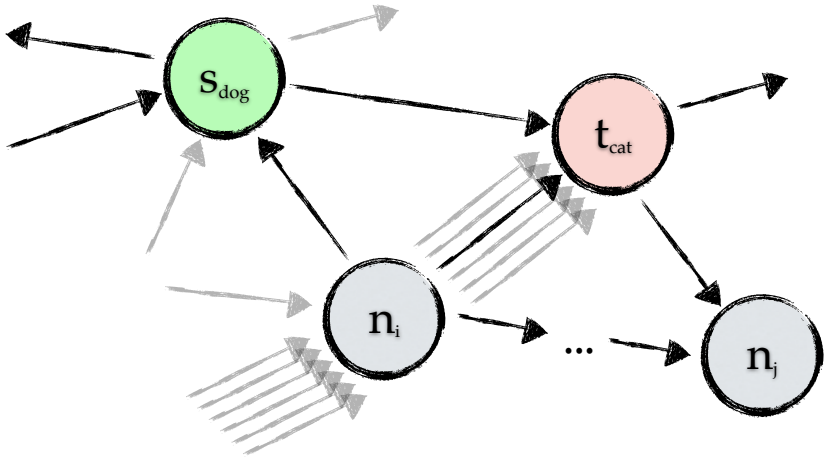
DISSERTATION

SIMULATING HUMAN ASSOCIATIONS WITH LINKED DATA

End-to-End Learning of Graph Patterns with an Evolutionary Algorithm



Thesis approved by the
Department of Computer Science
of the TU Kaiserslautern
for the award of the Doctoral Degree

DOCTOR OF NATURAL SCIENCES
(DR. RER. NAT.)

to

Jörn Hees

Date of the viva: 2018-04-09
Dean: Prof. Dr. Stefan Deßloch

Reviewers:
Prof. Dr. Prof. h.c. Andreas Dengel
Prof. Dr. Heiko Paulheim (University of Mannheim)

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN
D 386

Jörn Hees: *Simulating Human Associations with Linked Data* – End-to-End Learning of Graph Patterns with an Evolutionary Algorithm

SUPERVISORS:
Prof. Dr. Prof. h.c. Andreas Dengel
Prof. Dr. Heiko Paulheim (University of Mannheim)

SUPPLEMENTAL MATERIAL:
https://w3id.org/associations or
http://purl.org/associations

CONTACT INFORMATION:
http://joernhees.de
diss@joernhees.de

ABSTRACT

In recent years, enormous progress has been made in the field of Artificial Intelligence (AI). Especially the introduction of Deep Learning and end-to-end learning, the availability of large datasets and the necessary computational power in form of specialised hardware allowed researchers to build systems with previously unseen performance in areas such as computer vision, machine translation and machine gaming. In parallel, the Semantic Web and its Linked Data movement have published many interlinked RDF datasets, forming the world's largest, decentralised and publicly available knowledge base.

Despite these scientific successes, all current systems are still narrow AI systems. Each of them is specialised to a specific task and cannot easily be adapted to all other human intelligence tasks, as would be necessary for Artificial General Intelligence (AGI). Furthermore, most of the currently developed systems are not able to learn by making use of freely available knowledge such as provided by the Semantic Web. Autonomous incorporation of new knowledge is however one of the pre-conditions for human-like problem solving.

This work provides a small step towards teaching machines such human-like reasoning on freely available knowledge from the Semantic Web. We investigate how human associations, one of the building blocks of our thinking, can be simulated with Linked Data. The two main results of these investigations are a ground truth dataset of semantic associations and a machine learning algorithm that is able to identify patterns for them in huge knowledge bases.

The ground truth dataset of semantic associations consists of DBpedia entities that are known to be strongly associated by humans. The dataset is published as RDF and can be used for future research.

The developed machine learning algorithm is an evolutionary algorithm that can learn SPARQL queries from a given SPARQL endpoint based on a given list of exemplary source-target entity pairs. The algorithm operates in an end-to-end learning fashion, extracting features in form of graph patterns without the need for human intervention. The learned patterns form a feature space adapted to the given list of examples and can be used to predict target candidates from the SPARQL endpoint for new source nodes. On our semantic association ground truth dataset, our evolutionary graph pattern learner reaches a Recall@10 of $> 63\,\%$ and an MRR (& MAP) $> 43\,\%$, outperforming all baselines. With an achieved Recall@1 of $> 34\%$ it even reaches average human top response prediction performance. We also demonstrate how the graph pattern learner can be applied to other interesting areas without modification.

ACKNOWLEDGMENTS

This PhD thesis would not have been possible without the support of countless people.

Supervisors

First, I would like to thank Prof. Andreas Dengel for the opportunity to conduct my research. Without his ongoing support, supervision, feedback, the freedom to investigate different approaches, and gentle nudges in the right direction, this thesis would not have been possible. Further, I would like to thank Prof. Heiko Paulheim for becoming my external supervisor towards the end of this thesis. Despite the short time, his deep insights, invaluable feedback, fruitful discussions and many great ideas vastly improved this thesis. Finally, I would like to thank Prof. Karsten Berns, my early mentor in the PhD program for his initial guidance and feedback on my research and later for becoming the head of my PhD commission and providing valuable external feedback.

DFKI

I would also like to thank the DFKI, my colleagues and students, starting with my office mates Ralf, Bahaa, Benjamin, Damian, Joachim and Rouven. Besides being my first real office mate in DFKI and introducing me to fancy eye-tracking research, the many discussions with Ralf led to the first conceptual ideas and research questions for this thesis, such as how to rank triples by association strengths, and how to collect such information with GWAPs. Later, Bahaa gave me valuable insights into the world of semantic editing and Benjamin into ontology based information extraction, leading to me being involved in the NEXUS project and generating many ideas on how to automatically disambiguate named entities in the very short association strings that I am dealing with. Next, Damian briefly shared an office with me, allowed me to shape the MOM and DeFuseNN projects with him, took me onto the SVL adventure with him, and later in the MADM group always had an open ear for me, tons of advice and allowed me to do my research by connecting Linked Data with Multimedia Analysis and Data Mining. Then, Joachim, not only let me tap into his vast knowledge about computer graphics, deep learning, machine learning in general and mad coding and optimization skills, but also deserves my gratitude for keeping me happy with never ending humour, keeping me focused, being one of the hardest, but always constructive critiques, keeping the bar high and developing a gazillion ideas with me. Last but not least, Rouven, one of my first interns, then HiWi for many years and part-time office mate, not only helped me to test out many crazy ideas and develop the many systems and interactive visualisations for this work, but also never gave

Office

up on overcoming even the weirdest browser, JavaScript and CSS challenges. All of you have become much more than just colleagues to me and I enjoyed every second of creativity with you guys in the room. Your feedback, ideas and support were invaluable to me and made this work what it is.

Students Next, I would like to thank the many bachelor and master students whom I had the honour to supervise in seminars, projects and theses. You gave me the chance to look left and right, and to widen my scope much further than I could've done without your support. Exceptional thanks here go to Tim for investigating how similarities between Wikipedia topics can be used to predict access statistics and Khamis for developing the Wikipedia Knowledge Test game with me.

Research Group Further, I would like to thank the many other members of the former MADM, KM and current SDS research groups, starting with

SemWeb the Semantic Web and Linked Data people. The works of Ludger, Michael, Ansgar, Heiko, Benjamin, Manuel, Björn, Sven, Malte, Gunnar and Leo originally inspired me to join the DFKI in Kaiserslautern. During my time, this area was strengthened by Tristan, Mike and later Markus, Sven and Christian. Thank you all for always taking the time for the many fruitful discussions that not only challenged me and deepened my knowledge, but also helped me to develop many of the ideas behind this thesis. Special thanks to Benjamin, Malte, Leo and Gunnar for igniting the Linked Data flame in me, and to Gunnar for letting me glimpse into his huge machine learning and SemWeb toolbox, encouraging my use of bash pipelines and Unix tools to juggle massive amounts of data, and last but not least for pulling me into the RDFLib project.

NEXUS Next, I'd like to thank the people from and around the NEXUS project, so Benjamin, Martin, Heinz, Stephan, Darko and Reuschi. You allowed me to generate and test my many ideas about ranking Linked Data facts in a very creative and fruitful environment. Special thanks to Martin and Heinz for embedding my ideas into the ALOE system and the never ending supply of "Heit schunn Gelacht?" and cat jump fail videos, and many thanks to Reuschi for later allowing me to reuse the Wikipedia indices for the article similarity baselines.

MADM I would also like to thank the former and current MADM group, with Tom, Jane, Armin, Adrian, Joost, Markus, Matthias, Kofi, Christian, Damian, Marco and later Sebastian, Federico, Benjamin, Patrick, Tushar and Philipp. You all made me feel at home, always had time for discussions and never were shy to give feedback and comments leading to an endless stream of ideas. Special thanks to Armin and TRB who initially took me on as CBR HiWi, then later to TRB for being my first mentor within the DFKI, supervising my master thesis and thereby paving the way for my PhD topic. Many thanks also to Adrian for his long term mentoring, showing me how to throw papers over the door saddle, being the sickest hacker (only challenged

by Gunnar, Mike and Joachim), gently nudging me into the right direction, tons of very fruitful discussions, unstucking me with for him obvious ideas and comments, and all the feedback and proofreading towards the end of this thesis. Tons of thanks also to Markus and later Christian and Damian for maintaining and gradually extending the compute infrastructure for everyone and all their feedback especially on the fusion part of this thesis. Last but not least, special thanks to all the people involved in the MOM and DeFuseNN projects for the many discussions, ideas and in general the nice atmosphere and fruitful environment.

Behind the scenes

Further, I'd like to thank the many other people from DFKI who supported me during this thesis, such as Brigitte, Jane, Kieni, Stefan, AL, AW, Nico, Stephan. While less visible, without your efforts behind the scenes nothing would work.

Mensa-Crew

Finally, I'd like to thank Jane, Adrian, Joost, Damian, Kofi, Alex, Markus, Fobs, Marco, Christian, Kieni, Benjamin, Patrick, Sebastian, Federico and Rouven, for being my DFKI (and friends) Mensa-Crew, keeping me well nourished and glutamat fuelled, but also for the random mensa talk and many crazy ideas arising from it.

