the fact that the objects of the SEs' relations are usually either

the entity described by the article or an entity mentioned in a section title. We call these typical places for objects the relation *targets*. In Fig. 9.1, *Gilby Clarke* is an example of a *PageEntity* target, and *Guns N' Roses* as well as *Nancy Sinatra* are examples for *SectionEntity* targets. As a result, we use the type of the page entity, the top section title, and the section title as listing context.

Additionally, we use the type of entities mentioned in section titles. This enables learning more abstract rules, e.g., to distinguish between albums listed in a section describing a band:

$$\exists pageEntityType.\{\texttt{Person}\} \sqcap \exists topSection.\{\text{"Discography"}\} \\ \sqcap \exists sectionEntityType.\{\texttt{Band}\} \sqsubseteq \texttt{Album},$$

and songs listed in a section describing an album:

$$\exists pageEntityType.\{\texttt{Person}\} \sqcap \exists topSection.\{\text{"Discography"}\} \\ \sqcap \exists sectionEntityType.\{\texttt{Album}\} \sqsubseteq \texttt{Song}.$$

Threshold Selection

We want to pick the thresholds in such a way that we tolerate some errors and missing information in $\mathcal{K}$ but do not allow many over-generalized rules that create incorrect assertions. Our idea for a sensible threshold selection is based on two assumptions:

Assumption 1. Based on a maximum-likelihood estimation, rule confidence and consistency roughly order rules by the degree of prior knowledge we have about them.

Assumption 2. Assertions generated by over-generalized rules contain substantially more random noise than assertions generated by good rules.

Assumption 1 implies that the number of over-generalized rules increases with the decrease in confidence and consistency. As a consequence, assumption 2 implies that the amount of random noise increases with decreased confidence and consistency.

To measure the increase of noise in generated assertions, we implicitly rely on existing knowledge in $\mathcal{K}$ by using the named entity tags of SEs as a proxy. This works as follows: For a SE e that is contained in $\mathcal{K}$, we have its type information $\mathcal{T}_e$ from $\mathcal{K}$, and we have its named entity tag ψ_e from our named entity tagger. Going over all SEs of listings in Φ, we compute the probability of an entity with type t having the tag ψ by counting how often they co-occur:

$$tagprob(t, \psi) = \frac{|\{e | \exists \phi \in \Phi : e \in E_\phi \wedge t \in \mathcal{T}_e \wedge \psi = \psi_e\}|}{|\{e | \exists \phi \in \Phi : e \in E_\phi \wedge t \in \mathcal{T}_e\}|}. \tag{9.19}$$

Example 9.4.1. Tag probability computation

For the DBpedia type `Album`, we find the following tag probabilities: $WORK_OF_ART$: 0.49, ORG: 0.14, $PRODUCT$: 0.13, $PERSON$: 0.07. This shows that album titles are rather difficult to recognize. For the type `Person` and the tag $PERSON$, we find a high probability of 0.86.

We compute the tag-based probability for a set of assertions A by averaging over the tag probability produced by the individual assertions. To compute this metric, we compare the tag of the assertion's SE with some kind of type information about it. The type information is either the asserted type (in case of a type assertion) or the domain of the predicate[12] (in case of a relation assertion):

$$tagfit(A) = \begin{cases} \frac{\sum_{(s,p,o)\in A} tagprob(o,\psi_s)}{|A|} & \text{if } p = \texttt{rdf:type} \\ \frac{\sum_{(s,p,o)\in A} tagprob(domain_p,\psi_s)}{|A|} & \text{otherwise.} \end{cases} \tag{9.20}$$

While we do not expect the named entity tags to be perfect, our approach is based on the idea that the tags are consistent to a large extent. By comparing the *tagfit* of assertions produced by rules with varying levels of confidence and consistency, we expect to see a clear decline as soon as too many noisy assertions are added.

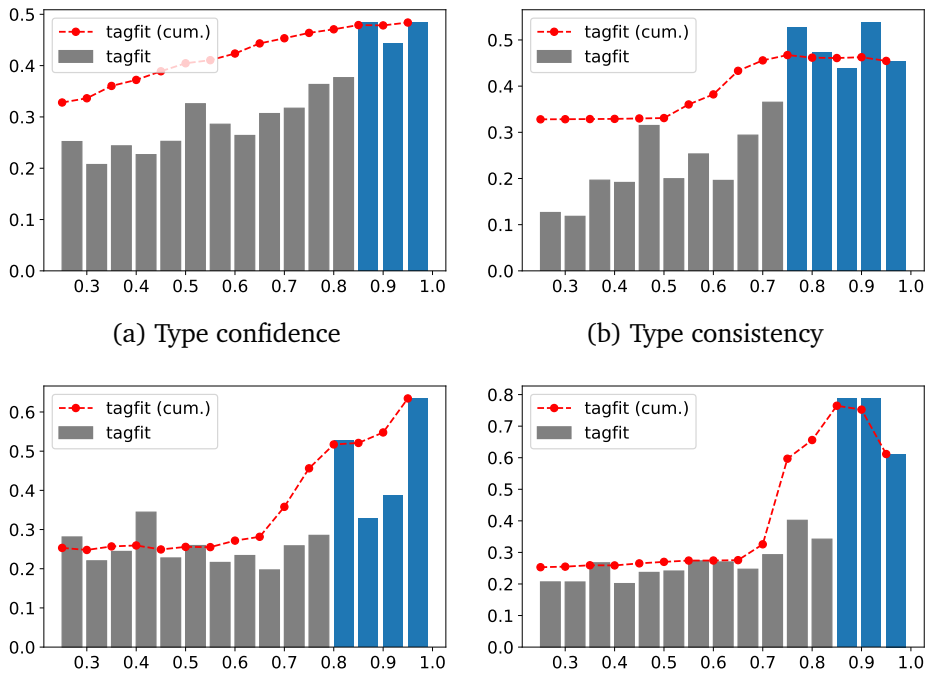
Results

Fig. 9.3 shows the *tagfit* for type and relation assertions generated with varying rule confidence and consistency levels. Our selection of thresholds is indicated by blue bars, i.e., we set the thresholds to the points where the *tagfit* has its steepest drop. The thresholds are picked conservatively to select only high-quality rules by selecting points before an accelerated decrease of cumulative *tagfit*. But more coverage-oriented selections are also possible. In Fig. 9.3d, for example, a threshold of *0.75* is also a valid option.

An analysis of rules with different levels of confidence and consistency has shown that minimum support for types is unnecessary. For relations, a support threshold of 2 is helpful to discard over-generalized rules. Further, we found that picking the thresholds independently from each other is acceptable, as the turning points for a given metric don't vary significantly when varying the remaining metrics.

Applying these thresholds, we find 5,294,921 type rules with 369,139 distinct contexts and 244,642 distinct types. Further, we find 3,028 relation rules with 2,602 distinct contexts and 516 distinct relations. 949 of the relation rules have the page entity as a target, and 2,079 have a section entity as a target.

[12] We use the domain of the predicate p as defined in $\mathcal{K}$. In the case of $p \in \mathcal{P}^{-1}$, we use the range of the original predicate.



(a) Type confidence (b) Type consistency

(c) Relation confidence (d) Relation consistency

Figure 9.3: *tagfit* of assertions generated from rules in a specified confidence or consistency interval. Bars show scores for a given interval (e.g., *(0.75,0.80]*); lines show cumulative scores (e.g., *(0.75,1.00]*); blue bars indicate the selected threshold.

Among those rules are straightforward ones like

$$\exists pageEntityType.\{\texttt{Person}\} \sqcap \exists topSection.\{\text{"Acting filmography"}\} \sqsubseteq \exists actor.\{<PageEntity>\},$$

and more specific ones like

$$\exists pageEntityType.\{\texttt{Location}\} \sqcap \exists topSection.\{\text{"Media"}\} \sqcap \exists section.\{\text{"Newspapers"}\} \sqsubseteq \texttt{Periodical_literature}.$$

9.4.5 Assertion Generation and Filtering

Assertion Generation

We apply the rules selected in the previous section to the complete dataset of listings to generate type and relation assertions. Subsequently, we remove any duplicate assertions and assertions that already exist in $\mathcal{K}$.

Tag-based Filtering

To get rid of errors introduced during the extraction process (e.g., due to incorrectly extracted SEs or incorrect rules), we employ a final filtering step for the generated assertions: every assertion producing a $tagprob \leq \frac{1}{3}$ is discarded. The rationale behind the threshold is as follows: Types typically have one and sometimes two corresponding named entity tags (e.g., the tag *PERSON* for the DBpedia type `Person`, or the tags *ORG* and *FAC* for the type `School`). As tag probabilities are relative frequencies, we make sure that, with a threshold of $\frac{1}{3}$, at most two tags are accepted for any given type.

For the tag probabilities of type `Album` from Section 9.4.4, the only valid tag is *WORK_OF_ART*. As a consequence, any assertions of the form $(s, rdf{:}type, \texttt{Album})$ with s having a tag other than *WORK_OF_ART* are discarded.

Results

Table 9.2 shows the number of generated type and relation assertions before and after the tag-based filtering. The number of inferred types is listed separately for DBpedia and CaLiGraph. For relations, we show two kinds: The entry *Relations* lists the number of extracted assertions from rules. DBpedia and CaLiGraph share the same predicates, so these assertions apply to both graphs. Furthermore, as *Relations (via CaLiGraph)*, we list the number of relations that can be inferred from the extracted CaLiGraph types via restrictions in the CaLiGraph ontology. CaLiGraph contains more than 300K restrictions that imply a relation based on a certain type. For example, the ontology contains the value restriction

$$\texttt{Pop_rock_song} \sqsubseteq \exists genre.\{\text{Pop music}\}.$$

As we extract the type `Pop_rock_song` for the Beach Boys song *At My Window*, we infer the fact $(\text{At My Window}, genre, \text{Pop music})$.

For CaLiGraph, we find assertions for 3.5M distinct SEs, with 3M not contained in the graph. We find assertions for 3.1M distinct SEs for DBpedia, with 2.9M of them not contained. To estimate the true number of novel entities, we rely on our analysis in Chapter 5, where we analyzed the overlap for red links in list pages. There, we estimate an overlap factor of 1.07, which would reduce the number of actual novel entities to roughly 2.8M for CaLiGraph and 2.7M for DBpedia when applied to our scenario. In relation to the current size of those graphs, this would be an increase of up to 38% and 54%, respectively.

Assertion Type	Raw	Filtered
Types (DBpedia)	11,459,047	7,721,039
Types (CaLiGraph)	47,249,624	29,128,677
Relations	732,820	542,018
Relations (via CaLiGraph)	1,381,075	796,910

Table 9.2: Number of generated assertions after removing existing assertions (Raw), and after applying tag-based filtering (Filtered).

Assertion Type	#Dataset	#Samples	Correct [%]
Types (DBpedia)			
frequency-based	6,680,565	414	91.55 ± 2.68
rule-based	7,721,039	507	93.69 ± 2.12
Types (CaLiGraph)			
frequency-based	26,676,191	2,000	89.40 ± 1.23
rule-based	29,128,677	2,000	91.95 ± 1.19
Relations			
frequency-based	392,673	1,000	93.80 ± 1.49
rule-based	542,018	1,000	95.90 ± 1.23

Table 9.3: Correctness of manually evaluated assertions.

9.5 Evaluation

Our performance evaluation judges the quality of generated assertions from our rule-based approach. As a baseline, we additionally evaluate assertions generated by the frequency-based approach (see Eq. (9.10)). For the latter, we use a threshold comparable to our rule-based approach (i.e., we set τ_{freq} to τ_{conf} and disregard listings with less than three SEs).

9.5.1 Evaluation Procedure

The evaluated assertions are created with a stratified random sampling strategy. The assertions are thus distributed proportionally over all page types (like `Person` or `Place`) and sampled randomly within these.

The authors perform the labelling of the assertions with the procedure as follows: For a given assertion, first, the page of the listing is inspected, then – if necessary and available – the page of the SE. If a decision cannot be made based on this information, a search engine is used to evaluate the assertion. Samples of the rule-based and frequency-based approaches are evaluated together and in random order to ensure objectivity.

Table 9.3 shows the results of the performance evaluation. We evaluated 2,000 examples per approach for types and 1,000 examples per approach

for relations. The taxonomy of CaLiGraph comprises the one of DBpedia. Thus, we evaluated the full sample for CaLiGraph types and reported the numbers for both graphs, which is why the sample size for DBpedia is lower. For relations, we only evaluate the ones generated directly from rules and not the ones inferred from CaLiGraph types, as the correctness of the inferred relations directly depends on the correctness of CaLiGraph types.

9.5.2 Type and Relation Extraction

The evaluation results in Table 9.3 show that the information extracted from listings in Wikipedia is of an overall high quality. The rule-based approach yields a larger number of assertions with higher correctness for both types and relations.

For both approaches, the correctness of the extracted assertions is substantially higher for DBpedia. The reason for that lies in the differing granularity of KG taxonomies. DBpedia has 764 different types, while CaLiGraph has 755,441, most of them being more specific extensions of DBpedia types. For example, DBpedia might describe a person as `Athlete`, while CaLiGraph describes it as `Olympic_field_hockey_player_of_South_Korea`. The average depth of predicted types is 2.06 for the former and 3.32 for the latter.

While the asserted types are very diverse (the most predicted type is `Agent` with 7.5%), asserted relations are dominated by the predicate *genus* with 69.8% followed by *isPartOf* (4.4%) and *artist* (3.2%). This divergence cannot be explained with a different coverage: In DBpedia, 72% of entities with type `Species` have a *genus*, and 69% of entities with type `MusicalWork` have an *artist*. However, we identify two other influencing factors: Wikipedia has very specific guidelines for editing species, especially with regard to standardization and formatting rules.[13] In addition to that, the *genus* relation is functional and hence trivially fulfilling the LCWA. As our approach strongly relies on this assumption and potentially inhibits the mining of practical rules for non-functional predicates (like, for example, for *artist*), we plan to investigate this relationship further.

The inferred relations from CaLiGraph types are not evaluated explicitly. However, based on the correctness of restrictions in CaLiGraph that we evaluated to be 95.6% in Chapter 6 and from the correctness of type assertions, we estimate the correctness of the resulting relation assertions to be around 85.5% for the frequency-based and around 87.9% for the rule-based approach.

[13] `https://species.wikimedia.org/wiki/Help:General_Wikispecies`

9.5.3 Novel Entities

For CaLiGraph, the frequency-based approach finds assertions for 2.5M distinct SEs (2.1M of them, novel). While the rule-based approach finds 9% more assertions, its assertions are distributed over 40% more entities (and over 43% more novel entities). This demonstrates the capabilities of the rule-based approach of applying contextual patterns to environments where information about entities is sparse.

Further, we analyzed the portion of evaluated samples that applies to novel entities and found that the correctness of these statements is slightly better (between 0.1% and 0.6%) than the overall correctness. Including CaLiGraph types, we find an average of 9.03 assertions per novel entity, with a median of 7. This is, again, due to the very fine-grained type system of CaLiGraph. For example, for the rapper *Dizzle Don*, which is a novel entity, we find eight types (from `Agent` over `Musician` to `American_rapper`) and four relations: (*occupation*, Singing), (*occupation*, Rapping), (*birthPlace*, United States), and (*genre*, Hip hop music).

9.5.4 Error Analysis

With Table 9.4, we analyse error type frequencies for the rule-based approach on the basis of the evaluated sample. (1) is caused by the entity linker, mostly due to incorrect entity borders. For example, the tagger identifies only a part of an album title. (2) is caused by errors in the SE identification approach, e.g., when the approach identifies the wrong column of a table as the one that holds SEs. (3) can have multiple reasons, but most often, the applied rule is over-generalized (e.g., implying `Football_player` when the listing is actually about athletes in general) or applied to the wrong listing (i.e., the context described by the rule is not expressive enough). Finally, (4) happens, for example, when a table holds the specifications of a camera, as this cannot be expressed with the given set of predicates in DBpedia or CaLiGraph.

Overall, most of the errors are produced by incorrectly applied rules. This is, however, unavoidable to a certain extent as KGs are not error-free, and the data corpus is not perfect. A substantial portion of errors is also caused by incorrectly parsed or identified SEs. Reducing these errors can also positively impact the generated rules, as correct information about entities is required for correct rules.