SemWeb, OpenData and OpenSource Communities

Outside of the DFKI, I'd like to thank the whole Semantic Web community, but also the various connected communities and standardisation committees. In this work, I heavily benefited from the incredible efforts that you have put in openly available results, making today's (semantic) web possible and leading to standards such as RDF and SPARQL. In addition to Heiko, I'd also like to specifically thank Harald, Steffen, Andreas, Achim, Andreas, Steffen, Javier, Wouter and Ruben for the many nice discussions at conferences and elsewhere, their positivity, encouragement and very constructive feedback on my work. Many thanks also go out to the whole OpenData movement, especially to the DBpedia, Freebase and Wikidata teams, but in general also to anyone who puts data online for researchers to use (e.g., the EAT dataset). Without you sharing your work results, this thesis would not have been possible. Finally, I'd like to shout out a big thank you to the many people from the OpenSource Community. This work was only possible as I could stand on the shoulders of giants providing software and libraries such as Virtuoso, Jena, RDFLib, NumPy, SciPy, sklearn, deap and SCOOP.

Friends and Family

Last, but not least, I'd like to thank my friends and family for all their friendship, encouragement, support, for keeping me grounded and reminding me what actually matters in life. I'd especially like to thank Micha, Alex, Seb, Gecko, Nils, Annika, Benny, Michi, Jenny, Günther, Matthias, Sabine, Florian, Theresia, Barbara, Elisa, Uli, Dreiser, Daniel, Damian, Andi, Micha, Alex, Markus, Fobs, Gunnar, Tristan, Mike, Ashley and Teresa for their long time friendship, being there for me when I needed it, but also for all the geeking out in

Bonn and KL, as well as the many cool vacations, activities and celebrations spent together. I'd like to thank my flatmates Alex, Matthias, Barbara, Günther, Tristan and Teresa, as well as Mike and Ashley, for spending countless hours together, becoming some of my closest friends if not family in KL and turning a simple flat into something that I liked to come home to. Additional thanks go to Matthias, Mike and Teresa for all the advice and proofreading during and towards the end of this thesis. My deepest thanks however go to Teresa, for all her love, having my back especially in tough and stressful times, always listening to my worries, searching and finding solutions with me, explaining the world to me, nerding out and gaming with me, taking me for walks when I'm stuck, caring for me, getting mad at me when I work too much, but also giving me the freedom to be in the zone and accepting me as I am, and for allowing me to do the same for her. Finally, I'd like to humbly thank my parents Ilka and Helmut. My father, being the endless tinkerer, early on introducing me to computers starting with a C64 when I could barely read, but also building many things with me like tree-houses and go-karts, thereby teaching me to fix things, the value of debugging and not giving up. My mother, not only always patching me back up when I got hurt or fell down (which I did a lot), but also showing me the artistic sides of life, especially with painting and music, thereby teaching me to try again after failure and making me value artistic and elegant code. Without your endless support, love, all the time you took for me, positive influence and values, this thesis wouldn't even have been started.

— Thank you, Jörn

GRANTS

This work has been supported by the TU Kaiserslautern CS department's PhD Program, the BMBF projects NEXUS (Grant 01IW11001), MOM (Grant 01IW15002), DeFuseNN (Grant 01IW17002), the AHRP MOM grant for computation time on the Elwetritsch cluster and the NVIDIA AI Lab program (NVAIL).

PUBLICATIONS AS PART OF THIS THESIS

Parts of the research and material (including figures, tables and algorithms) in this thesis have already been published in:

J. Hees, T. Roth-Berghofer, and A. Dengel. "Linked Data Games: Simulating Human Association with Linked Data." In: *LWA 2010*. Kassel, Germany, 2010, pp. 255–260. URL: http://www.kde.cs.uni-kassel.de/conf/lwa10/papers/wm2.pdf.

J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, and A. Dengel. "BetterRelations: Using a Game to Rate Linked Data Triples." In: *KI 2011: Advances in Artificial Intelligence*. Berlin: Springer Berlin / Heidelberg, 2011, pp. 134–138. DOI: 10.1007/978-3-642-24455-1_12.

J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, and A. Dengel. "BetterRelations: Detailed Evaluation of a Game to Rate Linked Data Triples." In: *Proc. of the ISWC 2011 Ordering and Reasoning Workshop (OrdRing)*. Bonn, 2011. URL: http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/Workshops/OrdRing/paper_4_new.pdf.

J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, and A. Dengel. "BetterRelations: Collecting Association Strengths for Linked Data Triples with a Game." In: *Search Computing - Broadening Web Search*. Vol. 7538. Springer LNCS Berlin / Heidelberg, 2012, pp. 223–239. DOI: 10.1007/978-3-642-34213-4_15.

J. Hees, M. Khamis, R. Biedert, S. Abdennadher, and A. Dengel. "Collecting Links between Entities Ranked by Human Association Strengths." In: *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC*. Vol. 7882. Montpellier, France: Springer LNCS, 2013, pp. 517–531. DOI: 10.1007/978-3-642-38288-8_35.

J. Hees, B. Adrian, R. Biedert, T. Roth-Berghofer, and A. Dengel. "TSSort: Probabilistic Noise Resistant Sorting." In: *CoRR* abs/1606.0 (2016), pp. 1–10. arXiv: 1606.05289.

J. Hees, R. Bauer, J. Folz, D. Borth, and A. Dengel. "Edinburgh Associative Thesaurus as RDF and DBpedia Mapping." In: *The Semantic Web - ESWC 2016 Satellite Events*. Vol. 9989 LNCS. Heraklion, Crete, Greece: Springer LNCS, May 2016, pp. 17–20. DOI: 10.1007/978-3-319-47602-5_4.

J. Hees, R. Bauer, J. Folz, D. Borth, and A. Dengel. "An Evolutionary Algorithm to Learn SPARQL Queries for Source-Target-Pairs." In: *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016*. Vol. 10024. Bologna, Italy: Springer LNCS, Nov. 2016, pp. 337–352. DOI: 10.1007/978-3-319-49004-5_22. arXiv: 1607.07249.

J. Hees, R. Bauer, J. Folz, D. Borth, and A. Dengel. "Predicting Human Associations with Graph Patterns Learned from Linked Data." In: *ISWC 2017 Posters & Demonstrations and Industry Tracks*. Vol. 1963. Vienna, Austria: CEUR-WS Proceedings, 2017. URL: http://ceur-ws.org/Vol-1963/paper594.pdf.

CONTENTS

Publications as Part of this Thesis ix
List of Figures xiv
List of Tables xiv
Listings xv
Acronyms xv
URI Prefixes (CURIEs) xvii

I INTRODUCTION

1 INTRODUCTION 3
1.1 Motivation . . . 3
1.2 Research Question & Goals . . . 5
1.3 Terminology . . . 5
1.3.1 Human Associations, Stimulus, Response . . . 5
1.3.2 Semantic Web & Linked Data . . . 6
1.3.3 Semantic Associations . . . 8
1.3.4 Graph Patterns . . . 8
1.3.5 Graph Pattern Learning . . . 10
1.4 Overview . . . 10
2 STATE OF THE ART 13
2.1 Semantic Web & Linked Data . . . 13
2.1.1 Semantic Web . . . 13
2.1.2 Linked Data . . . 17
2.2 Psychology: Associations . . . 18
2.2.1 History and Experimental Collections . . . 19
2.2.2 Communication and Linguistics . . . 21
2.2.3 Semantic Networks and the Semantic Web . . . 23
3 RELATED WORK 25
3.1 Human Association RDF Datasets . . . 25
3.2 Learning from RDF Graphs . . . 28
3.2.1 Ranking Linked Data . . . 28
3.2.2 Embeddings & Vectorisation . . . 31
3.2.3 SPARQL Query Learning . . . 33

II DATASET GENERATION

4 GAMES WITH A PURPOSE (GWAP) 41
4.1 Related Games . . . 42
4.2 Better Relations . . . 44
4.2.1 The Game . . . 44
4.2.2 Behind the Scenes . . . 47
4.2.3 Evaluation . . . 48
4.2.4 Discussion . . . 51
4.2.5 Conclusion & Outlook . . . 53

4.3 Knowledge Test Game . . . 54
4.3.1 The Game . . . 54
4.3.2 Behind the Scenes . . . 55
4.3.3 Evaluation . . . 58
4.3.4 Discussion . . . 63
4.3.5 Conclusion & Outlook . . . 64
5 MAPPING OF EXISTING DATASETS 65
5.1 Edinburgh Associative Thesaurus (EAT) . . . 65
5.2 Association Vocabulary and RDF Version of EAT . . . 66
5.3 Mapping EAT to DBpedia . . . 68
5.3.1 Expected Quantities and Identified Challenges . . . 69
5.3.2 Semi-Automatic Mapping Approach . . . 70
5.3.3 Mapping Results and Mapping RDF Dataset . . . 73
5.4 Conclusion & Outlook . . . 74
6 DATASET GENERATION RESULTS 75
6.1 Comparison of Dataset Generation Approaches . . . 75
6.2 Semantic Association Ground Truth . . . 77
6.3 First Analysis of Semantic Associations in DBpedia . . . 77
6.4 Conclusion . . . 80

III PATTERN LEARNING FROM LINKED DATA

7 LEARNING APPROACH INTRODUCTION 83
7.1 Learning Approach Overview . . . 83
7.2 Design Goals . . . 83
7.3 Basics and Definitions . . . 85
7.3.1 Good Patterns . . . 87
7.3.2 Search Space Complexity . . . 89
7.3.3 Canonical Form of Isomorphic Graph Patterns . . . 93
8 PATTERN LEARNING ALGORITHM 97
8.1 Evolutionary Algorithm Overview . . . 97
8.2 Runs & Coverage . . . 98
8.3 Fitness & Evaluation . . . 99
8.4 Mating . . . 100
8.5 Mutation . . . 100
8.6 Initial Population . . . 103
8.7 Next Generation & Population Control . . . 104
8.8 Practical Considerations . . . 105
8.8.1 Batching . . . 105
8.8.2 Limits and Timeouts as a Proxy for Complexity 106
8.8.3 Fit To Live Filter . . . 106
8.8.4 Parallelisation . . . 106
8.8.5 Caching . . . 107
8.8.6 Non-Deterministic SPARQL Results . . . 107
8.8.7 Pattern Simplification . . . 107
8.9 Visualisation . . . 108
9 PATTERN BASED PREDICTION 113

9.1 Ground Truth Coverage . . . 113
9.2 Query Reduction by Clustering . . . 114
9.3 Target Candidate Generation . . . 116
9.4 Patterns as Feature Space . . . 116
9.5 Fusion Methods . . . 117
9.5.1 Basic Fusion . . . 118
9.5.2 Advanced Fusion . . . 119
9.6 Demo . . . 122
10 EVALUATION 127
10.1 Overview . . . 127
10.2 Description of Local Setup . . . 127
10.2.1 Local Linked Data Mirror & Loaded Datasets . . . 127
10.2.2 Cluster Setup . . . 129
10.3 Used Quality Metrics . . . 130
10.3.1 Recall@k . . . 130
10.3.2 MAP & MRR . . . 130
10.3.3 NDCG . . . 131
10.4 Simulation of Human Associations . . . 132
10.4.1 Dataset . . . 132
10.4.2 Baselines . . . 132
10.4.3 Basic Statistics & Achieved Training Coverage . . . 138
10.4.4 Notable Learned Graph Patterns . . . 139
10.4.5 Full System Evaluation (Prediction & Fusion) . . . 139
10.4.6 Analysis of Rank-Degree Correlations . . . 142
10.4.7 Prediction Quality Spread Evaluation . . . 143
10.5 Pattern Injection . . . 149
10.5.1 Path Length Evaluation . . . 150
10.5.2 Enumeration Based Evaluation . . . 151
10.6 Comparison to KORE Entity Relatedness Rankings . . . 152
11 OTHER APPLICATIONS 155
11.1 Entity Relatedness: DBpediaNYD . . . 155
11.2 Recommender Engine: TasteDive . . . 156

IV CONCLUSION
12 SUMMARY 161
13 FUTURE WORK 165

V APPENDIX
A VERIFIED MAPPINGS OF EAT TO WIKIPEDIA 169
B SEMANTIC ASSOCATION GROUND TRUTH DATASET 179
C EVALUATION OF ALL RDF2VEC MODELS 189

Bibliography 191
Index 211
Academic Curriculum Vitæ: Jörn Hees 215

LIST OF FIGURES

Figure 1.1 Graph Pattern Example 9
Figure 2.1 Semantic Web Layer Cake 14
Figure 2.2 Semiotic Triangle 21
Figure 2.3 Word vs. Thought Associations 22
Figure 2.4 Contextual Semantic Network 23
Figure 4.1 GUI: Round Choosing Phase 45
Figure 4.2 Ranking Comparisons 52
Figure 4.3 Knowledge Test Game Screen-shot 55
Figure 4.4 Knowledge Test Game Result Quality Ratings 61
Figure 5.1 EAT RDF Examples 67
Figure 5.2 Mapping EAT to DBpedia Example 68
Figure 5.3 DBpedia Mapping Verification Web Application 72
Figure 7.1 Graph Pattern Learner System Overview . . . 84
Figure 7.2 Conceptual Visualisation of a Good Pattern . . 87
Figure 8.1 Mutation Examples 101
Figure 8.2 Pattern Simplification 108
Figure 8.3 Visualisation of Graph Patterns 110
Figure 8.4 Visualisation of Graph Pattern Coverage 111
Figure 9.1 Query Reduction Precision Loss 116
Figure 9.2 Graph Pattern Learner Application Phase . . . 122
Figure 9.3 Demo: Stimulus Auto-Complete Input-Box . . 123
Figure 9.4 Demo: Prediction Results 123
Figure 9.5 Demo: Patterns for Prediction 125
Figure 9.6 Demo: Fully Expanded Pattern 125
Figure 10.1 Comparison of Fusion Variants and Baselines . 142
Figure 10.2 Node Degrees vs Ranks 143
Figure 10.3 Fusion Method Comparison: MRR 146
Figure 10.4 Fusion Method Comparison: NDCG 147
Figure 10.5 Fusion Method Comparison: Recall@10 148

LIST OF TABLES

Table 2.1 Example SPARQL SELECT Results 16
Table 3.1 Comparison of Prior Datasets 27
Table 4.1 Online Survey Results 49
Table 4.2 Example Output Comparison 50
Table 4.3 Results of an Online Survey 60
Table 4.4 Association Example 62

Table 4.5 Top Played Topics' NDCGs 62
Table 5.1 Example of EAT Associations 66
Table 5.2 Verified Semantic Association Mappings Excerpt 73
Table 6.1 Comparison of Dataset Generation Approaches 75
Table 6.2 Semantic Association Ground Truth Excerpt . 76
Table 6.3 Most Frequent Response Nodes 77
Table 6.4 Top-20 Degrees in Core vs. Extended Dataset . 78
Table 10.1 Comparison of Baselines 137
Table 10.2 Comparison of Fusion Variants 141
Table 10.3 Node Degree vs Rank Correlations 144
Table 10.4 Injection Path Length Evaluation 150
Table 10.5 Injection Pattern Enumeration Evaluation . . . 151
Table 10.6 Comparison with KORE 153
Table 11.1 Other Applications 155
Table A.1 Verified Mappings of EAT to Wikipedia 169
Table B.1 Semantic Assocation Ground Truth Dataset . . 179
Table C.1 Comparison of all RDF2Vec Baselines 189

LISTINGS

Listing 7.1 SPARQL BGP Canonicalisation 95

ACRONYMS

AGI	Artificial General Intelligence
AI	Artificial Intelligence
BGP	SPARQL Basic Graph Pattern[1] [75, 156]
BNode	Blank Node[2] [104, 182]
CURIE	Compact URI [25]
EAT	Edinburgh Associative Thesaurus [101]
GWAP	Game With A Purpose [3, 5]

1 https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#sparqlBasicGraphPatterns
2 https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-blank-nodes

HTTP	HyperText Transport Protocol [56–59]
IQR	Inter-Quartile Range
IRI	Internationalised Resource Identifier [47]
LOD	Linked Open Data
LOD Cloud	Linking Open Data Cloud [1]
MAP	Mean Average Precision
MRR	Mean Reciprocal Rank
MSE	Mean Squared Error
NDCG	Normalised Discounted Cumulative Gain
NER	Named Entity Recognition
POS	Part Of Speech
RDF	Resource Description Framework [104, 182]
SPARQL	SPARQL Protocol And RDF Query Language [75, 156]
URI	Uniform Resource Identifier [22, 23]
USFA	University of South Florida free association, rhyme, and word fragment norms [131, 132]
W3C	World Wide Web Consortium[3]
XML	Extensible Markup Language (here in the sense of RDF/XML [17, 154])
WWW	World Wide Web [20]

3 http://www.w3c.org

URI PREFIXES (CURIES)

assoc:	https://w3id.org/associations/vocab#
dbo:	http://dbpedia.org/ontology/
dbpam:	https://w3id.org/associations/mapping_eat_dbpedia#
dbprop:	http://dbpedia.org/property/
dbr:	http://dbpedia.org/resource/
dcterms:	http://purl.org/dc/terms/
eat:	http://www.eat.rl.ac.uk/#
foaf:	http://xmlns.com/foaf/0.1/
gold:	http://purl.org/linguistics/gold/
ktg:	http://knowledgetestgame.org/resource/
owl:	http://www.w3.org/2002/07/owl#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
schema:	http://schema.org/
skos:	http://www.w3.org/2004/02/skos/core#
umbel:	http://umbel.org/umbel/rc/
wiki:	https://en.wikipedia.org/wiki/

Part I

INTRODUCTION

1 INTRODUCTION

1.1 MOTIVATION

In the past decade, many areas of Artificial Intelligence (AI), such as computer vision and speech recognition, have shown astonishing progress. Especially, the uptake of Deep Learning [110] has revolutionised machine learning. Modern hardware and training methods allow the training of models with previously unthinkable complexity and amounts of data. Additionally, end-to-end learning approaches often allow such deep learning systems to out-perform traditional machine learning pipelines, in which experts had to manually define the useful features up-front.

AI progress

At the same time, the Semantic Web [24] and its Linked Data [26] movement have made many large, machine accessible and interlinked Resource Description Framework [104, 182] (RDF) datasets available, prominently depicted as the Linking Open Data Cloud [1] (LOD Cloud). The semantic datasets are available as subject-predicate-object triples in the form of RDF as single documents, dataset dumps or directly query-able via publicly accessible SPARQL endpoints and form the currently largest openly available representation of machine accessible knowledge. Due to the encyclopaedic nature of Wikipedia[1], its machine accessible pendant DBpedia[2] [27] has become one of the most interlinked and central datasets of the LOD Cloud.

Semantic Web & Linked Data

However, despite all advances in AI and the availability of large, interlinked knowledge bases, the path to Artificial General Intelligence (AGI)[3] is still a long one. Current intelligent systems are narrow (weak, niche) AIs: systems that can solve a very specific problem (e.g., recognising objects in images, playing certain games), sometimes even with super-human performance (e.g., Chess: Deep Blue [35], Go: AlphaGo [158]), but cannot simply be applied to problems outside of their domain. Also, the vast majority of these systems do not make use of available knowledge in form of Linked Data, as it is still a very challenging task to incorporate such knowledge: At the moment knowledge experts have to manually select knowledge sources and extract useful "knowledge features" up-front.

Narrow AI

Use of knowledge in AI is challenging

A prominent example of an AI system that used large amounts of knowledge is Watson [55], a question answering system designed to beat human champions in the American TV quiz show Jeopardy. Al-

IBM Watson

1 http://www.wikipedia.org

2 http://dbpedia.org

3 We consider the potential benefits of AGI to outweigh its risks, but acknowledge the need for more research to align human and AGI interests (e. g., in the sense of [163]).

though Watson's achievements are very impressive because Jeopardy is an open domain quiz show, the designed system is not an AGI, but rather an engineering masterpiece: Based on a large amount of training questions and answers, Watson was developed in a performance driven manner. Repeatedly analysing in which tasks the system performed sub-optimally, humans decided which (new) knowledge sources to incorporate (e. g., which web-crawls, which text corpora or which parts of Linked Data) and how to make use of them for hypothesis generation by developing a multitude of heuristics. The final system then used a sophisticated late fusion approach to combine the hypotheses generated by all humanly developed algorithms back together. While similar approaches are used in modern search engines such as Google (based on the Google Knowledge Graph [160]) and Bing (based on Satori [143]), many of the underlying humanly developed algorithms are designed to heuristically solve very specific problems. While solving their problems well, and thereby improving the precision and recall of the overall system, these algorithms typically do not resemble human-like thinking at all. They are likely to fail when applied to problems outside of their intended scope, leading to sub-optimal results in scenarios for which there is no specially crafted heuristic. Hence, in this work, we would like to advance towards an alternative approach; an algorithm that is more general and tries to simulate human-like thinking per se.