9.6 Conclusion

In this work, we demonstrated the potential of exploiting co-occurring similar entities for IE, especially for discovering assertions for novel entities. We show that it is possible to mine expressive, descriptive rules for listings in Wikipedia, which can be used to extract information about millions of novel

Error type	Type	Relation
(1) Entity parsed incorrectly	2.6	0.2
(2) Wrong SE identified	1.4	1.6
(3) Rule applied incorrectly	3.7	2.3
(4) Semantics of listing too complex	0.3	0.0

Table 9.4: Error types partitioned by cause. The occurrence values are given as their relative frequency (per 100) in the samples evaluated in Table 9.3.

entities. The implemented approach is published as part of the CaLiGraph extraction framework.[14]

Besides Wikipedia, another potential data source for the approach is the Fandom[15] universe containing more than 380K wikis on various domains (among them many interesting wikis for our approach, like for example WikiLists[16]). For background knowledge, existing KGs in this domain, like DBkWik [77] or TiFi [31], can be used. The approach could also be extended towards arbitrary web pages, using Microdata and RDFa annotations [127] as hooks for background knowledge.

[14] https://github.com/nheist/CaLiGraph
[15] https://www.fandom.com/
[16] https://list.fandom.com/wiki/Main_Page

Part IV

Knowledge Graph Evaluation and Usage

CHAPTER 10

KGrEaT: Evaluating Knowledge Graphs via Downstream Tasks

Countless research papers have addressed the topics of KG creation, extension, or completion to create KGs that are larger, more correct, or more diverse. This research is typically motivated by the argument that using enhanced KGs to solve downstream tasks will improve performance. Nonetheless, this is hardly ever evaluated. To better judge how well KGs perform on actual tasks, we present KGrEaT – a framework to estimate the quality of KGs via actual downstream tasks like classification, clustering, or recommendation. Instead of comparing different methods of processing KGs with respect to a single task, the purpose of KGrEaT is to compare various KGs by evaluating them on a fixed task setup. The contributions of this chapter are as follows:

- With KGrEaT, we present a framework to judge the utility of KGs using extrinsic task-based metrics.
- In our experiments, we demonstrate the framework's capabilities by evaluating and comparing several well-known cross-domain KGs.

The work presented in this chapter is based on the following publication:

Nicolas Heist,† Sven Hertling† and Heiko Paulheim. KGrEaT: A Framework to Evaluate Knowledge Graphs via Downstream Tasks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (CIKM), pp. 3938-3942, Birmingham, UK, October 2023, ACM Press.[1] [65]

[1]The contributions of the author to this publication are as follows: design and implementation of the framework, KG evaluation experiments.

10.1 Motivation

Efforts related to KG construction, extension, and completion are typically motivated by the argument that leveraging enhanced KGs can lead to improved performance in downstream tasks. However, comparative evaluations of different KGs w.r.t. their utility for such tasks are rarely conducted.

In the literature, the vast majority of studies concerned with the evaluation of KGs have focused on intrinsic metrics that work exclusively with the triples of a graph. Several works introduce quality metrics like accuracy, consistency, or trustworthiness and propose ways to determine them quantitatively [198, 11, 51, 84, 214]. Färber et al. [43] compare KGs concerning size, complexity, coverage, and overlap. Additionally, they provide guidelines on which KG to select for a given problem.

Another line of work computes extrinsic task-based metrics to evaluate KG embedding approaches. They use a fixed input KG with a fixed evaluation setup while varying only the embedding approach. Frameworks like GEval [151] or kgbench [19] use data mining tasks like classification or regression for the evaluation. Others, like Ali et al. [4], evaluate primarily on link prediction tasks.

10.2 Framework

To address the evaluation gap of extrinsic metrics for KGs, we propose a framework called KGrEaT (**K**nowledge **Gr**aph **E**v**a**luation via Downstream **T**asks).[2] KGrEaT aims to comprehensively assess KGs by evaluating them on multiple kinds of tasks like classification, regression, or recommendation. The evaluation results (e.g., the accuracy of a classification model trained with the KG as background knowledge) serve as extrinsic task-based quality metrics for the KG. By defining a fixed evaluation set up in the framework and applying it to multiple KGs, it is possible to isolate the effect of every KG and compare how useful they are for solving different kinds of tasks. KGrEaT is built in a modular way to be open for extensions from the community, like additional tasks or datasets.

10.2.1 Purpose and Limitations

KGrEaT is a framework built to evaluate the performance impact of KGs on multiple downstream tasks. To that end, the framework implements various algorithms to solve tasks like classification, regression, or recommendation of entities. The impact of a given KG is measured by using its information as background knowledge for solving the tasks. To compare the performance of different KGs on downstream tasks, we use a fixed experimental setup

[2] `https://github.com/dwslab/kgreat`

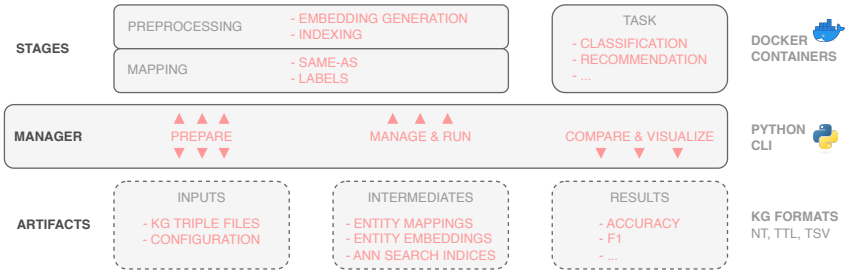


Figure 10.1: An overview of the KGrEaT framework.

with the KG as the only variable. The performance indicators may be used to make an informed decision when picking a KG for a given task. Further, they can be used to compare the performance of different versions of a single KG (e.g., during construction or its life cycle).

The implemented algorithms are not necessarily state-of-the-art because the primary objective is not to measure how well a task can be solved with a given KG in absolute numbers but rather in comparison to other KGs or different versions of the same KG. Hence, the absolute numbers of the results only have limited expressive power. However, the framework tries to reduce the bias in the results by averaging over multiple preprocessing methods, datasets, and algorithms.

KGrEaT automatically maps the entities of the KG to the dataset's entities using a set of configurable mappers. Undoubtedly, the quality of this mapping influences the results generated by the framework. But as the mapping procedure is applied similarly to all evaluated KGs, the mapping quality is mainly influenced by the accessibility of the graph (i.e., whether it provides sufficient context information like labels or descriptions for its entities). To reduce the influence of the mapping strategy on the overall results, the framework provides a way to run experiments with multiple mapping approaches (and, possibly, average over them).

10.2.2 Design

The framework is designed modularly (c.f. Fig. 10.1), making adding additional preprocessing steps, mappers, or tasks easy. Every step of a stage is implemented as an isolated docker container[3] with its own environment so that additions can be made without any constraints on the programming language. Another advantage of the container-based architecture is the easy distribution of containers via a container hub, eliminating the need for users to build the framework on their own machines.

The manager is responsible for making necessary preparations (e.g., downloading the input data or gathering entities to be mapped), executing

[3] `https://www.docker.com`

the stages (fetching and running containers of the steps), and visualizing the results (e.g., comparing KG performance on various aggregation levels). The *Preprocessing* and *Mapping* stages can be executed in parallel; the results are used to execute the *Task* stage. The process is steered via a Command Line Interface (CLI).

The only input to the evaluation process is the KG in the form of RDF files as well as a configuration. The latter defines how the stages should be run (i.e., which steps to execute in which order). Further, every step can be configured in depth to supply relevant hyperparameters. For example, one can configure how the KG should be mapped to the datasets (e.g., via matching labels) and define an acceptable similarity value for a match.

In the following, we provide details of the three main stages executed when evaluating a KG.

10.2.3 Preprocessing Stage

The *Preprocessing* stage creates all pre-computable artefacts that are needed in the subsequent *Task* stage (e.g., intermediate representations or statistics of the KG). So far, this stage comprises the computation of embeddings ($TransE$ [20], $TransR$ [110], $DistMult$ [213], $RESCAL$ [140], $ComplEx$ [190] via the DGL-KE framework [221], and RDF2vec [168] via the jRDF2vec framework [155]). Further, it supports the generation of indices for Approximate Nearest-Neighbor (ANN) search (via the hnswlib library [121]).

10.2.4 Mapping Stage

In the *Mapping* stage, the entities of the KG are automatically mapped to the entities in the datasets. So far, a *Same-As* mapper and a *Label* mapper are implemented. The former uses the same-as links of a KG to map its entities to those of the datasets. A dataset may provide URIs for an entity (e.g., from well-known KGs like DBpedia or Wikidata), but it has to provide at least one label. This label is used by the *Label* mapper to find a corresponding entity in the KG. It uses the RapidFuzz library[4] to estimate the similarity of labels via token-based edit distance. Mappers are composable, i.e., they can be executed in sequence. For example, entities are first mapped via same-as links where available, and the remaining entities are mapped via label similarity.

10.2.5 Task Stage

In the *Task* stage, the task types are executed for all combinations of datasets and algorithms. Table 10.1 gives an overview of all possible constellations.

[4] `https://github.com/maxbachmann/RapidFuzz`

KGrEaT contains 26 tasks (i.e., combinations of task types and datasets) that are run with one or more algorithms. Additionally, the algorithms are executed with multiple hyperparameter settings. How the individual tasks use the KG information depends on the task and the implemented algorithm. Generally, the tasks *Classification*, *Regression*, and *Clustering* use embeddings of the KG's entities as features of the models, and the remaining tasks use the distance between the entity embeddings to find related entities.

Several datasets are taken from Ristoski et al. [166] and from the GEval framework [151]. The *Recommendation* datasets MovieLens [61], LastFm,[5] and LibraryThing [220] are preprocessed as recommended by Di Noia et al. [141] except for using all entities instead of only those for which a mapping to DBpedia exists. The interested reader may refer to the respective publications and the information in the framework for detailed statistics of all datasets.

Every task type comes with suitable evaluation metrics computed for every constellation. As some KGs might not contain matches for all entities in the dataset, it would be unfair to compute metrics only over known entities (and discard unknown entities) or only over all entities. Hence, the framework reports metrics for both scenarios. Finally, the results can be aggregated over various levels (e.g., over embeddings, algorithms, and datasets) to produce metrics with a reduced bias.

10.3 Experiments

To show the capabilities of KGrEaT, we conduct experiments over multiple general-purpose KGs and analyze how well they perform on the implemented downstream tasks. We first give an overview of the evaluated KGs, define the experimental setup, and discuss the results.

10.3.1 Experimental Setup

Knowledge Graphs

We use the following KGs in our experiments:[6]

- *DBpedia*: Dumps from 2016-10 (*DBP16*) and 2022-09 (*DBP22*)[7]
- *YAGO*: Version 3
- *Wikidata* (*WD*): Dump from 2023-06-07
- *CaLiGraph* (*CLG*): Version 3.1.1 (as described in Chapter 11)
- *DBkWik*: Version 2[8]

[5] `http://www.lastfm.com`

[6] Links to the datasets are given in Appendix A.2.

[7] Using multiple versions allows us to compare not only between different KGs but also between different versions (here: with respect to time) of the same KG.

[8] Combined with DBpedia to also include the well-known entities of Wikipedia.

Task Type	Datasets	Algorithms	Evaluation Metrics
Classification	AAUP, Cities, Forbes, MillionSongDataset MetacriticAlbums, MetacriticMovies, ComicCharacters	Naive Bayes, KNN, SVM	Accuracy ↑
Regression	AAUP, Cities, Forbes, MetacriticAlbums, MetacriticMovies	Linear Regression, KNN, Decision Tree	RMSE ↓
Clustering	Cities2000AndCountries, CitiesAndCountries, Teams, CitiesMoviesAlbumsCompaniesUni, ComicCharacters	DBSCAN, KMeans, Agglomerative Clustering	ARI ↑, NMI ↑, Accuracy ↑
Document Similarity	LP50	Cosine Similarity	Spearman ↑, Pearson ↑
Entity Relatedness	KORE	Cosine Similarity	Kendall's Tau ↑
Semantic Analogies	AllCapitalCountryEntities, CapitalCountryEntities, CityStateEntities, CurrencyEntities	Cosine Similarity	Accuracy ↑
Recommendation	MovieLens, LastFm, LibraryThing	Item-Similarity recommender	F1 ↑

Table 10.1: Implemented tasks with their algorithms, datasets, and evaluation metrics.

Mapping

We first map the KGs with the *Same-As* mapper where applicable. Then, we apply two variants of the *Label* mapper: One with a similarity threshold of 1.0 for high-precision matches and one with a threshold of 0.7 for high recall. For the former, we compute metrics for known entities (**P**recision **K**nown - PK) and for all entities (**P**recision **A**ll - PA); for the latter, being recall-oriented, we report the metrics only for all entities (**R**ecall **A**ll - RA).

Embeddings

All experiments are executed with four embedding types (*TransE*, *DistMult*, *ComplEx*, and *RDF2vec*) to reduce the influence of the different embedding approaches on the overall results. For Wikidata, we could not compute all the embeddings due to the amount of computational resources necessary. Instead, we use precomputed *TransE* embeddings[9] with a comparable training configuration.

Hardware

All experiments are executed on NVIDIA RTX 2080 Ti graphic cards and Intel Xeon E5 processors (2.6GHz). On average, a full evaluation of a single KG takes roughly 30 hours, with 20 hours of embedding computation, 4 hours of mapping, and 6 hours of task execution.

10.3.2 Results and Discussion

Table 10.2 shows the final results of our evaluation for the three scenarios *PK*, *PA*, and *RA*. The results are averaged after aggregating for all embeddings, datasets, and algorithms. Detail results are given in Appendix B.1 and the complete data of the experiments are publicly available.[10]

For *Classification*, DBpedia2016 shows the best results in the precision setting, while CaLiGraph and YAGO achieve the best results in the recall setting. For *Regression*, both DBpedia versions and Wikidata perform well in the precision setup, while again, CaLiGraph and YAGO achieve the best results in the recall setting. The *Clustering* task is solved best by YAGO and DBpedia2016. For *Document Similarity*, version 2022 of DBpedia is the clear winner. For the *Entity Relatedness* task, using Wikidata, DbkWik, or DBpedia2022 as background knowledge produces the best results. *Recommendation* is solved best using DBpedia or Wikidata, *Semantic Analogies* is also solved best by DBpedia.

[9] `https://torchbiggraph.readthedocs.io/en/latest/pretrained_embeddings.html`

[10] `https://doi.org/10.5281/zenodo.8050446`

Task Type	DBP16	DBP22	YAGO	WD	CLG	DBkWik
PK (precision-oriented, known entities)						
Classification	**2.3**	3.0	4.0	**2.3**	4.8	4.0
Regression	3.0	3.6	3.6	**2.8**	3.0	5.0
Clustering	2.1	3.8	**2.0**	4.7	4.3	4.1
Document Similarity	2.0	**1.0**	5.0	5.3	3.0	4.7
Entity Relatedness	3.0	2.0	6.0	**1.0**	5.0	4.0
Semantic Analogies	**2.0**	**2.0**	3.2	6.0	4.0	3.8
Recommendation	2.7	2.7	5.3	**2.3**	3.3	4.7
PA (precision-oriented, all entities)						
Classification	2.2	4.8	3.3	5.2	3.3	**2.0**
Regression	**1.8**	3.6	3.8	4.4	4.0	3.4
Clustering	**2.0**	3.8	2.8	4.9	4.5	3.0
Document Similarity	2.0	**1.0**	5.0	5.7	3.0	4.3
Entity Relatedness	3.0	2.0	6.0	4.0	5.0	**1.0**
Semantic Analogies	**1.8**	**1.8**	3.0	6.0	4.5	4.0
Recommendation	3.0	**2.3**	5.0	4.3	3.7	2.7
RA (recall-oriented, all entities)						
Classification	3.4	5.6	2.6	4.9	**1.9**	2.7
Regression	4.0	4.4	1.6	5.2	**1.4**	4.4
Clustering	2.4	3.9	**2.3**	4.5	4.7	3.2
Document Similarity	2.3	**1.0**	4.0	6.0	5.0	2.7
Entity Relatedness	4.0	2.0	6.0	**1.0**	5.0	3.0
Semantic Analogies	**1.2**	2.5	3.8	6.0	3.2	4.2
Recommendation	**1.7**	3.7	5.7	4.0	3.3	2.7

Table 10.2: Evaluation results of the KGs given as average rank per task type. The results of the KGs are given for the dimensions PK (precision-oriented, known entities), PA (precision-oriented, all entities), and RA (recall-oriented, all entities). The best results are in bold; second-best results are underlined.

In general, DBpedia dominates the results to a large extent, which may be explained by the fact that some of the datasets used in the framework have been derived from the 2015 version of DBpedia. This might also explain why the 2022 version of DBpedia has no clear advantage over the older 2016 version. However, both versions of DBpedia perform strongly on the *Recommendation* task, which has no direct relation to DBpedia or even Wikipedia.