Search engines

Looking at human thought from a psychological point of view, associations are believed to be one of the fundamental parts of human thinking processes [101]. They are the mental connections between thoughts and concepts leading from a stimulus to a response (e.g., "cat - dog", "house - roof"). In psychology, they are sometimes represented as Semantic Networks [40], [13, p. 120f], which can be seen as simplistic forms (and motivation) of the knowledge graphs found in the Semantic Web.

Human thought & associations

Hence, in this work we tackle the challenge to simulate human associations. Analogously to humans, who use their own knowledge to associate "dog" with "cat", we investigate if machines can produce similar results in their world. As Linked Data can be seen as the memory component of AI systems, in this work we will focus on simulating such associations with the help of Linked Data: Given a stimulus semantic entity (e.g., `dbr:Dog`), we are searching for a system that is able to use the machine accessible knowledge to automatically extract features (in form of graph patterns) to predict a response semantic entity (e.g., `dbr:Cat`).

Simulating human associations with Linked Data

While the focus of this work lies on fundamental AI research, one of the direct applications of simulating human associations with Linked Data is human-like ranking of (intermediate) result sets. We see human association strengths as a good alternative to other, often used relevance heuristics. Especially in exploratory scenarios, such as man-

ual browsing, spreading activation or other expansion based (search) algorithms, which currently often suffer from the high node degrees in Linked Data, human associations could help to reduce the search space. In general, human associations and strengths between semantic entities could pave the way to simulate human-like thinking.

1.2 RESEARCH QUESTION & GOALS

The main research question of this thesis is:

Is it possible to simulate human association with Linked Data?

This question can be decomposed into two sub-questions and corresponding goals:

1. **Question**: How can we collect a high-quality dataset of semantic associations?

   Goal: Generate a high-quality dataset of semantic associations, which means that it should consist of associations that are undisputed in the general population. At the same time, the dataset should be as large as possible and made accessible to other researchers in a machine-usable form (i.e., RDF).

2. **Question**: How can we exploit this newly collected data and existing Linked Data to simulate human associations?

   Goal: Simulate human associations with Linked Data by using the previously generated dataset to discover regularities of associations in available Linked Data. Investigate whether this can be done with an end-to-end machine learning approach. Due to the volume of available Linked Data, the approach needs to be scalable. Further, it is desirable that the results are explainable[4].

1.3 TERMINOLOGY

After the previous motivation and goals, in this section we will briefly introduce the terminology used throughout the remainder of this thesis. The focus here lies on general understanding of the reader. A full state of the art introducing the prerequisites for this work can be found in Chapter 2. Formal definitions will follow in the relevant chapters.

1.3.1 *Human Associations, Stimulus, Response*

As mentioned in the introduction, *associations* are mental connections *association*

4 While explain-ability from a scientific point of view is desirable, but not mandatory in machine learning, the recent EU General Data Protection Regulation (GDPR) [50] effectively will grant humans a "right to explanation" [68] when affected by algorithmic decision making.

stimulus
response

between thoughts and concepts leading from a *stimulus* to a *response* (e.g., "cat - dog", or "house - roof").

association strength

Associations are of different *association strength*, for example the association "dog - cat" is much stronger than "dog - leash". Here, we measure strength by the percentage of people who agree on an association, or more precisely who agree on a response given a stimulus. Association strengths can vary greatly depending on the current *context*. In the scope of this work however, we do not consider contextual effects, but focus on *strong associations* existing independent of context.

context
strong associations

Furthermore, while associations can have various forms, in this work we focus on pairwise *word associations*: an association measurable in free-text form and representable as textual *stimulus-response pair*.

word associations
stimulus-response pair

Further discussion and psychological considerations on associations can be found in Section 2.2.

Associations and Other Relations

There exist other types of relations between concepts, such as similarity, hyper-/hyponyms, synonyms and antonyms. While many works do not distinguish them, as they are often overlapping, in this work we will differentiate between them and especially focus on associations. To illustrate the difference, we want to point out that associations exist between words (and entities) that are not similar, such as "horse - saddle" or "baby - crying". Also, similarities (even as strong ones as synonyms) exist that are not strongly associated, such as "dog - terrier" (or the synonymous "dog - canine").

associations vs. similarity

1.3.2 *Semantic Web & Linked Data*

Semantic Web
Linked Data

The *Semantic Web* [24] as introduced in 2001 and its *Linked Data* [21] movement[5] refer to the concept of putting data on the web in a machine readable, non-proprietary, standardised way. Published by the World Wide Web Consortium (W3C), this standardised way should be the Resource Description Framework [104, 182] (RDF). The linking aspect focuses on the reuse of existing URIs that have been published by others. Data published with a permissive licence is often also called Linked Open Data (LOD) to emphasise its openness.

RDF

Linked Open Data

1.3.2.1 *Knowledge Base, Knowledge Graph, SPARQL Endpoint*

Vast amounts of Linked (Open) Data already exist and can be accessed online thanks to the efforts of the Linking Open Data Commu-

5 http://www.w3.org/wiki/LinkedData

nity[6]. Many of these datasets are famously depicted as the Linking Open Data Cloud [1] (LOD Cloud). Due to the linking of information, the resulting structures can be considered as a graph, often also called the *knowledge graph*. In this work, we also refer to this graph or parts of it as *knowledge base*.

LOD Cloud

knowledge graph

knowledge base

While it would be in spirit of the Semantic Web and Linked Open Data (LOD) ideas to directly consume online knowledge from web hosting of their publishers, this is not practical for this work: For graph pattern learning, we need to ask many queries that sometimes are very computationally challenging and expensive. Doing this on publicly available servers would not be considered "fair-use". Hence, for this work we use a common solution to manually set up a local *Linked Data mirror* (cache) in form of a *SPARQL Protocol And RDF Query Language [75, 156] (SPARQL) endpoint*, with a subset of all available knowledge. The selection of the subset (sub-graph) is a trade-off between completeness, availability and computational feasibility. Details on the exact set-up can be found in Section 10.2.

Linked Data mirror

SPARQL endpoint

1.3.2.2 *Semantic Entity, Node, URI and Labels*

Within this work, we use Linked Data terminology to refer to parts of knowledge graphs. *Nodes* of the graph correspond to *semantic entities* and can be uniquely identified by their URI. As an example the node (or the semantic entity) "Dog" has the Uniform Resource Identifier [22, 23] (URI):

node

semantic entity

URI

`http://dbpedia.org/resource/Dog`

We typically denote this by the Compact URI [25] (CURIE):

CURIE

`dbr:Dog`

A list of all CURIE prefixes used in this work can be found on page xvii.

In this work, the terms "(semantic) entity", "node" and "URI" may be used interchangeably.

In the above example, the textual representation "Dog" is also called a *label* for the node `dbr:Dog`.

label

1.3.2.3 *Triple (Subject, Predicate, Object), Statement, Fact, Relation, Edge*

For us, a knowledge base consists of a set of *facts*[7]. Each fact can be represented as (subject, predicate, object)-*triple* (or (s, p, o)-triple). An example of such a fact is (`dbr:Dog`, `rdf:type`, `dbo:Mammal`) with the textual surface form "dog is a mammal".

fact

triple

In this work, the terms "triple", "statement" and "fact" may be used interchangeably. The terms "*relation*" and "*edge*" may as well be

relation, edge

6 https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

7 In the scope of this thesis, we do not actively distinguish between A-Box and T-Box.

used to refer to a triple, or depending on context, may focus on the predicate.

This notion of triples is extended below (Section 1.3.4) to allow variables, thereby forming a query.

1.3.2.4 *RDF Terms: URI (IRI), BNode, Literal*

IRI, BNode, Literal

RDF triples consist of Internationalised Resource Identifiers [47] (IRIs), Blank Nodes [104, 182] (BNodes) and *Literals*. IRIs can occur in subject, predicate and object position, BNodes in subject and object position, and Literals in object position [104, 182].

RDF term

To refer to any one of IRI, BNode or Literal, we will use the name *RDF term*[8] in accordance with [104, 182]. To ease readability, we will refer to IRI with the more common term URI whenever the distinction is negligible.

Further details on RDF can be found in Section 2.1.1.1.

1.3.3 *Semantic Associations*

surface form
symbolic form

The "same" association can have different textual *surface forms* (also called *symbolic forms*). For example, "New York - America" and "NYC - USA" can mean the same unique association. We are interested in such unique association that exists between the two semantic entities "New York" (with the synonyms "NYC", "the big apple", ...) and "United States of America" (with synonyms such as "America" and "USA"). In context of the Semantic Web and Linked Data, we can use URIs to refer to these entities (e.g., `dbr:New_York_City` and `dbr:United_States`). In this work, we call an association between a pair of such semantic entities a *semantic association*. The definition excludes associations that are purely within the textual surface form of one semantic entity, such as "New - York" or "Michael - Jackson".

semantic association

1.3.4 *Graph Patterns*

graph pattern gp
variable

Throughout this work, a *graph pattern* gp is a set of triples that can be composed of RDF terms as well as SPARQL *variables*. Graph patterns can be seen as templates for sub-graphs of a given knowledge base. The following is an example for a graph pattern with three triples and the three variables (`?v1`, `?source` and `?target`):

```
?source rdf:type dbo:Country .
?source ?v1 ?target .
?target rdf:type dbo:Capital .
```

8 https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-triples

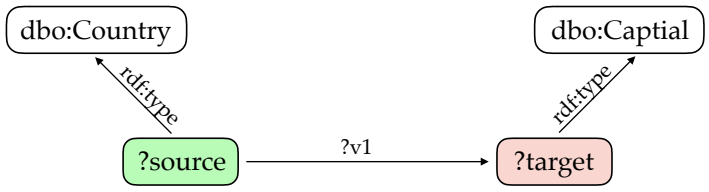


Figure 1.1: Graph Pattern Example

As can be seen, we typically use a line based sub-set of the SPARQL TriplesBlock[9] syntax to denote graph patterns. A graphical graph representation of the pattern can be found in Figure 1.1.

1.3.4.1 *Query, SPARQL, Variables, BGP*

A graph pattern is closely related to a SPARQL query. Being a set of triples with variables, a graph pattern formally forms a SPARQL Basic Graph Pattern [75, 156] (BGP). *SPARQL Basic Graph Pattern*

A SPARQL query can be executed against a SPARQL endpoint representing a knowledge base. During execution, the endpoint will try to unify (bind, instantiate) variables to existing RDF terms from the knowledge base. If an instantiation of all variables can be found, the instantiated set of triples (unifying all variables to their existing RDF terms) forms a sub-graph of the knowledge base. In such cases, we say the graph pattern (or query) can be *fulfilled* (by this sub-graph). *fulfilled*

SPARQL allows us to retrieve the bindings of the variables. In a SELECT query, the user can specify the desired output sub-set of variables and further manually bind variables with input.