Our assumption that the KGs with more entities (YAGO, Wikidata, CaLiGraph, and DBkWik) will have an advantage, especially in the *Recommendation* tasks, did only partially prove to be true. However, they have shown strong performances, especially in recall-oriented settings. This unsteady performance may be due to the increased complexity of training expressive embeddings for large KGs. In the future, we want to explore this further by running evaluations not only with multiple types of embeddings but also with multiple embedding configurations (e.g., number of trained epochs). Another interesting direction to explore is whether combining two KGs (e.g., by concatenating their entity vectors) yields improved results [187].

10.4 Conclusion

We presented KGrEaT, a framework for evaluating the performance of KGs on multiple downstream tasks. Our experiments found that, depending on the task, the performance of the KGs varies enormously. To judge the quality of a KG in its completeness, extrinsic evaluation metrics provided by KGrEaT can serve as a valuable addition to the established intrinsic evaluation criteria.

CHAPTER 11

CaLiGraph: Statistics, Evaluation and Usage

The approaches for extracting information from semi-structured sources in Wikipedia presented in previous chapters have been largely integrated into the CaLiGraph extraction framework. Since its first published version in 2019, CaLiGraph has been refined and extended multiple times, where changes have been introduced to the ontology, the covered entities, and the underlying data source. This chapter aims to provide an overview of CaLiGraph as a data source. We describe its versions, purpose, and vocabulary structure. Then, we detail the extraction procedure of CaLiGraph, including sources, provenance, stability, and sustainability. We explain how CaLiGraph can be accessed and how it is used already. Finally, we present statistics of the KG, summarize evaluation results of the included approaches, and apply KGrEaT to compare CaLiGraph to other general-purpose KGs. The contributions of this chapter are as follows:

- We give a comprehensive overview of CaLiGraph as a data source, including general details, usage information, and statistics of the main versions.
- We compare the latest version of CaLiGraph with existing general-purpose KGs in terms of their performance on downstream tasks and give insights into the strengths and weaknesses of these graphs.

The work presented in this chapter is based on the following publication:

Nicolas Heist and Heiko Paulheim. CaLiGraph: A Knowledge Graph from Wikipedia Categories and Lists. In *Semantic Web Journal (SWJ)*, 2024. [under review]

11.1 Description

CaLiGraph and all the associated information is accessible via `http://caligraph.org`. The dataset is licensed under CC BY 4.0,[1] giving everyone the right to use, share and adapt all material with the only liability of giving proper attribution.

The project to create CaLiGraph was initiated in 2018 [64] and, to date, three major versions have been published. Here is an overview of the versions that we use in the remainder of this chapter:[2]

- **CLGv1** (version 1.1.0): contains the full class hierarchy and axioms as described in Chapters 5 and 6.
- **CLGv2** (version 2.1.1): adds additional entities extracted from all listings in Wikipedia as described in Chapter 7 and additional facts from associated rules as described in Chapter 9. However, this version may contain duplicate entities not properly disambiguated during extraction.
- **CLGv3** (version 3.1.1): adds an entity disambiguation step as described in Chapter 8.

11.1.1 Purpose and Coverage

The purpose of CaLiGraph is to serve as a large-scale general-purpose KG covering all topics addressed in Wikipedia. In particular, CaLiGraph aims to incorporate all information given in a semi-structured format in Wikipedia. By exploiting the data structure, the extraction mechanisms of CaLiGraph can extract information, especially about long-tail entities, more precisely than from full text. Currently, the focus is on extracting information about entities mentioned in tables and enumerations.

Another feature distinguishing CaLiGraph from most other public general-purpose KGs is its large taxonomy containing expressive class descriptions. An example is shown in the upper part of Fig. 1.1 on page 6 where *restriction1* enforces that all entities in the class *Guns N' Roses album* have the band *Guns N' Roses* as an artist. With such restrictions, we model the meaning of the classes that is usually hidden behind their names.

11.1.2 Vocabulary

The CaLiGraph dataset builds on well-established vocabularies like *RDF(S)*, *OWL*, *SKOS*, *FOAF* and *PROV-O* to describe important concepts like classes, hierarchies, restrictions, labels, and provenance. Overall, the complexity of the CaLiGraph ontology can be categorised as $\mathcal{SHOD}$. From an ontological

[1] `https://creativecommons.org/licenses/by/4.0`
[2] Refer to Appendix A.2.4 for version details.

perspective, the feature distinguishing CaLiGraph most from other public general-purpose KGs is its extensive modelling of class restrictions via OWL. These *has-value* restrictions (an example is shown in Fig. 1.1) already justify the $\mathcal{SHO}$ part of the ontology's complexity. As CaLiGraph defines data types for extracted values and contains axioms implying literals, it is classified as $\mathcal{SHOD}$.

11.2 Extraction Procedure

CaLiGraph is extracted using the CaLiGraph Extraction Framework.[3] We describe the extraction's inputs, outputs, and organisation in the following.

11.2.1 Data Sources

The main inputs to the CaLiGraph extraction framework are an XML dump of the English Wikipedia and the English chapter of DBpedia [104] in the form of triples. Further, we use WebIsALOD [74] to gather additional hypernyms during taxonomy construction (see Chapter 5).

11.2.2 Provenance

In CaLiGraph, we provide provenance information for new classes and entities using PROV-O vocabulary. For existing classes, properties and entities taken from DBpedia, we add links via *rdfs:subClassOf*, *owl:equivalentProperty* and *owl:sameAs*, respectively. Similar to DBpedia, we use the namespaces *http://caligraph.org/ontology/* or short *clgo* for the ontology and we use *http://caligraph.org/resource/* or short *clgr* for resources.

For additional classes, we point to the Wikipedia categories or list pages used for extraction. For the additional entities, we include information about the listings they have been extracted from. Example 11.2.1 shows the provenance information generated when creating a new class from the list page *List of lakes of Powell County, Montana*.

Example 11.2.1. Expression of provenance in CaLiGraph

```
@prefix wiki: <https://en.wikipedia.org/wiki/> .
clgo:Lake_of_Powell_County,_Montana prov:wasDerivedFrom
    wiki:List_of_lakes_of_Powell_County,_Montana .
```

The example also shows that, when minting new classes or entities, we again follow DBpedia and create a URI similar to its main label. We use the source's name as a prefix in case of name clashes.

[3] `https://github.com/nheist/CaLiGraph`

11.2.3 Stability

CaLiGraph is built on information from Wikipedia and DBpedia. New releases are dependent on the information from these two resources. As we have no control over the data sources, CaLiGraph gives no guarantees for the stability of ontology and resources between major versions. The changes may affect any information contained in CaLiGraph. For example, it is possible that a page name in Wikipedia and, consequently, a resource in DBpedia changes. This change would then be taken over in CaLiGraph as well. Further, if the structure of the category graph in Wikipedia changes, this can influence the extraction of the CaLiGraph taxonomy. Finally, any changes in listings in Wikipedia may change how facts are extracted. This influences especially the URIs of entities without dedicated Wikipedia articles, as we generate them based on the listing context the entity is extracted from.

11.2.4 Sustainability

CaLiGraph is hosted and maintained by the Data and Web Science Group of the University of Mannheim. The release cycle for CaLiGraph was mostly irregular in the past, as new developments were integrated as quickly as possible. There are ongoing efforts to align the release cycle to the one of DBpedia and even to integrate the extraction of CaLiGraph into the DBpedia extraction workflow. Still, it is planned to improve and extend CaLiGraph further in various ways (see future work in Chapter 13).

11.3 Usage

The following section describes how to access and interact with CaLiGraph best. Further, we give an overview of potential and existing use cases.

11.3.1 Access

The main web resources to view, use, and extend CaLiGraph are:

- `http://caligraph.org`: Main website with relevant resources, a data explorer and a SPARQL endpoint.
- `https://zenodo.org/record/3484511`: Source files of all published versions of CaLiGraph.
- `https://databus.dbpedia.org/nheist/CaLiGraph`: CaLiGraph on the DBpedia Databus.
- `https://github.com/nheist/CaLiGraph`: Code of the extraction framework, including an issue tracker.

11.3.2 Use Cases

In general, CaLiGraph is intended to be used as a KG for various domains similar to DBpedia. Hence, it can be used in similar use cases, for example, for information retrieval or question answering. CaLiGraph is already used in several concrete scenarios:

- Qin and Iwaihara [158] use CaLiGraph as training data for a transformer model to annotate table columns with entity types.
- Biswas et al. [16] use CaLiGraph to evaluate models for entity typing using only the surface forms of the entities.
- In 2021, we submitted CaLiGraph as a dataset for the Semantic Reasoning Evaluation Challenge [69]. It has been used in every challenge edition to evaluate reasoning systems (for example, by Chowdhury et al. [30]).

11.4 Statistics

In this section, we show statistics about CaLiGraph and compare it with DBpedia and YAGO. We use the English chapters of DBpedia in the versions from 2016 (*DBP16*) and 2022 (*DBP22*) as well as YAGO version 3.1 (*YAGO3*). We select these KGs for comparison as they are, like CaLiGraph, mainly based on the English Wikipedia.

11.4.1 General Metrics

We compare the KGs w.r.t. classes and entities in Table 11.1. Compared to DBpedia, YAGO and CaLiGraph contain many more classes, largely retrieved from the WCG. The increase in classes and relations in the major CaLiGraph versions is caused by the Wikipedia version used for extraction (*CLGv1* uses a version from 2016, *CLGv2* from 2020 and *CLGv3* from 2022). *YAGO3* uses a Wikipedia version from 2017; an extraction on a recent version would likely increase the number of classes. Regarding the class tree's depth and branching factor, DBpedia and CaLiGraph are comparable, while YAGO has a denser and deeper taxonomy. With the increasing maturity of CaLiGraph, the depth of the class tree decreases while its breadth increases. This may be an effect of the changed Wikipedia version and our efforts to integrate the isolated parts of the ontology better.

Regarding entities, *CLGv2* and *CLGv3* contain the highest number, almost twice as many as the other KGs. While the entities in *CLGv2* may not be properly disambiguated, *CLGv3* employs a disambiguation mechanism to ensure that no duplicate entities exist (see Chapter 8). *CLGv3* contains the highest number of assertions about entities, but the average number of assertions per entity is higher in *YAGO3*. This can be attributed mostly to

	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
# Classes	760	1,245	819,292	755,440	1,061,597	1,285,484
# Relations	1,355	1,298	77	271	343	1,253
# HasValue Restrictions	–	–	–	110,180	128,016	145,631
Avg. depth of class tree	3.5	3.9	6.6	4.5	3.9	3.6
Avg. branching factor of class tree	4.5	4.2	8.5	4.5	4.4	4.9
Ontology complexity	$\mathcal{SHOFD}$	$\mathcal{SHOFD}$	$\mathcal{SHOIF}$	$\mathcal{SHOD}$	$\mathcal{SHOD}$	$\mathcal{SHOD}$
# Entities	5,044,223	7,495,054	6,349,359	6,516,892	15,230,974†	13,736,724
# Assertions	71,628,627	97,213,941	263,433,367	166,228,505	298,300,766	332,884,815
Avg. linking degree	2.8	2.7	1.9	1.6	0.7	1.7
Median ingoing edges	0	1	0	0	0	0
Median outgoing edges	15	11	35	17	12	12

Table 11.1: Basic metrics of all CaLiGraph versions and other KGs based on the English Wikipedia. †Entities are not disambiguated properly.

Figure 11.1: A sunburst diagram of frequent entity types in CaLiGraph.

literal assertions of YAGO, as the median of outgoing edges is high, but the average linking degree is comparably low. Compared to YAGO and CaLiGraph, DBpedia is interlinked very strongly, indicated by the high average linking degree. The lower linkage degree of CaLiGraph versions 2 and 3 is caused by the large number of entities added for which only little information is available (i.e., long-tail entities). We observe the same effect in the reduction of median outgoing edges.

11.4.2 Contents

Like DBpedia and YAGO, CaLiGraph covers many domains. Fig. 11.1 shows how the entities in *CLGv3* are distributed over the type hierarchy. Most entities describe *Species*, with a majority being *Persons*. Next in line are *Works*, most importantly musical works and movies. Entities describing

	CLGv1		**CLGv2**		**CLGv3**	
Types	Count	Rank	Count	Rank	Count	Rank
Person	1,827,240	2	4,599,249	2	3,262,511	5
Organization	593,462	13	1,106,098	19	691,732	25
Populated_place	648,673	9	1,014,971	24	867,281	20
Natural_place	132,618	102	180,930	125	164,987	102
Species	353,680	26	790,287	30	3,691,343	1
Work	607,858	12	2,468,257	7	1,832,985	8
Building_or_structure	202,888	67	619,983	35	524,310	37
Gene	1,112	9,149	25,852	817	14,019	1,269
Protein	6,049	1,882	3,882	4,455	5,264	3,138
Event	141,582	90	309,674	67	1,178,248	14
Properties	Count	Rank	Count	Rank	Count	Rank
birthPlace	2,827,536	1	2,844,951	1	3,887,146	1
birthYear	1,128	133	1,175,897	4	1,576,909	2
location	877,066	3	1,485,013	2	1,567,334	3
team	521,660	5	748,147	6	1,338,344	4
country	963,588	2	1,265,201	3	1,315,784	5
subdivision	-	-	844,344	5	1,233,513	6
birthDate	-	-	-	-	976,431	7
type	284,581	8	390,191	8	716,693	8
genre	326,955	7	405,484	7	711,336	9
deathYear	111,273	15	11,916	78	706,773	10

Table 11.2: Comparison of counts and ranks of prominent types and properties among CaLiGraph versions. Prominent types are taken from Chapter 3, and prominent properties are selected based on their frequency in CLGv3.

Places are mostly addressing *Populated places*. The large number of *Places in Myanmar* can be traced back to an incorrect mapping in the taxonomy, which will be fixed in the next release (see Chapter 13 for more details).

We compare the type and relation frequencies of the three CaLiGraph versions in Table 11.2. We use the prominent types mentioned in Chapter 3 to compare types. Unfortunately, the ranks are not perfectly comparable as DBpedia changed its taxonomy, taking effect in *CLGv3*. As a consequence, *Person* is a descendant of *Species* instead of *Agent*. This explains why *Species* is the most frequent type in *CLGv3* and why *Person* descended to rank five while almost doubling its numbers compared to *CLGv1*. While the coverage of organizations and places has not increased much between versions 1 and 3, the counts of *Work* (+200%), *Building_or_structure* (+160%), *Gene* (+1,160%) and *Event* (+700%) multiplied.

To compare properties, we take the most frequently used ones in *CLGv3*. Again, some changes can be explained with the changes in DBpedia, like the increased coverage of *birthYear* (about 1K in *DBP16* and 260K in *DBP22*) and the added support for *subdivision* and *birthDate*. The decline in rank

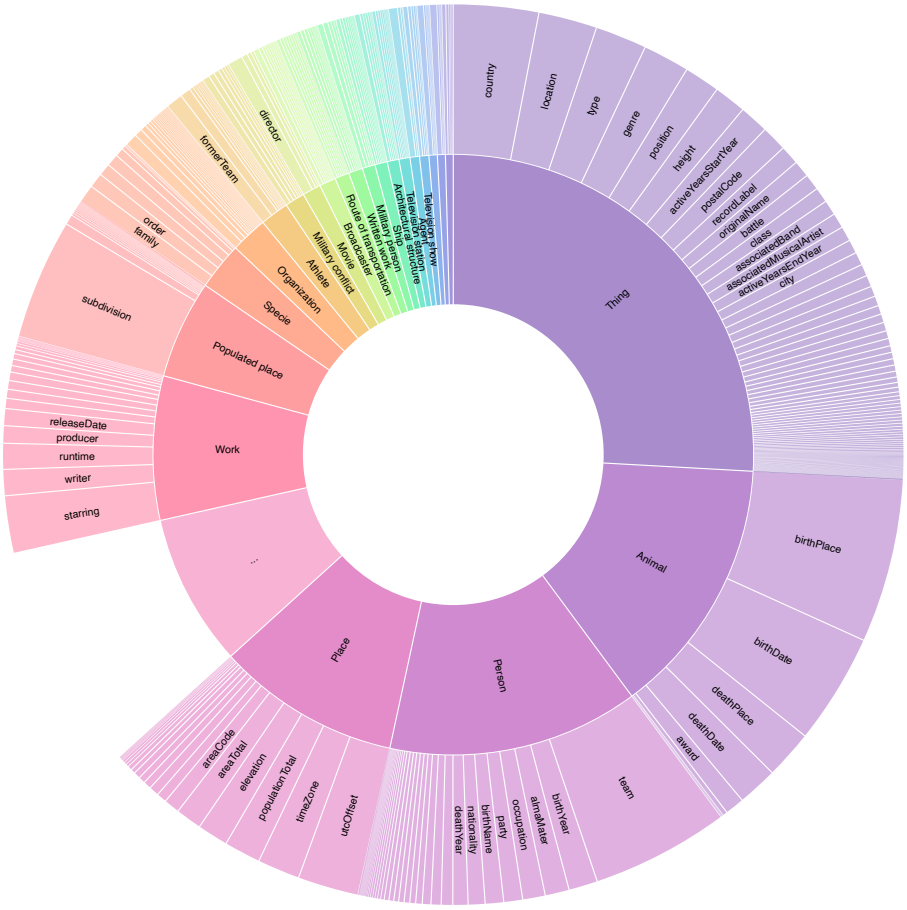


Figure 11.2: A sunburst diagram of frequent properties in CaLiGraph. The properties (outer ring) are grouped by their domain (inner ring).

for *country* aligns with the observations on types, indicating that locations are already well covered in DBpedia. Fig. 11.2 shows the distribution of properties in *CLGv3* grouped by domain.