Throughout this thesis, *variables* are marked with a preceding "?" (e.g., `?source`, `?target`, `?v`). *variable*

1.3.4.2 *Source, Target*

Also, since we're aiming at learning associations (stimulus-response pairs), our patterns will contain at least a `?source` or a `?target` variable[10]. A pattern containing both is called *complete pattern*, one containing only `?source` or `?target` is called an *incomplete pattern*. *?source, ?target* *complete pattern* *incomplete pattern*

Throughout this work, when talking about (the special case of) associations, we will also refer to source and target as stimulus and response.

Continuing our previous example, the following SPARQL *SELECT query* is given, which corresponds to the graph pattern in Figure 1.1, binds the `?source` variable to `dbr:Germany` and selects the `?target` variable: *SELECT query*

```
SELECT DISTINCT ?target WHERE {
  VALUES (?source) { (dbr:Germany) }
```

9 https://www.w3.org/TR/sparql11-query/#rTriplesBlock

10 Typically, after successful training they contain both, a `?source` and a `?target`.

```
    ?source rdf:type dbo:Country .
    ?source ?v1 ?target .
    ?target rdf:type dbo:Capital .
}
```

Executing the query against our local endpoint (cf. Section 10.2) returns the following values for the `?target` variable: `dbr:Berlin`, `dbr:Mainz`, `dbr:Munich`, `dbr:Stuttgart`, `dbr:Kiel`, and `dbr:Schwerin`.

1.3.5 *Graph Pattern Learning*

Graph pattern learning is the identification of a "good" set of graph patterns from a knowledge base for a given training list of exemplary source-target pairs (see Chapter 7 for more details).

1.3.5.1 *Machine Learning, Training Data, Ground Truth*

training data
ground truth GT
model

A machine learning algorithm uses *training data* (also called *ground truth* $\mathcal{GT}$) as input for its training phase to generate (learn) a so called *model*. The goal of machine learning is to generate a model that, during application phase, can replicate the learned behaviour. Depending on the field of application and type of model, this replication is often also called prediction.

1.3.5.2 *Source-Target Pairs*

source-target pairs

The training data for our graph pattern learner consists of a knowledge base and an exemplary list of *source-target pairs*. In our case, the latter is typically a list of semantic associations, which is generated by mapping stimulus-response pairs to semantic entities.

1.4 OVERVIEW

This thesis is structured as follows:

After the previous motivation (Section 1.1), the research questions and goals of this thesis (Section 1.2), and a short introduction of the used terminology (Section 1.3), in the remainder of Part I, we will describe the state of the art (Chapter 2) and related work (Chapter 3).

The rest of this thesis is split into two parts, one for each of the main goals elaborated in Section 1.2: dataset generation (Part II) and pattern learning from Linked Data (Part III).

In the dataset generation part (Part II), we will introduce several methods to collect a high-quality dataset of semantic associations. First, we present two Games With A Purpose (GWAPs) in Chapter 4, namely BetterRelations (Section 4.2) and the Knowledge Test Game (Section 4.3). While games allow us to use fun as a motivator for high-quality data collection, they still rely on a lot of human work. Hence, in Chapter 5 we present a semi-automatic mapping approach

to transform existing psychological human association datasets into RDF and map them to DBpedia entities. We conclude the dataset generation part with an analysis of the results in Chapter 6.

In the pattern learning part (Part III), we present our evolutionary end-to-end semantic graph pattern learning algorithm. After an introduction in Chapter 7, the core of the algorithm is presented in Chapter 8. Chapter 9 then describes how the training results of our algorithm can be used for pattern based prediction (e.g., of human associations), before we evaluate our algorithm in Chapter 10. While the original motivation of this thesis is to take a small step towards AGI by simulating human associations with Linked Data, we end Part III with other applications of our algorithm in Chapter 11.

The thesis is concluded in Part IV with a summary and future work.

2 STATE OF THE ART

After the introduction in the previous chapter, this chapter is dedicated to the foundations this work is built upon. As this thesis is situated in the intersection of two main research areas, this chapter is as well split into two main sections: In Section 2.1 we will introduce the necessary concepts from the computer science research field "Semantic Web & Linked Data", before focusing on the relevant results from psychological research around "associations" in Section 2.2.

2.1 SEMANTIC WEB & LINKED DATA

2.1.1 *Semantic Web*

Introduced to the general public in 2001[1], the Semantic Web [24] drew a futuristic picture by asking the question: What if the World Wide Web [20] (WWW) could be understood not only by humans, but also by machines? Envisioned was a world of smart agents that can support humans by being able to communicate freely, in a decentralised, standardised, extensible way, allowing agents to "understand each other" without prior human intervention.

To realise such communication, the traditional way would have been to rely on the existing communication means tailored towards humans and then apply sophisticated AI techniques (e.g., Natural Language Processing) on the receiving end to try and simulate human understanding. Such approaches however (especially at the time), suffered from too low accuracies and overall applicability. Hence, the idea of the Semantic Web is a different one. It tries to circumvent the communication problems between machine agents altogether: Rather than rendering information for humans only, the idea is to also provide information in a universal, machine accessible and extensible way that makes the extraction process either unnecessary or at least very uncomplicated and unambiguous.

Semantic Web main idea

To provide information in such a machine accessible way, the Semantic Web Community uses and standardises a series of core technologies. Their hierarchical relations are famously summarised in the (evolved) *Semantic Web Layer Cake*, as can be seen in Figure 2.1.

The two, for this work most relevant of these core technologies, RDF and SPARQL, will briefly be introduced in the following.

1 Many Semantic Web research activities pre-date this famous article. Tim Berners-Lee's earliest mention of the Semantic Web ideas can be found in 1994 [19] already.

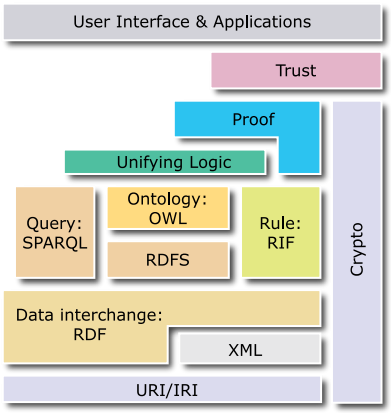


Figure 2.1: The Semantic Web Layer Cake [176]

2.1.1.1 *RDF*

The Resource Description Framework [104, 182] (RDF) is a universal data interchange format. Its central idea is that information is exchanged in form of simple subject-predicate-object *statements*, also known as (subject, predicate, object)-*triples*. Such RDF triples thereby resemble simplistic sentences in human communication. Unlike human communication however, they consist of Uniform Resource Identifiers [22, 23] (URIs) (or nowadays more precisely Internationalised Resource Identifiers [47] (IRIs)), Blank Nodes [104, 182] (BNodes) and *Literals*. URIs can occur in any, BNodes in subject and object, and Literals only in object position [104, 182]. We can formally define an RDF triple (s, p, o) as a statement of the infinite universe of all statements:

RDF statements

$$(s, p, o) \in (\text{IRIs} \cup \text{BNodes}) \times \text{IRIs} \times (\text{IRIs} \cup \text{BNodes} \cup \text{Literals})$$

While RDF was initially solely based on the Extensible Markup Language (XML) as its serialisation format, large data dumps are nowadays typically serialised in simpler formats such as N-Triples [155] or Turtle [142]. In this work, we will mainly write triples in Turtle syntax, as it is more readable than RDF+XML, more concise than N-Triples by using Compact URIs [25] (CURIEs) instead of URIs and as it is very close to the syntax used in SPARQL Basic Graph Patterns [75, 156] (BGPs).

As an example, the following triple in N-Triples syntax[2] expresses that Kaiserslautern is located in the country Germany:

```
<http://dbpedia.org/resource/Kaiserslautern>
 <http://dbpedia.org/ontology/country>
  <http://dbpedia.org/resource/Germany> .
```

The same triple in Turtle syntax using CURIEs looks like this:

```
dbr:Kaiserslautern dbo:country dbr:Germany .
```

2 Newlines added for readability, in N-Triples the triple would occupy one line only.

As mentioned before, triples can contain other components than URIs, such as BNodes or *Literals*.

BNodes

BNodes were initially often used to group multiple statements together without assigning an identifier to the group. Nowadays, large datasets however try to avoid BNodes for such simplistic use-cases and replace them with generated URIs in order to reduce the need to solve RDF (sub-)graph isomorphism problems during parsing (and re-parsing) time [116].

Literals

Literals contain the actual data that is interlinked by the triples. As an example, the following triple states that Kaiserslautern has a total area of 139.72 km^2:

```
dbr:Kaiserslautern dbo:PopulatedPlace/areaTotal 139.72 .
```

Another example shows how URIs are connected to the human linguistic world by stating: The entity identified by URI `http://dbpedia.org/resource/Kaiserslautern` has a German label "Kaiserslautern":

```
dbr:Kaiserslautern rdfs:label "Kaiserslautern"@de .
```

To state that Kaiserslautern is a town, we could use the following triple:

```
dbr:Kaiserslautern rdf:type dbo:Town .
```

Vocabularies

To define the meaning of such triples and allow for increasingly powerful reasoning, the Semantic Web community standardises and provides RDFS [71, 72] and OWL [43, 175] (see the OWL Primer [153] for a deeper introduction). RDFS and OWL can be used to describe ontologies, also called vocabularies, and try to close the gap between RDF triples and description logics to formally give those triples a meaning. RDF, RDFS and OWL are used to describe themselves (semantically grounded) and available as vocabularies with the prefixes `rdf`, `rdfs` and `owl`. The previous two examples already showed the use of `rdfs:label` and `rdf:type`. The following shows a triple that defines `dbo:Town` as a class:

```
dbo:Town rdf:type owl:Class .
```

and a triple that makes it a sub-class of `dbo:Settlement`:

```
dbo:Town rdfs:subClassOf dbo:Settlement .
```

The latter, for example allows for simple `rdfs:subClassOf` reasoning and would allow us to infer that Kaiserslautern is also a settlement in form of the triple:

```
dbr:Kaiserslautern rdf:type dbo:Settlement .
```

In practice, such reasoning is often performed during triple creation time (materialised) in order to reduce the complexity during query time.

This concludes the introduction of the most important aspects of RDF for this work. For a deeper introduction into RDF please refer to the excellent RDF Primer [118, 145].

Table 2.1: Example SPARQL SELECT Results

?town
dbr:Dornburg-Camburg
dbr:Eisenhüttenstadt
dbr:Göttingen
dbr:Kaiserslautern
...

2.1.1.2 *SPARQL*

SPARQL

With RDF triples available and accessible as described before, we can make use of them with the SPARQL Protocol And RDF Query Language [75, 156] (SPARQL). An overview over the SPARQL technology stack can be found in [177].

SPARQL allows us to formulate SQL like queries against a *SPARQL endpoint*. During query execution, the endpoint internally matches the query body against the RDF triples it has access to, forms a result set containing all matched sub-graphs and returns results following potential groupings and projections analogously to an SQL database.

SELECT query

For example, the following *SELECT query* can be used to get a list of all towns in Germany from the DBpedia SPARQL endpoint (`http://dbpedia.org/sparql`):

```
SELECT DISTINCT ?town WHERE {
  ?town a dbo:Town .
  ?town dbo:country dbr:Germany .
}
```

As `rdf:type` is a very common predicate, the above query makes use of SPARQL's shorthand `a` for it. An excerpt of the results is shown in Table 2.1.

SPARQL *variables* are prefixed with a `?` (e.g. `?town` above). The body of the above SELECT query only consists of triples and thereby forms a SPARQL Basic Graph Pattern [75, 156] (BGP).

COUNT query

For this work two more query forms are important: COUNT and ASK queries. A *COUNT query* (more precisely a SELECT query with COUNT aggregation) allows us to simply count the amount of results. For example, the query

```
SELECT (COUNT(DISTINCT ?town) as ?c) WHERE {
  ?town a dbo:Town .
  ?town dbo:country dbr:Germany .
}
```

returns a count of 1980. A query for cities only returns 58 results:

```
SELECT (COUNT(DISTINCT ?city) as ?c) WHERE {
  ?city a dbo:City .
  ?city dbo:country dbr:Germany .
}
```

ASK query

An *ASK query* allows us to ask simple boolean questions. They evaluate to true if any result set exists and false if none exists. For example, the following will negate the question if there are any subjects that are at the same time a city and a town in Germany:

```
ASK {
  ?s a dbo:Town .
  ?s a dbo:City .
  ?s dbo:country dbr:Germany .
}
```

As mentioned above, SPARQL endpoints match queries against the triples they have access to. Which triples an endpoint has access to is mostly a decision of the endpoint's administrator and can typically mean a mixture of three things:

Offline cache

- The endpoint operates in offline, cache-only mode. It only evaluates queries against triples already loaded in its backend (typically a so-called Triple- or Quad-Store). Triples are manually loaded into the backend, typically by locating existing relevant RDF dumps and bulk loading them, or via SPARQL Update Queries [65].

Online cache

- The endpoint operates on live information from the (Semantic) Web by de-referencing encountered URIs retrieving further RDF triples and adding them to its backend as cache. This mode is clearly closer to the original vision of the Semantic Web, but in practice has a tendency to cause a prohibitive amount of network load, very long query evaluation times and pollution of the endpoint to a degree that makes it unusable.

Query federation

- The endpoint co-operates with other SPARQL endpoints via SPARQL Query Federation [141], allowing users to explicitly direct a portion of a query to another SPARQL endpoint.

In this work, we exclusively rely on the first of the three options by hosting an own local RDF cache (as detailed in Section 10.2) in order not to disrupt the operation of community hosted services. Potential extensions to use newer techniques such as Triple Pattern Fragments [173] or Header Dictionary Triples (HDT) [54] are part of our future work as mentioned in Chapter 13.

2.1.2 *Linked Data*

Despite many successes in the areas of description logics, correct modelling of knowledge, reasoning and automated proofs, by 2006 the Semantic Web vision remained largely unrealised [157]. Only very limited amounts of RDF were available online. Hence, in an effort to refocus the activities on the Web aspects, Tim Berners-Lee's formulated his "Linked Data Design Issues" [21] in 2006 and coined the

Linked Data

term *Linked Data*. In four simple rules, he summarised the most important aspects of data to be Semantic Web conform: using URIs to name things, using HyperText Transport Protocol [56–59] (HTTP) URIs so people and machine can look them up, using standards such as RDF and SPARQL and last but not least, re-using URIs of others to allow discovering more things [21].

5-star rating for Linked Data

Later, Berners-Lee extended his note by an even simpler five star rating scheme, which emphasises that the availability of data is a pre-condition for correct modelling: The first star (1/5) is gained by putting data on the web (whatever format), the second (2/5) for using a machine-readable format (e.g., Excel). Putting CSV data online already gives you 3/5 stars, even before any Semantic Web technologies such as RDF or SPARQL are used (4/5) and before the data is linked with other data (5/5) [21]. The correct modelling of complicated relations became a secondary goal to putting data online at all. While this often meant that the provided data was too shallow, uncertain and noisy [95, 97] to be used by traditional (reasoning based) methods, it nowadays opens the door for machine learning and data mining approaches [148], such as the related works presented in Section 3.2 and our own approach in Part III.

LOD Cloud

After its formation, the Linked Open Data community revolutionised the Semantic Web landscape by quickly generating and publishing many large and interlinked RDF datasets with open access licenses [26]. The interlinking of many of these datasets are prominently depicted by the Linking Open Data Cloud [1] (LOD Cloud).

Extracted from Wikipedia, DBpedia [27] is one of the most central of these datasets. Due to its encyclopaedic nature, it provides information about entities from a large variety of domains, provides URIs for these entities and became a natural interlinking target for many other domain specific datasets in the LOD Cloud.

Aside from DBpedia many other knowledge bases and projects exist that collect commonsense knowledge [107], such as Wikidata [174], Freebase [30] (and later Google Knowledge Graph [160]), Satori [143], YAGO [166], Wordnet [52, 125], BabelNet [48, 130], Cyc [70], Open Mind Common Sense [159] (and later ConceptNet [115, 164]) ThoughtTreasure [129], Mindpixel [122] and NELL [36, 128]. Where publicly available as RDF dumps, the above datasets are included in our local SPARQL endpoint as detailed in Section 10.2. Their interlinking and centrality in the LOD cloud is however (still) weak in comparison to DBpedia, which is why in this work we mainly focus on and link against DBpedia entities.

2.2 PSYCHOLOGY: ASSOCIATIONS

In contrast to the relatively young field of Semantic Web research, the history of associations goes back to ancient times. Before diving into

the history of associations however, we will briefly mention our own definition of associations:

In this work, *associations* (also sometimes called associations of ideas or mental associations) are mental connections between thoughts allowing us to mentally navigate from one thought (the *stimulus*) to another (the *response*). *association*

Associations are a property of the so called semantic memory [13, pp. 113–121] and seen as one of the basic components of human thinking in modern cognitive science:

> It has always been and remains to be a general belief that associative processes are a basic component of thought and cognitive processes in general. — Kiss et al. [101]

They are also seen as especially important for language understanding, context forming, reasoning and learning [13, 66].

In the following, we will give a brief summary of the historical development of associations and their first collections (Section 2.2.1), their connections to communication and linguistics (Section 2.2.2), and their connections to Semantic Networks and the Semantic Web (Section 2.2.3).

2.2.1 *History and Experimental Collections*

Over time, associations sparked the interest of many philosophers and psychologists trying to explain human thinking. As the history of associations can easily fill whole books [179], we will confine ourselves to a very brief summary of the stages most relevant to this work.

The first treatises about associations are attributed back to Aristotle:

> When, therefore, we accomplish an act of Reminiscence, we pass through a certain series of precursive movements, until we arrive at a movement, on which the one we are in quest of is habitually consequent. Hence too it is, that we hunt through the mental train, excogiating [what we seek] from [its Concomitant in] the present or some other [time], and from its similar or contrary or coadjacent. Through this process Reminiscence is effected. For the movements [which and by which, we recollect,] are, in these cases, sometimes the same, sometimes at the same time, sometimes parts of the same whole; so that [having, from one or other of these, obtained a commencement,] the subsequent movement is already more than half accomplished.
>
> — Aristotle [7] as translated by Hamilton [74]

In the 18th century, the interest in associations led to the so called associationist theory and Associationist School, a group of thinkers who tried to ground explanations for human thinking on a few laws of association such as the "law of contiguity" and the "law of similarity". From a historical point of view, many interesting ideas about thinking and associations originate from this time and it is fascinating to see how scholars of the time were intrigued by the regularities, but also struggled with the irregularities of associations.

Law of contiguity
Law of similarity

For this work however, the psychological views starting to evolve from the late 19th century are of more importance. An interesting treatise on associations from this time can be found in [98, pp. 550-604]. Starting around this time, also the first experiments in the direction of collecting associations were reported by Galton [64]. Galton's method of collecting associations is in similar forms[3] still used today: Typically, a stimulus word is presented and the participant is asked to quickly name (or note down) the first thing that comes to mind. The response (and depending on the use-case also the response time) are recorded.

Collecting associations

When collected over many participants and stimulus words, these *primary word associations* show surprising regularities, a fact that led to their application in clinical psychology, e.g., in form of Jung's "Association Method" [100] for nearly a century. The collection of associations was repeated many times, across different demographic groups, locations and languages, and especially in the 20th century caused many publications in form of so called *word association norms*.

primary word associations

word association norms

While the early publications only include tabulations of a few hand selected stimuli and their responses, later experiments drastically expanded in size and took a more systematic approach. The most remarkable of these is the Edinburgh Associative Thesaurus [101] (EAT). It is to our knowledge the earliest and biggest available free-text association collection, containing ~ 788k associations collected directly from human participants. Unlike in previous experiments, modern computer technology of the time was used to form a gigantic association thesaurus, differing from a simple corpus in that the experiment was repeated in several rounds. Starting from a seed vocabulary of stimuli used in previous norms, the subsequent rounds primarily presented stimuli that were formed by the top responses of previous rounds, thereby generating a huge and well interconnected *association network*. Further explanations on associations can be found in Section 2.2.3. For more details on the EAT experimental setup, resulting dataset and its properties please refer to Section 5.1.

EAT

association network

After the EAT dataset, many later datasets started to focus on the rising interest in the connections between associations and linguistics, as will be explained in the following section. Even though not used

3 While Galton primarily reported experiments on himself, nowadays self-experimentation is the exception.

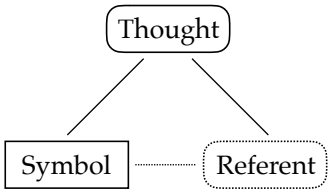


Figure 2.2: *Semiotic Triangle* after Ogden & Richards [136]

in this work, we want to mention one of the largest of these datasets, the University of South Florida free association, rhyme, and word fragment norms [131, 132] (USFA). *USFA*

2.2.2 *Communication and Linguistics*

While our aforementioned definition of associations focuses on connections between thoughts, associations are also closely related to communication and language [66, pp. 240ff]. No matter where an association originates from and leads to[4], it can (currently) only be communicated to others indirectly (not telepathically). In this work, we focus on the predominant form of such communication, which also allows us to record and use datasets of associations: spoken or written language.