11.5 Data Quality

We provide information about the data quality in CaLiGraph concerning its metadata (Section 11.5.1), the vocabulary use (Section 11.5.2), as well as class and instance data (Section 11.5.3).

11.5.1 Metadata

As described in Section 11.1, CaLiGraph's ontology builds on well-established vocabularies like SKOS, FOAF, and PROV-O. Further, the KG is described in various aspects (e.g., purpose, creators, version) with the vocabulary of the *Dublin Core*. In 2023, Andersen et al. [5] conducted an experiment evaluating the accountability of 670 KGs in the LOD Cloud. More concretely, they evaluated how much information KGs provide about their data collection, maintenance and usage (e.g., who created the KG and how was it created?). They retrieve this information via SPARQL queries. Of the 670 KGs, only 29 responded to queries; of those, CaLiGraph was ranked fifth. Based on the results of the experiments, we added more metadata so that, all else equal, CaLiGraph would now be ranked first.

11.5.2 Five Star Rating

According to the five-star rating for linked data vocabulary use (cf. Section 2.1.2), the CaLiGraph dataset can be categorized as a four-star dataset and will be a five-star dataset soon:
★☆☆☆☆ There is dereferenceable human-readable information about the used vocabulary on `http://caligraph.org`.
★★☆☆☆ The information is available as machine-readable explicit axiomatization of the vocabulary as the CaLiGraph ontology is published using *RDFS* and *OWL*.
★★★☆☆ The vocabulary is linked to other vocabularies, e.g., DBpedia (see Section 11.2.2).
★★★★☆ Metadata about the vocabulary is available (see Section 11.5.1).
★★★★★ The vocabulary is linked to by other vocabularies soon, as DBpedia is preparing to provide backlinks to CaLiGraph similar to the ones from CaLiGraph to DBpedia.[4]

11.5.3 Class and Instance Data

In Table 11.3, we collect all evaluation results of parts of CaLiGraph data conducted using direct or indirect human supervision. CaLiGraph intends to ingest as much of the semi-structured information in Wikipedia as possible. The results show that most of the information is extracted with an accuracy of over 90%, with entity linking approaches being the only exception.

The CaLiGraph extraction pipeline is a sequence of steps, with later ones depending on the results of previous steps. It is, hence, unavoidable that errors are propagated through the pipeline. The evaluations listed in Table 11.3 identify such errors explicitly. In the results of NASTyLinker (Chapter 8), the errors are not contained in the final accuracy of 89.4% for

[4] `https://www.dbpedia.org/resources/latest-core/`

Target	Method	Metric	#Samples	Result	Source
Edges in the taxonomy (cf. Chapter 5)	Manual evaluation via Amazon MTurk[a] (majority of three votes)	Accuracy	2,000	96.25% (±0.86%)	[68]
Type restrictions (cf. Chapter 6	Manual evaluation via Amazon MTurk (majority of three votes)	Accuracy	250	96.8%	[67]
Relation restrictions (cf. Chapter 6)	Manual evaluation via Amazon MTurk (majority of three votes)	Accuracy	250	95.6%	[67]
Subject entities in arbitrary listings (cf. Chapter 7)	Evaluation on manually labelled dataset	F1-score (*Exact* match)	9,400	74%	[71]
Single mention assigned to an entity (cf. Chapter 8)	Manual evaluation by authors	Accuracy	100	89.4% (±9.6%)	[72]
All mentions assigned to an entity (cf. Chapter 8)	Manual evaluation by authors	Accuracy	100	82.3% (±7.0%)	[72]
Entity types derived from listings (cf. Chapter 9)	Manual evaluation by authors	Accuracy	2,000	91.95% (±1.19%)	[70]
Relations derived from listings (cf. Chapter 9)	Manual evaluation by authors	Accuracy	1,000	95.90% (±1.23%)	[70]

Table 11.3: Collection of evaluation results of CaLiGraph data.

[a] https://www.mturk.com/

single mentions and 82.3% for all mentions. Considering the SE labelling errors (Chapter 7), the results for single mentions would decrease by 5.4%, and the results for all mentions would decrease by 3.3%. The results for extracting facts from listings (Chapter 9) include errors caused by incorrectly parsed entities already. The errors are responsible for an accuracy decrease of 2.6% for entity types and 0.2% for relations.

11.6 Evaluation via Downstream Tasks

In the following, we use KGrEaT to assess the utility of CaLiGraph and compare it to related KGs.

11.6.1 Experimental Setup

We consider CaLiGraph (*CLGv1*, *CLGv2*, *CLGv3*), DBpedia (*DBP16*, *DBP22*) and *YAGO3* in our comparison. We run the evaluation for all seven tasks in KGrEaT: Classification, Regression, Clustering, Document Similarity, Entity Relatedness, Semantic Analogies and Recommendation. The tasks are evaluated on 20 datasets covering areas like geography, music, movies or literature. *MillionSongDataset*, *ComicCharacters*, *MovieLens*, *LibraryThing* and *LastFm* are datasets derived from independent sources; the remaining datasets are created from DBpedia version 2015.

We report the results for two entity mapping scenarios: precision-oriented mapping and recall-oriented mapping. Both scenarios link the task dataset's entities to KG entities using *owl:sameAs* links and labels. The former scenario uses a precision-focused label mapper, while the latter uses a label mapper focused on recall. In the precision-oriented scenario, we consider only mapped entities in the evaluation, while in the recall-oriented scenario, we consider all entities.

We compute the results using four embedding methods: $TransE$ [20], $DistMult$ [213], $ComplEx$ [190], and $RDF2vec$ [168]. We run evaluations with embeddings trained for one and two epochs, respectively. In total, we compute results for eight configurations for every KG and scenario; we take the best approach w.r.t. embedding and algorithm, and then we aggregate the results by task, dataset and metric.

11.6.2 Results and Discussion

Table 11.4 shows the average rank of the KGs w.r.t. the datasets of a task. In both scenarios, DBpedia shows superior performance in the Clustering, Entity Relatedness, and Semantic Analogies tasks, YAGO works best for Document Similarity, and CaLiGraph for Regression and Recommendation. While

Task Type	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
Precision-oriented mapping						
Classification	**1.8**	2.2	3.5	5.0	5.3	3.2
Regression	3.0	3.2	3.6	4.0	5.6	**1.6**
Clustering	**2.3**	2.8	2.8	4.4	4.4	4.3
Document Similarity	4.0	6.0	**1.3**	3.0	1.7	5.0
Entity Relatedness	2.0	**1.0**	4.0	5.0	6.0	3.0
Semantic Analogies	**1.8**	3.5	4.0	3.8	6.0	2.0
Recommendation	2.7	3.3	4.3	3.0	5.3	**2.3**
Recall-oriented mapping						
Classification	3.3	5.1	3.4	**2.3**	4.4	2.4
Regression	5.6	5.4	3.0	2.2	3.6	**1.2**
Clustering	**2.5**	4.5	3.0	3.5	3.9	3.7
Document Similarity	4.0	6.0	**1.0**	2.3	2.7	5.0
Entity Relatedness	**1.0**	3.0	4.0	5.0	6.0	2.0
Semantic Analogies	**1.8**	3.8	4.0	3.0	6.0	2.5
Recommendation	2.0	3.3	5.0	4.0	5.0	**1.7**

Table 11.4: Evaluation results of the KGs given as average rank per task type. The results are computed for a precision-oriented mapping scenario and a recall-oriented mapping scenario. The best results are bold, and the second-best are underlined.

DBpedia tends to work better in the precision-oriented scenario, CaLiGraph works better in the recall-oriented scenario.

On a dataset level (see Appendix B.2 for details), it becomes clear that the choice of a KG for a given task is always dependent on the domain. As expected, DBpedia performs well on DBpedia-based datasets. The superior performance of *DBP16* compared to *DBP22* may be explained by the temporal proximity of *DBP16* to DBpedia version 2015, serving as the source for many datasets. For almost all independent datasets, we find that CaLiGraph and YAGO have much higher coverage than DBpedia (see Table 11.5). Especially *CLGv3* shows the highest coverage for all these datasets in the recall-oriented scenario. Consequently, using CaLiGraph for *ComicCharacters* and *MillionSongDataset* (used in Classification and Clustering) as well as for *MovieLens* and *LibraryThing* (used in Recommendation) produces superior results. Against our expectations, however, DBpedia shows a competitive performance for the independent datasets of the Recommendation task.

11.7 Conclusion

This chapter gave a comprehensive overview of CaLiGraph, a KG extracted from Wikipedia categories and lists. We presented general details, usage information, and statistics of the KG and compared it to other general-purpose KGs w.r.t. its performance on downstream tasks. The chapter contains all the information necessary to access and use CaLiGraph in a real-world application. The evaluation painted a rather differentiated picture, where a general recommendation for one or another KG is impossible. The right KG for a given task has to be picked based on multiple factors of influence, but our experiments showed that CaLiGraph could be an attractive option given the right circumstances.

Dataset	DBP16		DBP22		YAGO3		CLGv1		CLGv2		CLGv3	
	P	R	P	R	P	R	P	R	P	R	P	R
Cities	97	96	87	88	96	100	95	100	97	100	93	100
Forbes	87	87	81	81	99	100	92	100	97	100	91	100
AAUP	99	98	88	88	99	100	95	100	99	100	94	99
MetacriticMovies	98	98	95	94	90	100	100	100	100	100	98	100
MetacriticAlbums	99	99	97	97	95	100	97	100	99	100	96	100
MillionSongDataset†	6	22	6	21	11	51	10	51	20	60	20	64
Teams	100	100	94	94	77	83	99	100	95	97	94	95
ComicCharacters†	0	22	0	17	0	59	0	53	0	57	0	62
CitiesAndCountries	100	100	95	95	96	100	100	100	97	100	96	100
Cities2000AndCountries	100	100	93	93	94	99	100	100	96	100	94	100
CitiesMoviesAlbumsCompaniesUni	90	88	85	85	94	100	96	99	96	93	88	99
LP50	90	90	88	88	83	99	92	100	89	100	92	100
KORE	100	100	100	100	100	100	100	100	100	100	100	100
CurrencyEntities	100	100	93	93	100	100	100	100	97	100	97	100
CityStateEntities	96	97	97	97	99	100	98	100	97	100	82	100
CapitalCountryEntities	100	100	100	100	100	100	100	100	100	100	100	100
AllCapitalCountryEntities	99	99	96	97	99	100	98	100	97	99	96	99
MovieLens†	16	62	15	60	14	92	16	95	16	95	15	96
LibraryThing†	18	42	18	41	21	84	22	86	27	91	28	92
LastFm†	94	94	91	92	94	99	94	99	94	99	93	99

Table 11.5: Dataset coverage (per cent) of the KGs evaluated with KGrEaT for the precision- and recall-oriented mapping scenarios. Datasets marked with a dagger are independent of DBpedia.

Part V

Conclusion and Outlook

CHAPTER 12

Summary

In this thesis, we presented and applied various approaches for the automated extraction of information from semi-structured data in Wikipedia. The main focus of the extraction efforts was set on categories and listings, where especially the co-occurrence of entities in specific contexts was exploited. The extracted information, namely a fine-grained general-purpose ontology, novel entities, and assertions, is published as a KG with the name CaLiGraph. Our analyses and evaluations showed that CaLiGraph is a valuable addition to the field of publicly accessible general-purpose KGs, especially due to its rich ontology and coverage of long-tail entities.

With this thesis, we added contributions to the field of AKGC with respect to the following challenges:

(C1) Ontology with expressive, fine-grained types In Part II of this thesis, we developed an approach that forms an expressive taxonomy by combining Wikipedia categories with list pages in Chapter 5. The taxonomy was refined with axioms generated by the Cat2Ax approach from Chapter 6. The axioms describe the semantics expressed by the categories and were integrated into the ontology as *has-value restrictions*. The created ontology, containing in its most recent version almost 1.3 million classes and 150 thousand restrictions, forms the TBox of CaLiGraph.

(C2) Coverage of long-tail entities Part III targeted Wikipedia listings for the identification and extraction of entities along with information derived from context. In Chapter 7, we showed how to effectively identify mentions of subject entities in listings using a Transformer model. To properly assign the identified mentions to entities in a KG, we developed the NASTyLinker

approach in Chapter 8, which is capable of linking to existing and unknown entities in a KG. We further presented an approach to derive additional assertions from the context of listings in Chapter 9. The information extracted from Wikipedia was included in the ABox of CaLiGraph, making the graph describe almost 14 million entities. Most of the included entities are located in the long tail, as they are not popular enough for a dedicated article in Wikipedia.

(C3) Maintain high data quality In Part IV, we developed the KGrEaT framework for evaluating KGs on real downstream tasks as most existing KG evaluation frameworks use primarily intrinsic metrics. With the evaluation on downstream tasks, we add an extrinsic perspective to the evaluation of KGs. We further gave an overview of CaLiGraph as a dataset and showed its competitiveness in terms of coverage and performance on downstream tasks.

While the approaches presented in this thesis were primarily applied to extract information from Wikipedia, they are, with some exceptions, formulated in a generic form to allow an application to other data sources as well. In several chapters, we have already mentioned potential further sources like wikis from the Fandom universe or websites annotated using Microdata or RDFa.

CHAPTER 13

Limitations and Future Work

In this chapter, we discuss the limitations of CaLiGraph as a KG and reflect on the individual approaches used for extracting the information. Finally, we end this thesis by showing opportunities for future work in this field.

13.1 Limitations

While AKGC is a never-ending process as the world changes constantly, we can still try to transform the information available on the Web in an implicit format into structured knowledge as efficiently as possible. For CaLiGraph, we identified several topics as the main limiting factors. However, many of them currently apply to any AKGC system.

Error Accumulation. AKGC in CaLiGraph is executed as a pipeline of automatic processing steps. Errors in early steps are propagated to subsequent steps and may create distortions with a high impact on the outcome. Currently, the discussed approaches, which are integrated into the extraction framework, act independently. Hence, later steps have to live with the errors introduced in earlier steps.

For example, in the recent version of CaLiGraph, an extraction error made *Place in Myanmar* a superclass of *Village* (explaining its large proportion in Fig. 11.1 on page 165). This error occurred during OC, so it affects all steps of KGP.

Entity Ambiguity. Ambiguity is one of the biggest challenges when identifying and disambiguating mentions of entities in text. As information about

long-tail entities during extraction is sparse, the quality of such entities in CaLiGraph is not satisfactory yet.

Data Source Limitation. Currently, the CaLiGraph extraction targets a single version of Wikipedia only. Any information not contained in that version can consequently not be part of the KG. Further, we have no direct influence on the content of Wikipedia and hence have to deal with potential problems only during extraction.

Background Knowledge Dependency. CaLiGraph builds on the ontology of DBpedia, as the individual approaches apply distant supervision using information from DBpedia. Hence, CaLiGraph is taking over all existing types and properties. While types are extended, the set of properties remains fixed, and knowledge can only be modelled within the bounds defined by the DBpedia ontology.

13.2 Future Work

For future work in CaLiGraph, the focus should be divided between improving the quality of the existing KG and extending its coverage to incorporate more knowledge. However, it is also necessary to put more effort into the evaluation of KGs to better judge the quality of AKGC approaches and be able to pick the best KG for a given task.