In such language, we can differentiate between *symbol*s, *thought*s and *referent*s, as illustrated in the *Semiotic Triangle*[5] in Figure 2.2. When communicating about Kaiserslautern for example, we use the word "Kaiserslautern" (the symbol) to indirectly invoke the mental representation of Kaiserslautern (the thought) in another person's head, which hopefully stands for the same referent, the physical city of Kaiserslautern in Germany. While often present, in this work, we explicitly allow symbols and thoughts for which there is no easily identifiable referent. An example for this is the symbol "City", for which we have a thought in our mind, but can't easily present a real-world referent. Many other classes and concepts, belong into this category as well. *symbol, thought, referent* *Semiotic Triangle*

Much confusion can arise if these connections of the semiotic triangle are referred to as associations as well. Hence, in this work, *associations* in general do not refer to these connections, but to the connection of two symbols or the connection of two thoughts, as depicted in Figure 2.3, which is showing the *word association* "Kaiserslautern - City" and the corresponding *thought association* within the mind be- *word association* *thought association*

4 We explicitly want to point out that associations can occur within and across different modalities such as senses (visual, auditory, taste, smell, touch), explicit or implicit thought, emotions and instincts.

5 From a Semantic Web point of view, the Semiotic Triangle seems very familiar: symbols correspond to literals (`rdfs:label`s), thoughts to URIs and BNodes. Referents correspond to themselves. For further comparisons refer to Section 2.2.3.

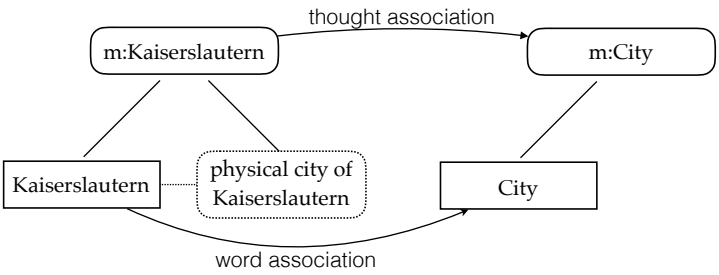


Figure 2.3: Difference of word associations and association of thoughts. The prefix `m:` here stands for "mind".

tween `m:Kaiserslautern` and `m:City`. In our example, `m:City` does not have a referent.

In communication, one of the biggest challenges arises due to the connection between symbols and thoughts not being a bijection: A single thought can have many symbolic representations, such as "K-Town" being an alternative name for the harder to pronounce "Kaiserslautern". Additionally, a single symbol can stand for many different thoughts (and thoughts of different levels of granularity and specialisation), causing many difficulties for the field of computer linguistics and being subject of its sub-fields Named Entity Recognition (NER) and Disambiguation. A simple example of such an *ambiguity* is the word "Jaguar" that can mean the animal or the car. Another example of an *ambiguity* caused by different levels of granularity is "Kaiserslautern", that can refer to both, the inner city, the city or the administrative district.

ambiguity

While still problematic for computer linguistics, humans quite successfully deal with these ambiguities in everyday communication. One of the keys for this are associations [66, pp. 240ff], which allow us to form an evolving *context* during communication, that allows us to disambiguate symbols to thoughts rapidly. A famous example for such a contextual disambiguation is the sentence: "Last year the pen was abandoned as it was too dirty for the animals to live in." [66, p. 241]. When first encountering the word pen, it invokes the thought of the writing instrument. Later however, when encountering animals in the same sentence, the symbol is re-interpreted to the less frequent meaning of pen as an enclosure for animals.

context

Given that associations play a central role in our communication, it is not surprising that works exist in computer linguistics that investigate the relations between text corpora and "associations". An exceptional overview over the field is given by Evert [51]. The vast majority of approaches focus on first (syntagmatig) and second (paradigmatic) order co-occurrences/collocations, but do not distinguish between relatedness, similarity and human associations. Further, many works use the word "association" with different meanings (e. g., in the sense of two words being "associated" if frequent collocations of them are

Computer linguistic

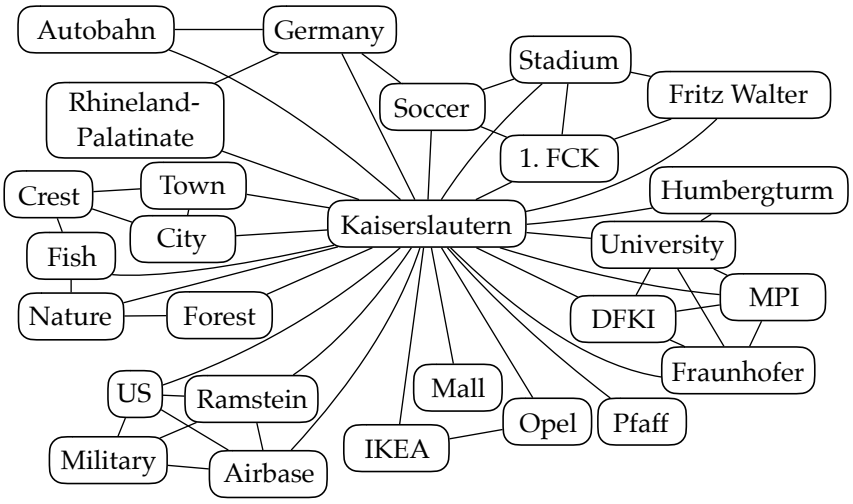


Figure 2.4: Contextual Semantic Network around Kaiserslautern

observable in a text corpus, or in the sense of "association rule mining") than this work, in which we focus on human or mental associations (cf. Section 1.3.1, Section 2.2). We are only aware of the works of Rapp [146], Washtell and Markert [180] and Galea and Bruza [63] that approximate human association strengths based on text corpus analyses and evaluate against existing psychological association collections. While such heuristics are relevant and very exciting for future work to provide big and up-to-date association training datasets (as mentioned in Chapter 13), in this thesis, we only use datasets collected directly from humans as ground truth, in order to avoid potential negative biases of any heuristic for our training data collection.

2.2.3 *Semantic Networks and the Semantic Web*

As shown in the previous section, associations are relevant for communication and context forming. Especially in the latter case, when looking at all encountered thoughts within a context, they form a graph of connected thoughts, also called *association network* or more general a *semantic network*. An example for such a contextual semantic network around Kaiserslautern can be found in Figure 2.4.

association network
semantic network

Semantic networks are nowadays often encountered in brainstormings or in form of mind maps. They were however first introduced as a spreading-activation theory of human semantic processing as the so called Semantic Memory [144]. The theory was formed trying to implement human memory search in computer simulations and based on experiments later refined to the so called spreading activation theory of semantic networks [40] and even later to the spreading activation theory of memory [6]. While sometimes criticised for being too unspecific, the theory explains how contexts are formed via

activation of encountered thoughts and spreading of that activation to associated thoughts.

Looking at the resulting network structures, the similarities to the Semantic Web become apparent. In fact, the edges in a semantic network are often directed and assigned textual labels, and semantic networks appear in early description logics long before the invention of the Semantic Web. Hence, when seeing semantic networks as a precursor of the Semantic Web, apart from its standardisation, the main novelty of the Semantic Web was its consequent use of URIs which unifies semantic networks with the ideas of the WWW.

Semantic networks: a precursor of the Semantic Web

human communication

symbolic indirection

When comparing *human communication* with the Semantic Web, the use of URIs turns out to be one of the main differences as well. Humans can only communicate about their thoughts via *symbolic indirection*: serialisation of thoughts to symbols on the sending party and de-serialisation of the symbols to thoughts on the receiving end. In this process, two thoughts (one in each brain) are involved, not one, which is a fundamental difference to the Semantic Web. In the Semantic Web, the use of URIs allows us to more directly link "into a different brain".

Furthermore, the Semantic Web differs from associative networks in that its edges are clearly labelled. In contrast to this, the connection types of associations are a lot less formal and composed of many different connection types (e.g., similarities, contrasts, hyper-/hyponyms, contiguity, to name just a few). Additionally, naming their type is often more challenging for humans than telling that they are somehow associated.

association strength

As a final difference, we want to mention that associations have different *association strengths*, as measured by the number of humans agreeing on an association, or more precisely the percentage of humans naming the same response to a given stimulus. Semantic Web edges in contrast are facts, which in a logical sense are not true to varying degrees.

This concludes our comparison of Semantic Networks with associations, association networks and semantic networks, and also our summary of the state of the art, on which this work is built. Further thoughts on the relation of human communication, associations and the Semantic Web can be found in our previous work [87].

3 RELATED WORK

After the state of the art in the previous chapter, we will present related works to our two main contribution areas in this chapter: those related to our semantic association dataset collection approaches in Section 3.1 and to our graph pattern learning approach in Section 3.2.

Before doing so, we however want to note other approaches, that try to approximate human association strengths based on text corpus analyses, such as those of Rapp [146], Washtell and Markert [180] and Galea and Bruza [63]. As mentioned in Section 2.2.2, such approaches are highly interesting for future work to provide big and up-to-date association training datasets (cf. Chapter 13). In this work however, as indicated by the title, we want to learn patterns from Linked Data to simulate human associations. This means that, rather than extracting further information by analysing text corpora, we primarily focus on learning from the already extracted (fully decomposed) existing knowledge (RDF triples) in form of graph structures.

Text corpus analyses

3.1 HUMAN ASSOCIATION RDF DATASETS

As previously mentioned, in order to develop a machine learning algorithm that can simulate human associations with Linked Data, a training dataset is needed. To the best of our knowledge, the only works in the direction of generating such a human association ground truth RDF dataset of *semantic associations* have been our own (and are detailed in Part II). In this section, we will however present other previously existing and related datasets. An overview over these datasets can be found in Table 3.1.