Improving Extraction Quality. While error propagation is currently problematic in CaLiGraph, it is also a chance to improve the overall quality of the graph by gradually improving the individual parts. Fixing errors in the early stages of the extraction may positively influence the complete extraction pipeline. To that end, it would be favourable to implement a rigorous error-monitoring system in the extraction framework to capture errors early and monitor all parts of the pipeline to identify opportunities for improvement.

A possible way to deal with errors would be to integrate the isolated parts of the pipeline. By establishing feedback loops and mutual refinement, errors could be identified earlier, and later steps would be enabled to influence the extraction results of earlier steps.

A concrete improvement would be to replace or augment the taxonomy induction step described in Chapter 5 with a Transformer model that is tuned on identifying subclass relationships (e.g., from Hertling and Paulheim [79]). This may improve the class hierarchy substantially as we currently rely on manually combined hypernym information from multiple sources. As parts of OC and KGP are then based on the same technical foundation (i.e.,

Transformer models), an integration of these approaches with joint learning would be possible.

Putting more emphasis on the dependency of SEs expressed through co-occurrence might be particularly helpful when trying to disambiguate entities in text. We are only implicitly using the context of an entity mention during disambiguation. Explicitly providing information about related entities might improve the disambiguation capabilities of NASTyLinker. Similarly, adding more context, like substring patterns or section text, to the IE approach from Chapter 9 may improve extraction results. Adding all this complexity, however, comes at the cost of increased run time for the extraction procedure.

Extending KG Coverage. CaLiGraph could be extended in the three dimensions of ontology, assertions and data sources. To extend the ontology, we can discover additional axioms by extending the Cat2Ax approach from categories to list pages, as this is currently done only via the transitivity of the axioms in the taxonomy.

Additionally, we may derive more axioms by relying on common sense knowledge from another KG (e.g., CSKG [88]). New properties could be discovered by using the existing data in CaLiGraph as a foundation to automatically exploit dependencies between co-occuring entities where the relation underlying the co-occurrence pattern is not in the ontology yet.

Like YAGO, we can extend the coverage of CaLiGraph to more dimensions like temporal or geospatial information. As the KG currently reflects only the point in time when the Wikipedia dump was created, we may consider incorporating edits in Wikipedia pages to reflect the temporal dimension. Alternatively, one could explore the possibility of extracting CaLiGraph from multiple dumps and merging the results to include a temporal perspective.

CaLiGraph currently targets only the English Wikipedia. An extension to other languages would have the benefit of providing multilingual labels. Still, all the automatic extraction mechanisms may be able to derive much more complementary information from the diverse language chapters. The main challenge here is to merge the information derived from all the language chapters into a unified KG. Finally, we may extend the extraction to other data sources. As most extraction methods in the pipeline are built for encyclopedic content, a first step is to follow the example of DBkWik and target other Wikis than Wikipedia.

Broadening Evaluation Capabilities. As argued in Chapter 10, evaluations of KGs have to cover various aspects, including intrinsic and extrinsic metrics. For a more reliable evaluation of the performance of a KG on downstream tasks, KGrEaT would benefit strongly from additional datasets, mappers, tasks, and algorithms. We purposefully designed the framework to be as easy to extend as possible so that the user community can enhance the framework.

The framework could also be improved from the implementation perspective by adding more capabilities for KG analytics and a graphical user interface to explore the evaluations more comfortably.

Employing LMs for AKGC. With the advent of (large) LMs, the state of the art in many NLP and IE tasks is dominated by approaches using or extending such models. This is also the case for some of the approaches presented in this thesis. SLHCat [199] maps Wikipedia categories and lists to DBpedia, similar to our approaches presented in Chapters 5 and 6. Using BERT with prompt-based fine-tuning, they manage to produce mappings from CaLiGraph classes (i.e., categories and list pages) to DBpedia that are more specific and correct than Cat2Ax. The Cat2Type approach [17] also relies on BERT in combination with embeddings of Wikipedia categories to precisely predict a category's related type. Kouagou et al. [98] use embeddings of LMs to generalize the task and learn class expressions from groups of examples (e.g., members of a category).

More recently, Large Language Models (LLMs) have shown human-level performance on a large variety of tasks. While there are still severe limitations (like hallucinations and catastrophic forgetting) due to the implicit representation of knowledge in LLMs, many opportunities may arise from their combination with KGs [144]. Especially in AKGC, they can be applied to enhance many of the tasks addressed in this thesis, like taxonomy induction [90], entity linking, and information extraction [223]. Further, LLMs can reduce the effort of evaluating extraction results [114].

Bibliography

[1] Dhruv Agarwal, Rico Angell, Nicholas Monath, and Andrew McCallum. Entity linking and discovery via arborescence-based supervised clustering. *CoRR*, abs/2109.01242, 2021.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[3] Carlos A Alfaro, Sergio L Perez, Carlos E Valencia, and Marcos C Vargas. The assignment problem revisited. *Optimization Letters*, 16(5):1531–1548, 2022.

[4] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):8825–8845, 2021.

[5] Jennie Andersen, Sylvie Cazalens, and Philippe Lamarre. Assessing knowledge graphs accountability. In *The Semantic Web: ESWC 2023 Satellite Events - Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings*, volume 13998 of *Lecture Notes in Computer Science*, pages 37–42. Springer, 2023.

[6] Rico Angell, Nicholas Monath, Sunil Mohan, Nishant Yadav, and Andrew McCallum. Clustering-based inference for biomedical entity linking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2598–2608, 2021.

[7] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Extending the coverage of DBpedia properties using distant supervision over wikipedia. In *Proceedings of the NLP & DBpedia workshop co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 22, 2013*, volume 1064 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[8] Julián Arenas-Guerrero, Mario Scrocca, Ana Iglesias-Molina, Jhon Toledo, Luis Pozo-Gilo, Daniel Doña, Óscar Corcho, and David Chaves-Fraga. Knowledge graph construction with R2RML and RML: an ETL system-based overview. In *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*, volume 2873 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.

[9] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *International Semantic Web Conference*, pages 722–735. Springer, 2007.

[10] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference*, pages 79–85. Morgan Kaufmann Publishers / ACL, 1998.

[11] Taiyu Ban, Xiangyu Wang, Lyuzhou Chen, Xingyu Wu, Qiuju Chen, and Huanhuan Chen. Quality evaluation of triples in knowledge graph by incorporating internal with external consistency. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[12] Wouter Beek, Joe Raad, Jan Wielemaker, and Frank van Harmelen. sameas.cc: The closure of 500m OWL: sameas statements. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2018.

[13] Tim Berners-Lee. Linked data - design issues. `https://www.w3.org/DesignIssues/LinkedData.html`, 2006 (accessed Dec 15, 2023).

[14] Tim Berners-Lee. Semantic web and linked data. `https://www.w3.org/2009/Talks/0120-campus-party-tbl/#(14)`, 2009 (accessed Dec 15, 2023).

[15] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[16] Russa Biswas, Radina Sofronova, Mehwish Alam, Nicolas Heist, Heiko Paulheim, and Harald Sack. Do judge an entity by its name! entity typing using language models. In *The Semantic Web: ESWC 2021 Satellite Events: Virtual Event, June 6–10, 2021, Revised Selected Papers 18*, pages 65–70. Springer, 2021.

[17] Russa Biswas, Radina Sofronova, Harald Sack, and Mehwish Alam. Cat2type: Wikipedia category embeddings for entity typing in knowledge graphs. In *K-CAP '21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021*, pages 81–88. ACM, 2021.

[18] Kevin Blissett and Heng Ji. Cross-lingual NIL entity clustering for low-resource languages. In *Proceedings of the Second Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 20–25, 2019.

[19] Peter Bloem, Xander Wilcke, Lucas van Berkel, and Victor de Boer. kgbench: A collection of knowledge graph datasets for evaluating relational and multimodal machine learning. In *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, pages 614–630. Springer, 2021.

[20] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[21] Samuel Broscheit. Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 677–685, 2019.

[22] Volha Bryl, Christian Bizer, Robert Isele, Mateja Verlic, Soon Gill Hong, Sammy Jang, Mun Yong Yi, and Key-Sun Choi. Interlinking and knowledge fusion. In *Linked Open Data–Creating Knowledge Out of Interlinked Data*, pages 70–89. Springer, 2014.

[23] Volha Bryl, Christian Bizer, and Heiko Paulheim. Gathering alternative surface forms for dbpedia entities. In *Proceedings of the Third NLP&DBpedia Workshop (NLP & DBpedia 2015) co-located with the 14th International Semantic Web Conference 2015 (ISWC 2015), Bethlehem, Pennsylvania, USA, October 11, 2015*, volume 1581 of *CEUR Workshop Proceedings*, pages 13–24. CEUR-WS.org, 2015.

[24] Bruce G. Buchanan. A (very) brief history of artificial intelligence. *AI Mag.*, 26(4):53–60, 2005.

[25] Georg Buchgeher, David Gabauer, Jorge Martinez-Gil, and Lisa Ehrlinger. Knowledge graphs in manufacturing and production: A systematic literature review. *IEEE Access*, 9:55537–55554, 2021.

[26] Matteo Cannaviccio, Lorenzo Ariemma, Denilson Barbosa, and Paolo Merialdo. Leveraging wikipedia table schemas for knowledge graph augmentation. In *Proceedings of the 21st International Workshop on the Web and Databases, Houston, TX, USA, June 10, 2018*, pages 5:1–5:6. ACM, 2018.

[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, 2010.

[28] Lihu Chen, Simon Razniewski, and Gerhard Weikum. Knowledge base completion for long-tail entities. In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*, pages 99–108, 2023.

[29] Yuanzhe Chen, Jun Kuang, Dawei Cheng, Jianbin Zheng, Ming Gao, and Aoying Zhou. Agrikg: an agricultural knowledge graph and its applications. In *Database Systems for Advanced Applications: DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA, Chiang Mai, Thailand, April 22–25, 2019, Proceedings 24*, pages 533–537. Springer, 2019.

[30] Sulogna Chowdhury, Monireh Ebrahimi, Aaron Eberhart, and Pascal Hitzler. Memory networks for RDFS reasoning: Experiments. In *Joint Proceedings of SemREC 2022 and SMART 2022 co-located with 21st International Semantic Web Conference (ISWC 2022), Hybrid event, Hangzhou, China, October 24-27, 2022*, volume 3337 of *CEUR Workshop Proceedings*, pages 28–32. CEUR-WS.org, 2022.

[31] Cuong Xuan Chu, Simon Razniewski, and Gerhard Weikum. Tifi: Taxonomy induction for fictional domains. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2673–2679. ACM, 2019.

[32] Philipp Cimiano. *Ontology learning and population from text: algorithms, evaluation and applications*, volume 27. Springer Science & Business Media, 2006.

[33] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[34] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[35] Aba-Sah Dadzie, Daniel Preotiuc-Pietro, Danica Radovanovic, Amparo Elizabeth Cano Basave, and Katrin Weller, editors. *Proceedings of the 6th Workshop on 'Making Sense of Microposts' co-located with the 25th International World Wide Web Conference (WWW 2016), Montréal, Canada, April 11, 2016*, volume 1691 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[36] Rajarshi Das, Ameya Godbole, Dilip Kavarthapu, Zhiyu Gong, Abhishek Singhal, Mo Yu, Xiaoxiao Guo, Tian Gao, Hamed Zamani, Manzil Zaheer, and Andrew McCallum. Multi-step entity-centric information retrieval for multi-hop question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 113–118. Association for Computational Linguistics, 2019.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[38] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. Knowledge-Based Trust: Estimating the trustworthiness of web sources. *VLDB Endowment*, 8(9):938–949, 2015.

[39] Sourav Dutta and Gerhard Weikum. C3EL: A joint model for cross-document co-reference resolution and entity linking. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 846–856. The Association for Computational Linguistics, 2015.

[40] Ivan Ermilov and Axel-Cyrille Ngonga Ngomo. TAIPAN: automatic property mapping for tabular data. In *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings*, volume 10024 of *Lecture Notes in Computer Science*, pages 163–179, 2016.

[41] Angela Fahrni, Benjamin Heinzerling, Thierry Göckel, and Michael Strube. HITS' monolingual and cross-lingual entity linking system at

TAC 2013. In *Proceedings of the Sixth Text Analysis Conference, TAC 2013, Gaithersburg, Maryland, USA, November 18-19, 2013*. NIST, 2013.

[42] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web*, 9(1):77–129, 2018.

[43] Michael Färber and Achim Rettinger. Which knowledge graph is best for me? *CoRR*, abs/1809.11099, 2018.

[44] Michael Färber, Achim Rettinger, and Boulos El Asmar. On emerging entity detection. In *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings*, volume 10024 of *Lecture Notes in Computer Science*, pages 223–238, 2016.

[45] Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 7(3):46–76, 2011.

[46] Besnik Fetahu, Avishek Anand, and Maria Koutraki. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2736–2742. ACM, 2019.

[47] Tiziano Flati, Daniele Vannella, et al. Two is bigger (and better) than one: the wikipedia bitaxonomy project. In *52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 945–955, 2014.

[48] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[49] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal*, 24(6):707–730, 2015.

[50] Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, 2017.

[51] Junyang Gao, Xian Li, Yifan Ethan Xu, Bunyamin Sisman, Xin Luna Dong, and Jun Yang. Efficient knowledge graph accuracy evaluation. *Proceedings of the VLDB Endowment*, 12(11):1679–1691, 2019.

[52] Abhishek Gattani, Digvijay S Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach. *Proceedings of the VLDB Endowment*, 6(11):1126–1137, 2013.

[53] Dan Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537, 2019.

[54] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. HermiT: An OWL 2 reasoner. *J. Autom. Reason.*, 53(3):245–269, 2014.

[55] Kara Greenfield, Rajmonda Sulo Caceres, Michael Coury, Kelly Geyer, Youngjune Gwon, Jason Matterer, Alyssa C. Mensch, Cem Safak Sahin, and Olga Simek. A reverse approach to named entity extraction and linking in microposts. In *Proceedings of the 6th Workshop on 'Making Sense of Microposts' co-located with the 25th International World Wide Web Conference (WWW 2016), Montréal, Canada, April 11, 2016*, volume 1691 of *CEUR Workshop Proceedings*, pages 67–69. CEUR-WS.org, 2016.

[56] Ralph Grishman. Information extraction. *IEEE Intelligent Systems*, 30(5):8–15, 2015.

[57] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.

[58] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. Schema.org: evolution of structured data on the web. *Communications of the ACM*, 59(2):44–51, 2016.

[59] Claudio Gutiérrez and Juan F Sequeda. Knowledge graphs. *Communications of the ACM*, 64(3):96–104, 2021.

[60] Harry Halpin, Patrick J. Hayes, Jamie P. McCusker, Deborah L. McGuinness, and Henry S. Thompson. When owl:sameas isn't the same: An analysis of identity in linked data. In *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2010.

[61] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016.

[62] Qi He, Bee-Chung Chen, and Deepak Agarwal. Building The LinkedIn Knowledge Graph. `https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph`, 2016 (accessed Nov 15, 2023).

[63] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *14th conference on Computational linguistics-Volume 2*, pages 539–545, 1992.

[64] Nicolas Heist. Towards knowledge graph construction from entity co-occurrence. In *Proceedings of the EKAW Doctoral Consortium 2018 co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France, November 13, 2018*, volume 2306 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.

[65] Nicolas Heist, Sven Hertling, and Heiko Paulheim. KGrEaT: A framework to evaluate knowledge graphs via downstream tasks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*, pages 3938–3942. ACM, 2023.

[66] Nicolas Heist, Sven Hertling, Daniel Ringler, and Heiko Paulheim. Knowledge graphs on the web–an overview. *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, pages 3–22, 2020.

[67] Nicolas Heist and Heiko Paulheim. Uncovering the semantics of wikipedia categories. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 219–236. Springer, 2019.

[68] Nicolas Heist and Heiko Paulheim. Entity extraction from wikipedia list pages. In *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2020.

[69] Nicolas Heist and Heiko Paulheim. The caligraph ontology as a challenge for OWL reasoners. In *Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Event, October*

27th, 2021, volume 3123 of *CEUR Workshop Proceedings*, pages 21–31. CEUR-WS.org, 2021.

[70] Nicolas Heist and Heiko Paulheim. Information extraction from co-occurring similar entities. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3999–4009. ACM / IW3C2, 2021.

[71] Nicolas Heist and Heiko Paulheim. Transformer-based subject entity detection in Wikipedia listings. In Mehwish Alam and Michael Cochez, editors, *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2022) co-located with the 21th International Semantic Web Conference (ISWC 2022), Virtual Conference, online, October 24, 2022*, volume 3342 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

[72] Nicolas Heist and Heiko Paulheim. Nastylinker: Nil-aware scalable transformer-based entity linker. In *The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings*, volume 13870 of *Lecture Notes in Computer Science*, pages 174–191. Springer, 2023.

[73] James Hendler, Fabien Gandon, and Dean Allemang. *Semantic web for the working ontologist: Effective modeling for linked data, RDFS, and OWL*. Morgan & Claypool, 2020.

[74] Sven Hertling and Heiko Paulheim. WebIsALOD: Providing hypernymy relations extracted from the web as linked open data. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, volume 10588 of *Lecture Notes in Computer Science*, pages 111–119. Springer, 2017.

[75] Sven Hertling and Heiko Paulheim. DBkWik: A consolidated knowledge graph from thousands of wikis. In *2018 IEEE International Conference on Big Knowledge, ICBK 2018, Singapore, November 17-18, 2018*, pages 17–24. IEEE Computer Society, 2018.

[76] Sven Hertling and Heiko Paulheim. DBkWik: extracting and integrating knowledge from thousands of wikis. *Knowl. Inf. Syst.*, 62(6):2169–2190, 2020.

[77] Sven Hertling and Heiko Paulheim. DBkWik: extracting and integrating knowledge from thousands of wikis. *Knowledge and Information Systems*, 62(6):2169–2190, 2020.

[78] Sven Hertling and Heiko Paulheim. DBkWik++- multi source matching of knowledge graphs. In *Knowledge Graphs and Semantic Web - 4th*

Iberoamerican Conference and third Indo-American Conference, KGSWC 2022, Madrid, Spain, November 21-23, 2022, Proceedings, volume 1686 of *Communications in Computer and Information Science*, pages 1–15. Springer, 2022.

[79] Sven Hertling and Heiko Paulheim. Transformer based semantic relation typing for knowledge graph integration. In *The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings*, volume 13870 of *Lecture Notes in Computer Science*, pages 105–121. Springer, 2023.

[80] Pascal Hitzler. A review of the semantic web field. *Communications of the ACM*, 64(2):76–83, 2021.

[81] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial intelligence*, 194:28–61, 2013.

[82] Alexandra Hofmann, Samresh Perchani, Jan Portisch, Sven Hertling, and Heiko Paulheim. DBkWik: Towards knowledge graph creation from thousands of wikis. In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*, volume 1963 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[83] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

[84] Elwin Huaman, Amar Tauqeer, and Anna Fensel. Towards knowledge graphs validation through weighted knowledge sources. In *Knowledge Graphs and Semantic Web: Third Iberoamerican Conference and Second Indo-American Conference, KGSWC 2021, Kingsville, Texas, USA, November 22–24, 2021, Proceedings 3*, pages 47–60. Springer, 2021.

[85] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 105–113. ACM, 2019.

[86] Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, and Maria-Esther Vidal. SDM-RDFizer: An RML interpreter for the efficient creation of RDF knowledge graphs. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge*

Management, Virtual Event, Ireland, October 19-23, 2020, pages 3039–3046. ACM, 2020.

[87] Filip Ilievski. *Identity of Long-tail Entities in Text*, volume 43 of *Studies on the Semantic Web*. IOS Press, 2019.

[88] Filip Ilievski, Pedro A. Szekely, and Bin Zhang. CSKG: the commonsense knowledge graph. In *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*, volume 12731 of *Lecture Notes in Computer Science*, pages 680–696. Springer, 2021.

[89] Anastasiia Iurshina, Jiaxin Pan, Rafika Boutalbi, and Steffen Staab. Nilk: Entity linking dataset targeting NIL-linking cases. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4069–4073, 2022.

[90] Devansh Jain and Luis Espinosa Anke. Distilling hypernymy relations from language models: On the effectiveness of zero-shot taxonomy induction. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2022, Seattle, WA, USA, July 14-15, 2022*, pages 151–156. Association for Computational Linguistics, 2022.

[91] Krzysztof Janowicz, Pascal Hitzler, Benjamin Adams, Dave Kolas, and Charles Vardeman II. Five stars of linked data vocabulary use. *Semantic Web*, 5(3):173–176, 2014.

[92] Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. Overview of the TAC 2010 knowledge base population track. In *Third text analysis conference (TAC 2010)*, volume 3, page 3, 2010.

[93] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[94] Nora Kassner, Fabio Petroni, Mikhail Plekhanov, Sebastian Riedel, and Nicola Cancedda. EDIN: an end-to-end benchmark and pipeline for unknown entity discovery and indexing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 8659–8673. Association for Computational Linguistics, 2022.

[95] Mayank Kejriwal. Knowledge graphs: Constructing, completing, and effectively applying knowledge graphs in tourism. In *Applied Data Science in Tourism: Interdisciplinary Approaches, Methodologies, and Applications*, pages 423–449. Springer, 2022.

[96] Tomáš Kliegr and Ondřej Zamazal. LHD 2.0: A text mining approach to typing entities in knowledge graphs. *Journal of Web Semantics*, 39:47–61, 2016.

[97] Bhushan Kotnis, Kiril Gashteovski, Daniel Rubio, Ammar Shaker, Vanesa Rodriguez-Tembras, Makoto Takamoto, Mathias Niepert, and Carolin Lawrence. MILIE: Modular & iterative multilingual open information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6939–6950, 2022.

[98] N'Dah Jean Kouagou, Stefan Heindorf, Caglar Demir, and Axel-Cyrille Ngonga Ngomo. Neural class expression synthesis. In *The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings*, volume 13870 of *Lecture Notes in Computer Science*, pages 209–226. Springer, 2023.

[99] Zornitsa Kozareva and Eduard H. Hovy. Learning arguments and supertypes of semantic relations using recursive patterns. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 1482–1491. The Association for Computer Linguistics, 2010.

[100] Patrick Kuhn, Sven Mischkewitz, Nico Ring, and Fabian Windheuser. Type inference on wikipedia list pages. In Heinrich C. Mayr and Martin Pinzger, editors, *46. Jahrestagung der Gesellschaft für Informatik, Informatik von Menschen für Menschen, INFORMATIK 2016, Klagenfurt, Austria, September 26-30, 2016*, volume P-259 of *LNI*, pages 2101–2111. GI, 2016.

[101] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, 2016.

[102] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[103] Jens Lehmann. DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research*, 10(Nov):2639–2642, 2009.

[104] Jens Lehmann, Robert Isele, Max Jakob, et al. DBpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[105] Jens Lehmann and Johanna Voelker. An introduction to ontology learning. *Perspectives on Ontology Learning*, 18, 2014.

[106] Oliver Lehmberg and Christian Bizer. Stitching web tables for improving matching quality. *VLDB Endowment*, 10(11):1502–1513, 2017.

[107] Douglas Lenat and E Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence: Critical Concepts*, 2:298, 2000.

[108] Douglas B Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[109] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. BOND: BERT-assisted open-domain named entity recognition with distant supervision. In *26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064, 2020.

[110] 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, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187. AAAI Press, 2015.

[111] Xiao Ling, Sameer Singh, and Daniel S Weld. Design challenges for entity linking. *Transactions of the ACL*, 3:315–328, 2015.

[112] Guiliang Liu, Xu Li, Jiakang Wang, Mingming Sun, and Ping Li. Extracting knowledge from web text with monte carlo tree search. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2585–2591. ACM / IW3C2, 2020.

[113] Qiaoling Liu, Kaifeng Xu, Lei Zhang, Haofen Wang, Yong Yu, and Yue Pan. Catriple: Extracting triples from wikipedia categories. In *The Semantic Web, 3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand, December 8-11, 2008. Proceedings*, volume 5367 of *Lecture Notes in Computer Science*, pages 330–344. Springer, 2008.

[114] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2511–2522. Association for Computational Linguistics, 2023.

[115] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[116] Robert L Logan IV, Andrew McCallum, Sameer Singh, and Dan Bikel. Benchmarking scalable methods for streaming cross document entity coreference. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4717–4731, 2021.

[117] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, 2019.

[118] Erin Macdonald and Denilson Barbosa. Neural relation extraction on wikipedia tables for augmenting knowledge graphs. In *29th ACM International Conference on Information & Knowledge Management*, pages 2133–2136, 2020.

[119] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent systems*, 16(2):72–79, 2001.

[120] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org, 2015.

[121] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[122] Jose L Martinez-Rodriguez, Aidan Hogan, and Ivan Lopez-Arevalo. Information extraction meets the semantic web: a survey. *Semantic Web*, 11(2):255–335, 2020.

[123] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *28th International Joint Conference on Artificial Intelligence*, pages 3137–3143, 2019.

[124] Robert Meusel, Christian Bizer, and Heiko Paulheim. A web-scale study of the adoption and evolution of the schema.org vocabulary over time. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, page 15. ACM, 2015.

[125] Robert Meusel and Heiko Paulheim. Linked data for information extraction challenge 2014 tasks and results. In *Proceedings of the Second International Conference on Linked Data for Information Extraction-Volume 1267*, pages 3–8, 2014.

[126] Robert Meusel and Heiko Paulheim. Heuristics for fixing common errors in deployed schema.org microdata. In *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings*, volume 9088 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2015.

[127] Robert Meusel, Petar Petrovski, and Christian Bizer. The WebDataCommons microdata, RDFa and microformat dataset series. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2014.

[128] George A. Miller. WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[129] David N. Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 509–518. ACM, 2008.

[130] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011. The Association for Computer Linguistics, 2009.

[131] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha P. Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. *Commun. ACM*, 61(5):103–115, 2018.

[132] Sean Monahan, John Lehmann, Timothy Nyberg, Jesse Plymale, and Arnold Jung. Cross-lingual cross-document coreference with entity linking. In *Proceedings of the Fourth Text Analysis Conference, TAC*

2011, Gaithersburg, Maryland, USA, November 14-15, 2011. NIST, 2011.

[133] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Triplifying wikipedia's tables. In *Proceedings of the First International Workshop on Linked Data for Information Extraction (LD4IE 2013) co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 21, 2013*, volume 1057 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[134] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Using linked data to mine RDF from wikipedia's tables. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pages 533–542. ACM, 2014.

[135] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[136] Vivi Nastase and Michael Strube. Decoding wikipedia categories for knowledge acquisition. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1219–1224. AAAI Press, 2008.

[137] Roberto Navigli, Michele Bevilacqua, Simone Conia, Dario Montagnini, and Francesco Cecconi. Ten years of BabelNet: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4559–4567. ijcai.org, 2021.

[138] Roberto Navigli and Simone Paolo Ponzetto. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250, 2012.

[139] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.

[140] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816. Omnipress, 2011.

[141] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. SPrank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–34, 2016.

[142] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, 62(8):36–43, 2019.

[143] Yaser Oulabi and Christian Bizer. Using weak supervision to identify long-tail entities for knowledge base completion. In *Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019, Karlsruhe, Germany, September 9-12, 2019, Proceedings*, volume 11702 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 2019.

[144] Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. Large language models and knowledge graphs: Opportunities and challenges. *TGDK*, 1(1):2:1–2:38, 2023.

[145] Eleni Partalidou, Despina Christou, and Grigorios Tsoumakas. Improving zero-shot entity retrieval through effective dense representations. In *SETN 2022: 12th Hellenic Conference on Artificial Intelligence, Corfu Greece, September 7 - 9, 2022*, pages 30:1–30:5. ACM, 2022.

[146] Heiko Paulheim. What the adoption of schema.org tells about linked open data. In *Joint Proceedings of the 5th International Workshop on Using the Web in the Age of Data (USEWOD '15) and the 2nd International Workshop on Dataset PROFIling and fEderated Search for Linked Data (PROFILES '15) co-located with the 12th European Semantic Web Conference (ESWC 2015), Portorož, Slovenia, May 31 - June 1, 2015*, volume 1362 of *CEUR Workshop Proceedings*, pages 85–90. CEUR-WS.org, 2015.

[147] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[148] Heiko Paulheim. How much is a triple? estimating the cost of knowledge graph creation. In *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*, volume 2180 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.

[149] Heiko Paulheim and Johannes Fürnkranz. Unsupervised generation of data mining features from linked open data. In *2nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, Craiova, Romania, June 6-8, 2012*, pages 31:1–31:12. ACM, 2012.

[150] Heiko Paulheim and Simone Paolo Ponzetto. Extending dbpedia with wikipedia list pages. In *Proceedings of the NLP & DBpedia workshop co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 22, 2013*, volume 1064 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[151] Maria Angela Pellegrino, Abdulrahman Altabba, Martina Garofalo, Petar Ristoski, and Michael Cochez. GEval: a modular and extensible evaluation framework for graph embedding techniques. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, pages 565–582. Springer, 2020.

[152] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. YAGO 4: A reason-able knowledge base. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, pages 583–596. Springer, 2020.

[153] Simone Paolo Ponzetto and Roberto Navigli. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2083–2088, 2009.

[154] Simone Paolo Ponzetto and Michael Strube. Deriving a large-scale taxonomy from wikipedia. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1440–1445. AAAI Press, 2007.

[155] Jan Portisch, Michael Hladik, and Heiko Paulheim. RDF2Vec Light - A lightweight approach for knowledge graph embeddings. In *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC)*, volume 2721 of *CEUR Workshop Proceedings*, pages 79–84. CEUR-WS.org, 2020.

[156] Mina Abd Nikooie Pour et al. Results of the ontology alignment evaluation initiative 2022. In *Proceedings of the 17th International Workshop on Ontology Matching (OM 2022) co-located with the 21th International Semantic Web Conference (ISWC 2022), Hangzhou, China, held as a virtual conference, October 23, 2022*, volume 3324 of *CEUR Workshop Proceedings*, pages 84–128. CEUR-WS.org, 2022.

[157] Abhishek Pradhan, Ketan Kumar Todi, Anbarasan Selvarasu, and Atish Sanyal. Knowledge graph generation with deep active learning. In

2020 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2020.

[158] Jiaxin Qin and Mizuho Iwaihara. Annotating column type utilizing bert and knowledge graph over wikipedia categories and lists. In *DEIM Forum*, 2022.

[159] Gorjan Radevski, Kiril Gashteovski, Chia-Chien Hung, Carolin Lawrence, and Goran Glavaš. Linking surface facts to large-scale knowledge graphs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7189–7207, 2023.

[160] Will Radford, Joel Nothman, James R. Curran, Ben Hachey, and Matthew Honnibal. Naïve but effective NIL clustering baselines - CMCRC at TAC 2011. In *Proceedings of the Fourth Text Analysis Conference, TAC 2011, Gaithersburg, Maryland, USA, November 14-15, 2011*. NIST, 2011.

[161] Delip Rao, Paul McNamee, and Mark Dredze. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 93–115. Springer, 2013.

[162] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[163] Ridho Reinanda, Edgar Meij, Maarten de Rijke, et al. Knowledge graphs: An information retrieval perspective. *Foundations and Trends® in Information Retrieval*, 14(4):289–444, 2020.

[164] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web. *Data Mining and Knowledge Discovery*, 24(3):613–662, 2012.

[165] Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all? analyzing the differences between DBpedia, YAGO, Wikidata & co. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 366–372. Springer, 2017.

[166] 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 *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II 15*, pages 186–194. Springer, 2016.

[167] Petar Ristoski, Zhizhong Lin, and Qunzhi Zhou. KG-ZESHEL: knowledge graph-enhanced zero-shot entity linking. In *K-CAP '21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021*, pages 49–56. ACM, 2021.

[168] Petar Ristoski and Heiko Paulheim. RDF2vec: RDF graph embeddings for data mining. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, volume 9981 of *Lecture Notes in Computer Science*, pages 498–514, 2016.

[169] Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 251–261. ACM, 2016.

[170] Dominique Ritze and Heiko Paulheim. Towards an automatic parameterization of ontology matching tools based on example mappings. In *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*, volume 814 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[171] Giuseppe Rizzo, Bianca Pereira, Andrea Varga, Marieke Van Erp, and Amparo Elizabeth Cano Basave. Lessons learnt from the Named Entity rEcognition and linking (NEEL) challenge series. *Semantic Web*, 8(5):667–700, 2017.

[172] Ahmad Sakor, Kuldeep Singh, Anery Patel, and Maria-Esther Vidal. Falcon 2.0: An entity and relation linking tool over Wikidata. In *29th ACM International Conference on Information & Knowledge Management*, pages 3141–3148, 2020.

[173] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[174] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*, pages 245–260. Springer, 2014.

[175] Edward W Schneider. Course modularization applied: The interface system and its implications for sequence control and data analysis. In *Association for the Development of Instructional Systems*. ERIC, 1973.

[176] Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. Semeval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 341–350. The Association for Computer Linguistics, 2013.

[177] Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. A large database of hypernymy relations extracted from the web. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 360–367, 2016.

[178] Özge Sevgili, Artem Shelmanov, Mikhail Y. Arkhipov, Alexander Panchenko, and Chris Biemann. Neural entity linking: A survey of models based on deep learning. *Semantic Web*, 13(3):527–570, 2022.

[179] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2014.

[180] Amit Singhal. Introducing the knowledge graph: things, not strings. `https://www.blog.google/products/search/introducing-knowledge-graph-things-not/`, 2012 (accessed Jan 09, 2024).

[181] Robyn Speer and Catherine Havasi. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 3679–3686. European Language Resources Association (ELRA), 2012.

[182] Gabriel Stanovsky and Ido Dagan. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2300–2305, 2016.

[183] Julien Subercaze. Chaudron: Extending DBpedia with measurement. In *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, volume 10249 of *Lecture Notes in Computer Science*, pages 434–448, 2017.

[184] 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 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706. ACM, 2007.

[185] Thomas Pellissier Tanon, Denny Vrandecic, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From Freebase to Wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1419–1428. ACM, 2016.

[186] Andon Tchechmedjiev, Pavlos Fafalios, Katarina Boland, Malo Gasquet, Matthäus Zloch, Benjamin Zapilko, Stefan Dietze, and Konstantin Todorov. ClaimsKG: A knowledge graph of fact-checked claims. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 309–324. Springer, 2019.

[187] 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*, volume 10587 of *Lecture Notes in Computer Science*, pages 694–710. Springer, 2017.

[188] Alberto Tonon, Victor Felder, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. VoldemortKG: Mapping schema.org and web entities to linked open data. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*, volume 9982 of *Lecture Notes in Computer Science*, pages 220–228, 2016.

[189] Gerald Töpper, Magnus Knuth, and Harald Sack. Dbpedia ontology enrichment for inconsistency detection. In *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*, pages 33–40. ACM, 2012.

[190] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

[191] 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 *Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4373–4379, 2016.

[192] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[193] Paola Velardi, Stefano Faralli, and Roberto Navigli. OntoLearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707, 2013.

[194] Johanna Völker and Mathias Niepert. Statistical schema induction. In *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*, volume 6643 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 2011.

[195] Denny Vrandecic and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.

[196] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[197] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. Learning intents behind interactions with knowledge graph for recommendation. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 878–887. ACM / IW3C2, 2021.

[198] Xiangyu Wang, Lyuzhou Chen, Taiyu Ban, Muhammad Usman, Yifeng Guan, Shikang Liu, Tianhao Wu, and Huanhuan Chen. Knowledge graph quality control: A survey. *Fundamental Research*, 1(5):607–626, 2021.

[199] Zhaoyi Wang, Zhenyang Zhang, Jiaxin Qin, and Mizuho Iwaihara. SLHCat: Mapping wikipedia categories and lists to DBpedia by leveraging semantic, lexical, and hierarchical features. In *Leveraging Generative Intelligence in Digital Libraries: Towards Human-Machine Collaboration - 25th International Conference on Asia-Pacific Digital Libraries, ICADL 2023, Taipei, Taiwan, December 4-7, 2023, Proceedings, Part I*, volume 14457 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2023.

[200] Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang. Retrieval, re-ranking and multi-task learning for knowledge-base question answering. In *Proceedings of the 16th Conference of the European Chapter*

of the Association for Computational Linguistics: Main Volume, pages 347–357, 2021.

[201] Gerhard Weikum. Knowledge graphs 2021: A data odyssey. *Proc. VLDB Endow.*, 14(12):3233–3238, 2021.

[202] Gerhard Weikum, Xin Luna Dong, Simon Razniewski, Fabian Suchanek, et al. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends® in Databases*, 10(2-4):108–490, 2021.

[203] Wikimedia. List of wikipedia editions. `https://en.wikipedia.org/wiki/List_of_Wikipedias`, accessed Jan 09, 2024.

[204] Wikimedia. Wikimedia language proposal policy. `https://meta.wikimedia.org/wiki/Language_proposal_policy`, accessed Jan 09, 2024.

[205] Wikimedia. Wikipedia size. `https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`, accessed Jan 09, 2024.

[206] Wikimedia. Wikidata statistics. `https://wikidata-todo.toolforge.org/stats.php`, accessed Nov 02, 2023.

[207] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics, 2020.

[208] Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 41–50. ACM, 2007.

[209] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 118–127. The Association for Computer Linguistics, 2010.

[210] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, 2020.

[211] Bo Xu, Chenhao Xie, Yi Zhang, Yanghua Xiao, Haixun Wang, and Wei Wang. Learning defining features for categories. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3924–3930. IJCAI/AAAI Press, 2016.

[212] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. InfoGather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 97–108. ACM, 2012.

[213] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[214] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.

[215] Shuo Zhang and Krisztian Balog. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–35, 2020.

[216] Shuo Zhang, Krisztian Balog, and Jamie Callan. Generating categories for sets of entities. In *29th ACM International Conference on Information & Knowledge Management*, pages 1833–1842, 2020.

[217] Shuo Zhang, Edgar Meij, Krisztian Balog, and Ridho Reinanda. Novel entity discovery from web tables. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 1298–1308. ACM / IW3C2, 2020.

[218] Ziqi Zhang. Towards efficient and effective semantic table interpretation. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*, pages 487–502. Springer, 2014.

[219] Lingyun Zhao, Lin Li, Xinhao Zheng, and Jianwei Zhang. A BERT based sentiment analysis and key entity detection approach for online financial texts. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 1233–1238. IEEE, 2021.

[220] Tong Zhao, Julian McAuley, and Irwin King. Improving latent factor models via personalized feature projection for one class recommendation. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 821–830, 2015.

[221] Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. DGL-KE: training knowledge graph embeddings at scale. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 739–748. ACM, 2020.

[222] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A comprehensive survey on automatic knowledge graph construction. *CoRR*, abs/2302.05019, 2023.

[223] Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *CoRR*, abs/2305.13168, 2023.

APPENDIX A

Data Sources

This chapter provides links to the data sources used in experiments that compare KGs.

A.1 Data Sources for Knowledge Graph Comparison

For the comparison of KGs in Chapter 3, we used the following data sources:

A.1.1 DBpedia

Version 2016-10

- http://downloads.dbpedia.org/2016-10/dbpedia_2016-10.owl
- http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_en.ttl.bz2
- http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_transitive_en.ttl.bz2
- http://downloads.dbpedia.org/2016-10/core-i18n/en/interlanguage_links_en.ttl.bz2
- http://downloads.dbpedia.org/2016-10/core-i18n/en/labels_en.ttl.bz2
- http://downloads.dbpedia.org/2016-10/core-i18n/en/mappingbased_literals_en.ttl.bz2
- http://downloads.dbpedia.org/2016-10/core-i18n/en/mappingbased_objects_en.ttl.bz2

A.1.2 YAGO

Version 3.1

- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoTransitiveType.ttl.7z

- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoSchema.ttl.7z
- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoTypes.ttl.7z
- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoTaxonomy.ttl.7z
- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoLiteralFacts.ttl.7z
- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoLabels.ttl.7z
- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoDateFacts.ttl.7z
- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoFacts.ttl.7z
- http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoDBpediaInstances.ttl.7z

A.1.3 Wikidata

Version 20190628
The original version is not available anymore, but the most recent dump can be found here:

- https://dumps.wikimedia.org/wikidatawiki/entities/latest-truthy.nt.gz

A.1.4 BabelNet

Version 3.6
This version is not publicly available for download but has been provided by the developers upon request.

A.1.5 NELL

Version 0.3#1100

- http://wdaqua-nell2rdf.univ-st-etienne.fr/archive/NELL2RDF_0.3_1100_ontology.ttl.gz
- http://wdaqua-nell2rdf.univ-st-etienne.fr/archive/NELL2RDF_0.3_1100.nt.gz

A.1.6 OpenCyc

Version 4.0

- https://github.com/asanchez75/opencyc/blob/master/opencyc-latest.owl.gz

A.1.7 VoldemortKG

- http://voldemort.exascale.info

A.2 Data Sources for KGrEaT

For the comparison of KGs with KGrEaT in Chapters 10 and 11, we used the following additional data sources:

A.2.1 DBpedia

Version 2022-09

- http://akswnc7.informatik.uni-leipzig.de/dstreitmatter/archivo/dbpedia.org/ontology--DEV/2022.10.09-192003/ontology--DEV_type=parsed.nt
- https://downloads.dbpedia.org/repo/dbpedia/mappings/instance-types/2022.09.01/instance-types_lang=en_specific.ttl.bz2
- https://downloads.dbpedia.org/repo/dbpedia/generic/labels/2022.09.01/labels_lang=en.ttl.bz2
- https://downloads.dbpedia.org/repo/dbpedia/mappings/mappingbased-literals/2022.09.01/mappingbased-literals_lang=en.ttl.bz2
- https://downloads.dbpedia.org/repo/dbpedia/mappings/mappingbased-objects/2022.09.01/mappingbased-objects_lang=en.ttl.bz2

A.2.2 Wikidata

Version 20230607
The original version is not available anymore, but the most recent dump can be found here:

- https://dumps.wikimedia.org/wikidatawiki/entities/latest-truthy.nt.gz

A.2.3 DBkWik

Version 2.0

- https://figshare.com/articles/dataset/DBkWik_Plus_Plus/20407864

A.2.4 CaLiGraph

Version 1.1.0 (extracted from Wikipedia2016)

- https://doi.org/10.5281/zenodo.4050308

Version 2.1.1 (extracted from Wikipedia2020)

- https://doi.org/10.5281/zenodo.5524052

Version 3.1.1 (extracted from Wikipedia2022)

- https://doi.org/10.5281/zenodo.8068322

APPENDIX B

Experimental Details for KGrEaT

In this chapter, we provide the experiment details for evaluations with KGrEaT. The results are given for individual tasks, datasets, and metrics.

B.1 Results for General-Purpose Knowledge Graphs

Table B.1 shows the coverage of the evaluated KGs for the task datasets. Tables B.2 to B.4 hold the performance values for the evaluated tasks.

B.2 Results for CaLiGraph

Tables B.5 and B.6 show the results of the KGrEaT evaluation for the individual datasets in the precision- and recall-oriented scenarios.

Dataset	DBP16			DBP22			YAGO			WD			CLG			DBkWik		
	PK	PA	RA	PK	PA	RA	PK	PA	RA	PK	PA	RA	PK	PA	RA	PK	PA	RA
Cities	97	97	96	87	87	88	96	96	100	67	67	75	93	93	100	98	98	98
Forbes	87	87	87	81	81	81	99	99	100	85	85	92	91	91	100	93	93	92
AAUP	99	99	98	88	88	88	99	99	100	65	65	88	94	94	99	99	99	99
MetacriticMovies	98	98	98	95	95	94	90	90	100	87	87	85	98	98	100	99	99	99
MetacriticAlbums	99	99	99	97	97	97	95	95	100	70	70	91	96	96	100	99	99	99
MillionSongDataset†	6	6	22	6	6	21	11	11	51	11	11	40	20	20	64	13	13	47
Teams	100	100	100	94	94	94	77	77	83	94	94	96	94	94	95	100	100	100
ComicCharacters†	0	0	22	0	0	17	0	0	59	0	0	31	0	0	62	92	92	96
CitiesAndCountries	100	100	100	95	95	95	96	96	100	46	46	74	96	96	100	100	100	100
Cities2000AndCountries	100	100	100	93	93	93	94	94	99	61	61	84	94	94	100	100	100	100
CitiesMoviesAlbumsCompaniesUni	90	90	88	85	85	85	94	94	100	72	72	81	88	88	99	97	97	97
LP50	90	90	90	88	88	88	83	83	99	55	55	71	92	92	100	93	93	93
KORE	100	100	100	100	100	100	102	102	102	100	100	101	102	102	102	100	100	100
CurrencyEntities	100	100	100	93	93	93	100	100	100	34	34	41	97	97	100	100	100	100
CityStateEntities	96	96	97	97	97	97	99	99	100	31	31	54	82	82	100	97	97	100
CapitalCountryEntities	100	100	100	100	100	100	100	100	100	30	30	22	100	100	100	100	100	100
AllCapitalCountryEntities	99	99	99	96	96	97	99	99	100	44	44	54	96	96	99	100	100	100
MovieLens†	16	16	62	15	15	60	14	14	92	1	1	73	15	15	96	43	43	84
LibraryThing†	18	18	42	18	18	41	21	21	84	12	12	63	28	28	92	25	25	75
LastFm†	94	94	94	91	91	92	94	94	99	81	81	86	93	93	99	96	96	97

Table B.1: Dataset coverage (per cent) of the KGs evaluated with KGrEaT for the PK, PA, and RA scenarios. Datasets marked with a dagger are independent of DBpedia.

Task Type	Dataset	Metric	DBP16	DBP22	YAGO	WD	CLG	DBkWik
Classification	Cities	Accuracy ↑	**0.614**	0.601	0.596	0.506	0.612	0.600
	Forbes	Accuracy ↑	0.523	0.529	0.525	**0.543**	0.494	0.499
	AAUP	Accuracy ↑	0.549	0.562	0.550	**0.583**	0.561	0.539
	MetacriticMovies	Accuracy ↑	0.621	0.599	0.618	**0.627**	0.592	0.595
	MetacriticAlbums	Accuracy ↑	**0.627**	0.571	0.562	0.578	0.561	0.584
	ComicCharacters	Accuracy ↑	–	–	–	–	–	**0.462**
	MillionSongDataset	Accuracy ↑	**0.524**	0.505	0.502	0.518	0.480	0.492
Regression	Cities	RMSE ↓	0.912	**0.839**	0.943	0.840	1.069	0.993
	Forbes	RMSE ↓	0.707	0.700	0.697	**0.679**	0.699	0.707
	AAUP	RMSE ↓	0.691	0.666	0.684	0.681	**0.650**	0.692
	MetacriticMovies	RMSE ↓	**0.574**	0.596	0.575	0.591	**0.574**	0.598
	MetacriticAlbums	RMSE ↓	**0.581**	0.636	0.631	0.630	0.617	0.599
Clustering	Teams	Accuracy ↑	0.930	0.889	0.910	**0.947**	0.926	0.915
		ARI ↑	0.040	0.025	**0.041**	0.012	0.026	0.013
		NMI ↑	0.033	0.021	**0.036**	0.010	0.020	0.008
	ComicCharacters	Accuracy ↑	0.667	0.667	0.648	0.500	**0.672**	0.409
		ARI ↑	0.000	0.000	0.021	0.000	**0.099**	0.002
		NMI ↑	**0.389**	**0.389**	0.208	0.258	0.171	0.002
	CitiesAndCountries	Accuracy ↑	0.785	0.766	**0.798**	0.591	0.713	0.780
		ARI ↑	0.188	0.124	**0.220**	0.000	0.011	0.175
		NMI ↑	0.191	0.137	**0.230**	0.077	0.049	0.165
	Cities2000AndCountries	Accuracy ↑	0.705	0.685	0.715	**0.765**	0.616	0.699
		ARI ↑	0.284	0.254	**0.348**	0.209	0.165	0.270
		NMI ↑	0.268	0.241	**0.321**	0.216	0.154	0.256
	CitiesMoviesAlbums-	Accuracy ↑	**0.725**	0.692	0.701	0.589	0.654	0.652
	CompaniesUni	ARI ↑	**0.609**	0.547	0.571	0.311	0.471	0.498
		NMI ↑	**0.632**	0.573	0.595	0.330	0.500	0.502
Doc. Sim.	LP50	Spearman ↑	0.207	**0.226**	0.165	0.131	0.204	0.200
		Pearson ↑	0.294	**0.306**	0.235	0.241	0.279	0.274
		Harm. Mean ↑	0.241	**0.257**	0.184	0.172	0.233	0.164
Ent. Rel.	KORE	Kendall's Tau ↑	0.135	0.179	0.012	**0.203**	0.096	0.134
Sem. Ana.	CurrencyEnts	Accuracy ↑	**0.156**	0.142	0.142	0.000	0.072	0.093
	CityStateEnts	Accuracy ↑	0.166	0.149	0.109	0.000	**0.218**	0.118
	CapitalCountryEnts	Accuracy ↑	0.337	**0.375**	0.345	0.000	0.276	0.331
	AllCapitalCountryEnts	Accuracy ↑	0.353	**0.394**	0.287	0.002	0.275	0.319
Recom.	MovieLens	F1-score ↑	0.006	0.009	0.007	**0.031**	0.009	0.004
	LibraryThing	F1-score ↑	0.017	0.013	0.007	**0.018**	0.007	0.012
	LastFm	F1-score ↑	**0.021**	0.019	0.011	0.014	0.019	0.017

Table B.2: KGrEaT evaluation results of the KGs aggregated by task type, dataset and metric for the *PK* scenario.

Task Type	Dataset	Metric	DBP16	DBP22	YAGO	WD	CLG	DBkWik
Classification	Cities	Accuracy ↑	**0.596**	0.525	0.570	0.341	0.572	0.589
	Forbes	Accuracy ↑	0.456	0.429	**0.518**	0.459	0.449	0.464
	AAUP	Accuracy ↑	**0.542**	0.492	0.542	0.381	0.527	0.534
	MetacriticMovies	Accuracy ↑	**0.610**	0.570	0.556	0.547	0.580	0.590
	MetacriticAlbums	Accuracy ↑	**0.619**	0.552	0.533	0.406	0.538	0.579
	ComicCharacters	Accuracy ↑	–	–	–	–	–	**0.434**
	MillionSongDataset	Accuracy ↑	0.035	0.033	0.066	0.063	**0.109**	0.077
Regression	Cities	RMSE ↓	1.584	**1.559**	1.811	1.582	1.725	1.575
	Forbes	RMSE ↓	1.262	1.269	1.280	1.310	1.301	**1.237**
	AAUP	RMSE ↓	**1.355**	1.372	1.358	1.387	1.395	1.426
	MetacriticMovies	RMSE ↓	**1.074**	1.097	1.075	1.091	**1.074**	1.098
	MetacriticAlbums	RMSE ↓	**1.080**	1.137	1.131	1.131	1.118	1.099
Clustering	Teams	Accuracy ↑	**0.930**	0.833	0.700	0.889	0.866	0.915
		ARI ↑	**0.082**	0.059	0.007	0.067	0.053	0.045
		NMI ↑	**0.069**	0.057	0.008	0.056	0.049	0.037
	ComicCharacters	Accuracy ↑	0.000	0.000	0.000	0.000	0.000	**0.383**
		ARI ↑	0.000	0.000	0.000	0.000	0.000	**0.004**
		NMI ↑	**0.183**	**0.183**	**0.183**	**0.183**	**0.183**	0.062
	CitiesAndCountries	Accuracy ↑	**0.783**	0.726	0.770	0.271	0.683	0.779
		ARI ↑	0.184	0.147	**0.255**	0.000	0.045	0.171
		NMI ↑	0.188	0.151	**0.230**	0.077	0.090	0.163
	Cities2000AndCountries	Accuracy ↑	**0.705**	0.636	0.672	0.466	0.579	0.699
		ARI ↑	0.284	0.214	**0.300**	0.170	0.141	0.270
		NMI ↑	0.267	0.213	**0.271**	0.180	0.157	0.256
	CitiesMoviesAlbums-	Accuracy ↑	0.652	0.586	**0.657**	0.426	0.575	0.631
	CompaniesUni	ARI ↑	**0.541**	0.459	0.519	0.222	0.416	0.476
		NMI ↑	0.530	0.461	**0.532**	0.322	0.435	0.482
Doc. Sim.	LP50	Spearman ↑	0.207	**0.226**	0.165	0.165	0.204	0.200
		Pearson ↑	0.294	**0.306**	0.235	0.069	0.279	0.274
		Harm. Mean ↑	0.241	**0.257**	0.184	0.103	0.233	0.164
Ent. Rel.	KORE	Kendall's Tau ↑	0.104	0.108	0.015	0.071	0.068	**0.119**
Sem. Ana.	CurrencyEnts	Accuracy ↑	**0.156**	0.123	0.142	0.000	0.067	0.093
	CityStateEnts	Accuracy ↑	**0.143**	0.136	0.106	0.000	0.127	0.105
	CapitalCountryEnts	Accuracy ↑	0.337	**0.375**	0.345	0.000	0.276	0.331
	AllCapitalCountryEnts	Accuracy ↑	0.346	**0.360**	0.282	0.000	0.252	0.319
Recom.	MovieLens	F1-score ↑	0.004	**0.005**	0.004	**0.005**	**0.005**	**0.005**
	LibraryThing	F1-score ↑	**0.006**	**0.006**	0.003	0.002	**0.006**	**0.006**
	LastFm	F1-score ↑	**0.022**	0.021	0.012	0.011	0.019	0.018

Table B.3: KGrEaT evaluation results of the KGs aggregated by task type, dataset and metric for the *PA* scenario.

Task Type	Dataset	Metric	DBP16	DBP22	YAGO	WD	CLG	DBkWik
Classification	Cities	Accuracy ↑	0.579	0.524	**0.599**	0.402	**0.599**	0.590
	Forbes	Accuracy ↑	0.456	0.429	**0.514**	0.474	0.491	0.463
	AAUP	Accuracy ↑	0.535	0.494	0.534	0.503	**0.555**	0.535
	MetacriticMovies	Accuracy ↑	**0.610**	0.568	0.590	0.517	0.591	0.573
	MetacriticAlbums	Accuracy ↑	**0.621**	0.552	0.553	0.520	0.560	0.577
	ComicCharacters	Accuracy ↑	0.112	0.074	0.443	0.185	0.447	**0.463**
	MillionSongDataset	Accuracy ↑	0.129	0.115	**0.457**	0.249	0.447	0.305
Regression	Cities	RMSE ↓	1.727	1.549	**1.003**	1.507	1.461	1.886
	Forbes	RMSE ↓	1.261	1.269	**0.699**	1.312	0.708	1.240
	AAUP	RMSE ↓	1.461	1.372	0.685	1.462	**0.663**	1.424
	MetacriticMovies	RMSE ↓	1.073	1.092	0.583	1.105	**0.578**	1.101
	MetacriticAlbums	RMSE ↓	1.080	1.138	0.628	1.121	**0.618**	1.098
Clustering	Teams	Accuracy ↑	0.930	0.849	0.915	**0.937**	0.920	0.915
		ARI ↑	**0.082**	0.078	0.036	0.081	0.015	0.045
		NMI ↑	0.069	**0.073**	0.031	0.062	0.010	0.037
	ComicCharacters	Accuracy ↑	0.104	0.067	0.411	0.170	**0.428**	0.408
		ARI ↑	0.000	0.000	0.003	0.002	**0.004**	0.002
		NMI ↑	0.168	**0.172**	0.004	0.163	0.003	0.002
	CitiesAndCountries	Accuracy ↑	0.783	0.747	0.791	0.560	0.692	**0.792**
		ARI ↑	0.184	0.184	**0.215**	0.000	0.005	0.202
		NMI ↑	0.188	0.166	**0.219**	0.071	0.048	0.181
	Cities2000AndCountries	Accuracy ↑	0.705	0.636	**0.713**	0.621	0.621	0.699
		ARI ↑	0.284	0.212	**0.313**	0.201	0.160	0.269
		NMI ↑	0.267	0.209	**0.290**	0.198	0.137	0.256
	CitiesMoviesAlbums-	Accuracy ↑	0.643	0.587	**0.667**	0.456	0.618	0.632
	CompaniesUni	ARI ↑	**0.536**	0.460	0.499	0.233	0.424	0.477
		NMI ↑	0.521	0.462	**0.521**	0.321	0.443	0.482
Doc. Sim.	LP50	Spearman ↑	0.207	**0.226**	0.160	0.102	0.154	0.214
		Pearson ↑	0.294	**0.306**	0.233	0.228	0.233	0.283
		Harm. Mean ↑	0.241	**0.257**	0.191	0.149	0.184	0.241
Ent. Rel.	KORE	Kendall's Tau ↑	0.109	0.119	0.008	**0.130**	0.084	0.117
Sem. Ana.	CurrencyEnts	Accuracy ↑	**0.154**	0.133	0.133	0.000	0.090	0.083
	CityStateEnts	Accuracy ↑	0.153	0.130	0.098	0.000	**0.157**	0.120
	CapitalCountryEnts	Accuracy ↑	**0.369**	0.362	0.332	0.000	0.327	0.321
	AllCapitalCountryEnts	Accuracy ↑	**0.369**	0.361	0.292	0.000	0.294	0.299
Recom.	MovieLens	F1-score ↑	**0.004**	0.003	0.003	**0.004**	0.003	**0.004**
	LibraryThing	F1-score ↑	**0.006**	0.005	0.003	0.005	0.005	**0.006**
	LastFm	F1-score ↑	**0.022**	0.021	0.012	0.010	0.018	0.018

Table B.4: KGrEaT evaluation results of the KGs aggregated by task type, dataset and metric for the *RA* scenario.

Task Type	Dataset	Metric	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
Classification	Cities	Accuracy ↑	0.801	**0.806**	0.768	0.764	0.656	0.787
	Forbes	Accuracy ↑	**0.617**	0.599	0.604	0.580	0.556	0.584
	AAUP	Accuracy ↑	0.633	**0.668**	0.630	0.587	0.554	0.644
	MetacriticMovies	Accuracy ↑	0.739	0.746	**0.764**	0.720	0.712	0.723
	MetacriticAlbums	Accuracy ↑	**0.764**	0.660	0.654	0.637	0.613	0.658
	ComicCharacters	Accuracy ↑	–	–	–	–	–	–
	MillionSongDataset	Accuracy ↑	**0.635**	0.616	0.606	0.614	0.629	0.617
Regression	Cities	RMSE ↓	**0.495**	0.500	0.545	0.535	0.606	0.511
	Forbes	RMSE ↓	0.582	0.587	0.582	0.596	0.605	**0.576**
	AAUP	RMSE ↓	0.576	0.528	0.571	0.580	0.591	**0.524**
	MetacriticMovies	RMSE ↓	0.469	0.467	0.465	0.460	0.466	**0.459**
	MetacriticAlbums	RMSE ↓	**0.462**	0.533	0.538	0.534	0.549	0.514
Clustering	Teams	Accuracy ↑	0.996	**0.999**	0.994	0.995	0.997	0.998
		ARI ↑	0.259	**0.333**	0.063	0.052	0.039	0.249
		NMI ↑	0.215	**0.285**	0.056	0.042	0.030	0.211
	ComicCharacters	Accuracy ↑	0.667	0.667	**0.875**	0.800	0.800	**0.875**
		ARI ↑	0.000	0.000	0.495	0.231	0.000	**0.505**
		NMI ↑	**0.734**	**0.734**	0.562	0.380	0.101	0.529
	CitiesAndCountries	Accuracy ↑	0.935	0.940	**0.982**	0.790	0.898	0.800
		ARI ↑	0.740	0.755	**0.916**	0.071	0.614	0.037
		NMI ↑	0.657	0.658	**0.831**	0.205	0.545	0.203
	Cities2000AndCountries	Accuracy ↑	**0.975**	0.972	0.962	0.956	0.972	0.956
		ARI ↑	**0.902**	0.891	0.854	0.833	0.892	0.830
		NMI ↑	**0.841**	0.818	0.777	0.745	0.818	0.744
	CitiesMoviesAlbums-	Accuracy ↑	**0.994**	0.970	0.990	0.982	0.932	0.959
	CompaniesUni	ARI ↑	**0.988**	0.942	0.976	0.958	0.892	0.893
		NMI ↑	**0.975**	0.915	0.958	0.933	0.867	0.887
Doc. Sim.	LP50	Spearman ↑	0.412	0.342	**0.447**	0.433	0.440	0.384
		Pearson ↑	0.631	0.596	0.658	0.655	**0.660**	0.613
		Harm. Mean ↑	0.492	0.431	**0.532**	0.521	0.528	0.472
Ent. Rel.	KORE	Kendall's Tau ↑	0.440	**0.441**	0.299	0.225	0.197	0.389
Sem. Ana.	CurrencyEnts	Accuracy ↑	0.368	0.356	**0.465**	0.037	0.031	0.182
	CityStateEnts	Accuracy ↑	0.500	0.453	0.212	0.480	0.072	**0.721**
	CapitalCountryEnts	Accuracy ↑	**0.958**	0.881	0.798	0.919	0.435	0.923
	AllCapitalCountryEnts	Accuracy ↑	0.911	0.884	0.697	0.880	0.643	**0.929**
Recom.	MovieLens	F1-score ↑	0.013	0.019	0.022	0.023	0.022	**0.033**
	LibraryThing	F1-score ↑	**0.047**	0.031	0.016	0.021	0.012	0.019
	LastFm	F1-score ↑	**0.050**	0.040	0.022	0.029	0.021	0.042

Table B.5: KGrEaT evaluation results of the KGs aggregated by task type, dataset and metric for the *precision-oriented* mapping scenario.

Task Type	Dataset	Metric	DBP16	DBP22	YAGO3	CLGv1	CLGv2	CLGv3
Classification	Cities	Accuracy ↑	0.751	0.698	0.760	**0.807**	0.656	0.760
	Forbes	Accuracy ↑	0.537	0.486	**0.590**	0.586	0.546	0.556
	AAUP	Accuracy ↑	0.612	0.584	0.588	0.595	0.557	**0.634**
	MetacriticMovies	Accuracy ↑	**0.725**	0.709	0.691	0.719	0.710	0.718
	MetacriticAlbums	Accuracy ↑	**0.751**	0.638	0.625	0.632	0.613	0.657
	ComicCharacters	Accuracy ↑	0.119	0.081	0.478	**0.484**	0.476	0.475
	MillionSongDataset	Accuracy ↑	0.156	0.139	0.584	0.584	0.588	**0.589**
Regression	Cities	RMSE ↓	1.318	1.203	0.558	0.539	0.623	**0.516**
	Forbes	RMSE ↓	1.137	1.156	**0.595**	0.596	0.607	**0.595**
	AAUP	RMSE ↓	1.347	1.227	0.572	0.578	0.596	**0.540**
	MetacriticMovies	RMSE ↓	0.969	0.965	0.482	**0.461**	0.468	0.462
	MetacriticAlbums	RMSE ↓	0.962	1.035	0.547	0.534	0.546	**0.518**
Clustering	Teams	Accuracy ↑	**0.996**	0.954	0.994	0.995	0.995	0.995
		ARI ↑	**0.318**	0.151	0.061	0.043	0.050	0.090
		NMI ↑	**0.260**	0.106	0.054	0.040	0.035	0.075
	ComicCharacters	Accuracy ↑	0.115	0.076	0.473	0.460	0.465	**0.474**
		ARI ↑	0.000	0.000	0.016	0.016	0.014	**0.018**
		NMI ↑	0.171	**0.174**	0.008	0.008	0.006	0.006
	CitiesAndCountries	Accuracy ↑	0.933	0.893	**0.969**	0.791	0.898	0.792
		ARI ↑	0.733	0.676	**0.863**	0.065	0.609	0.030
		NMI ↑	0.641	0.473	**0.756**	0.204	0.523	0.199
	Cities2000And-	Accuracy ↑	**0.974**	0.903	0.934	0.951	0.965	0.936
	Countries	ARI ↑	**0.900**	0.764	0.754	0.814	0.865	0.761
		NMI ↑	**0.836**	0.573	0.680	0.721	0.783	0.658
	CitiesMoviesAlbums-	Accuracy ↑	0.880	0.823	0.941	**0.963**	0.877	0.916
	CompaniesUni	ARI ↑	0.856	0.758	0.871	**0.919**	0.827	0.871
		NMI ↑	0.743	0.650	0.845	**0.882**	0.727	0.841
Doc. Sim.	LP50	Spearman ↑	0.412	0.342	**0.451**	0.443	0.444	0.386
		Pearson ↑	0.631	0.596	**0.670**	0.669	0.659	0.618
		Harm. Mean ↑	0.492	0.431	**0.539**	0.533	0.531	0.475
Ent. Rel.	KORE	Kendall's Tau ↑	**0.386**	0.333	0.278	0.221	0.208	0.372
Sem. Ana.	CurrencyEnts	Accuracy ↑	0.355	0.322	**0.436**	0.038	0.027	0.202
	CityStateEnts	Accuracy ↑	0.440	0.409	0.214	0.444	0.066	**0.484**
	CapitalCountryEnts	Accuracy ↑	**0.990**	0.844	0.800	0.925	0.490	0.923
	AllCapitalCountryEnts	Accuracy ↑	**0.928**	0.818	0.719	0.831	0.624	0.857
Recom.	MovieLens	F1-score ↑	0.013	0.010	0.011	0.011	0.011	**0.016**
	LibraryThing	F1-score ↑	0.013	0.013	0.006	0.009	0.007	**0.014**
	LastFm	F1-score ↑	**0.052**	0.048	0.021	0.029	0.024	0.041

Table B.6: KGrEaT evaluation results of the KGs aggregated by task type, dataset and metric for the *recall-oriented* mapping scenario.