Several works have been published about creating fact ranking ground truth datasets, such as WhoKnows [105] and FRanCo [29]. In comparison to associations, fact ranking in general only focuses on existing facts. FRanCo, however, in its first step also collected free-text fact input about the entity in question, i.e., "Please tell us the most important facts about Munich". While the question formulation is useful for FRanCo's task to check for missing facts, it is questionable for collecting unbiased associations in the sense of this work. Nevertheless, it is conceptually closest to creating a semantic association dataset, as explained in Part II. In contrast to our mappings however, the published NER mapping of the free-text facts back to semantic entities[1] does not seem to have been manually verified and is very noisy. For these reasons, we currently do not use them for our

Fact ranking

1 http://s16a.org/node/13

work, but look forward to maybe in future verify and de-noise the mapping with our semi-automatic mapping approach.

EAT & USFA

Focusing on associations, the most relevant existing datasets for our work are the aforementioned Edinburgh Associative Thesaurus [101] (EAT) and the University of South Florida free association, rhyme, and word fragment norms [131, 132] (USFA). Both consist of large amounts of free-text primary word associations (EAT: ~ 788k, USFA: ~ 724k) collected directly from humans (mainly students). The EAT was collected in Edinburgh in the 1970s. The USFA collection started in Tampa from 1973, but continued to the 1990s. Apart from the different locations and slightly different time frames, the main difference between the datasets is the extensive post-processing performed by the authors of the USFA corpus focusing on linguistic aspects of the stimuli and responses, such as Part Of Speech (POS) tagging, rhyming and fragment overlaps. In contrast to the desirable semantic association in this work, both of these datasets consist of plain text associations that have not been mapped (and disambiguated) to semantic entities. Hence, in this work we primarily focus on mapping EAT (the bigger of the two datasets) to DBpedia entities (Chapter 5). An analogous mapping of the USFA is left for future work, as mentioned in Chapter 13. Due to existing overlap, especially w. r. t. strong associations, it is however to be expected that the benefit of the additional mapping with the same effort is much lower.

WordNet

Apart from the USFA dataset, many other linguistic datasets exist, such as the well-known *WordNet* [52, 125]. WordNet is a lexical database of words, listing their different meanings, grouping them into ~ 117k synsets (sets of synonyms), providing their word type (e.g, adjective, verb, noun), glosses (short description) and linking them to related terms (e.g., antonyms (hot - cold), hyper-/hyponyms (colour - red), holo-/meronyms (hand - finger)). Even though many heuristics exist that use WordNet's rich structures to measure relatedness or similarity of terms [32], none of the existing connection types in WordNet directly represents associations as described in this work. Multiple RDF versions [9, 119] offer interesting mapping targets for existing association datasets. However, their inter-linkage to the remainder of the LOD Cloud is weak in comparison to DBpedia.

BabelNet

Another famous linguistic dataset, *BabelNet* [48, 130], also misses association links, but would partially solve the problem of missing inter-linkage. Amongst others it interlinks WordNet, Wikipedia, Wiktionary, Wikidata, and DBpedia. BabelNet currently even provides its information as RDF and offers a SPARQL endpoint, but this access is by default limited to 1000 requests per day. Unfortunately, according to its authors, no RDF dumps are available or are planned to be made available in the future, making BabelNet unsuitable for machine learning algorithms such as ours (which perform in the order of millions of requests within minutes) and the conservation of its RDF

Table 3.1: Comparison of prior datasets: The *RDF* column indicates if accessible RDF entities are used. The *Relation Type* column abbreviations stand for: (A)ssociation, (F)act (R)anking, (I)mportance, (L)inguistic, (R)elatedness, (S)imilarity. The *Existing Only* column shows if the dataset is constrained to existing facts only.

Dataset	Relation Type	RDF	Existing Only	Size
FRanCo [29]	FR, I, (A)	✓	(✓)	+
EAT [101]	A			++
USFA [131, 132]	A, (L)			++
WordNet [9, 52, 119, 125]	L	✓		++
BabelNet [48, 130]	L	(✓)		++
DBpediaNYD [137]	R, (S)	✓	✓	+
WordSim353 [60]	S		(✓)	-
KORE [93]	FR, R	✓	✓	-

version for future research (e.g., via efforts such as the LODLaundromat [18] or LOD-a-lot [53]) all but certain.

Apart from such linguistic datasets, other datasets exist that focus more on relatedness and similarity. One of these is the *DBpediaNYD* dataset [137], consisting of the calculated symmetric and asymmetric Normalised Yahoo Distances (adaptations of the Normalised Google Distance [38]) between subject and object labels of about 7000 randomly drawn DBpedia facts. We do not see the dataset in the core focus of this work, as it represents relatedness and not necessarily associations in our sense and as it is not collected directly from humans (hence also called "silver standard"). However, due to its more than sufficient size and conceptual closeness of relatedness and associations, we applied our graph pattern learning algorithm to the top 1000 asymmetric relatedness scores and present the results in Section 11.1. *DBpediaNYD*

The *WordSim353* [60] is another famous dataset, which is often used in the Information Retrieval community. It consists of 353 word pairs that were prepared to represent various degrees of similarity. 29 participants were asked to rate the word relatedness w. r. t. similarity on a scale from 0 to 10. Sadly by question design, the resulting dataset mixes similarity and relatedness [2]. Due to this, its restriction to the given word pairs and unavailability of a verified RDF mapping, we do not compare against the WordSim353 dataset in this work. *WordSim353*

In Section 10.6 we will however compare against the *KORE* entity relatedness ranking corpus [93] that consists of 21 entities from 4 different domains (IT companies, Hollywood celebrities, video games and television series). For each of the 21 entities, a ranked list of 20 selected linked entities was created via a crowd-sourcing experiment in which all pairwise relatedness comparisons of each of the linked entities were performed by 5 participants each. Strictly speaking the *KORE*

KORE dataset is not an association dataset in our sense either, as it is too small for training and restricted to existing facts of which additionally only a sub-set is compared against each other. However, as all entities of the aggregated ranked lists are Wikipedia articles, which have a 1:1 correspondence to DBpedia URIs, this allows us to easily gain some insights by comparing our prediction model (which is fully trained on semantic associations) to a slightly different domain: entity relatedness ranking.

3.2 LEARNING FROM RDF GRAPHS

In this section, we will focus on machine learning and data mining approaches that are designed to extract information from graph structures such as the aforementioned RDF datasets. However, as there is a large and active community working on such approaches, and as excellent surveys of the field exist, such as [133, 148, 151], we will restrict ourselves to approaches most relevant for this work.

To the best of our knowledge, our developed graph pattern learning algorithm (Part III) is the first of its kind as it uniquely combines the following properties:

- It learns SPARQL BGP queries (without restricting directionality, paths or loops) for a given input list of source-target pairs (not a flat list of entities) directly from a given SPARQL endpoint.
- It learns an ensemble of queries and is designed for cases in which there is no single pattern that covers all source-target pairs. Each of the learned patterns can be used to predict target candidates given a source. A fusion component can be used to combine all target candidate lists generated by the pattern ensemble to generate a single ranked and scored list of predicted targets.
- The developed evolutionary algorithm is designed with scalability and real world considerations (such as noise and partial query results) in mind.

Also, we are the first to explicitly focus on simulating human associations with Linked Data. However, many other approaches that rank Linked Data (Section 3.2.1), learn embeddings (Section 3.2.2), and SPARQL queries (Section 3.2.3) exist and will be detailed in the following.

3.2.1 *Ranking Linked Data*

The need for Linked Data ranking mechanisms grows with the amount of available data and ongoing adoption of Semantic Web technologies [97]. Ranking methods for Linked Data are typically desirable

in situations where the information need of the user is not very specific (i. e., the user did not use SPARQL to refine the result to only the desired information in the desired order). In these situations, the amount of possible result facts and entities quickly becomes unmanageable. An example for such a situation is the common browsing use-case, in which a user wants to display an entity such as `dbr:Germany`. At the time of writing `dbr:Germany` has 1563 outgoing triples (1511 distinct objects over 94 distinct predicates) and 165314 incoming triples (123202 distinct subjects over 277 distinct predicates) on the public DBpedia SPARQL endpoint[2]. With that many facts in just the 1-neighbourhood of a node, it becomes hard to present a generic and concise result to users.

Browsing

In recent years, a variety of approaches have been developed to deal with such ranking or Linked Data entity summarisation [167] scenarios. In the following, we structure them into approaches which mainly analyse the graph structure of Linked Data itself and approaches which make use of external information sources for their ranking.

3.2.1.1 *Ranking by Graph Features*

As Linked Data forms a graph it is not surprising that many ranking approaches concentrate on graph inherent structural aspects.

The simplest of these approaches rely on graph intrinsic features such as simple node (in-/out-)*degrees*, according to which a candidate list is sorted. Slightly more sophisticated approaches use the pairwise neighbourhood overlaps like the *Milne-Witten Relatedness* [126] that focuses on Wikilinks. As soon as human generated links make up a large part of the interlinking, these approaches seem to work surprisingly well, as we will see when using them as baselines in Section 10.4.2.

degrees

Milne-Witten Relatedness

More sophisticated approaches often base their ideas on well-known ranking algorithms for the WWW such as *PageRank* [31] or *HITS* [102] and apply them to triples. From the perspective of this work, we count ObjectRank [14], Swoogle's OntoRank [46], Naming Authority [78], DING (Dataset Ranking) [44] (which was used in Sindice [172]), SUMMARUM [170] and LinkSUM [169] amongst others into this category of approaches.[3] While very meaningful for a general purpose ranking of large amounts of crawled Linked Data, none of the approaches explicitly rank according to (or compare to) human association strengths. As we focus on DBpedia, we represent these approaches by the pre-computed PageRank and HITS scores on DB-

PageRank

HITS

2 `http://dbpedia.org/sparql` (counts over all graphs, including ~ 120K incoming and ~ 1.3K outgoing `dbo:wikiPageWikiLink`s)

3 For a more extensive comparison of the approaches see our previous publication [86].

pedia entities [171], which can be found as the PageRank and HITS baselines in Section 10.4.2.

tensor factorisation

In contrast to the aforementioned algorithms, *tensor factorisation* approaches such as TripleRank [62] and RESCAL [134] exist, that can use more complex relations (spanning multiple predicates) between entities for ranking. TripleRank represents the RDF graph as 3D tensor and uses a tensor variant of HITS. By this, TripleRank would allow the identification of and grouping by similar properties. Sadly, in order to avoid problems with scaling, TripleRank includes a pruning step which removes properties that could potentially encode very useful information (such as `dbo:wikiPageWikiLink`). RESCAL is demonstrated to work on knowledge bases as big as YAGO [135]. In contrast to YAGO however, our scenario includes nearly two orders of magnitude more facts, and several orders of magnitude more predicates.

3.2.1.2 *Ranking by Graph External Features*

After the previously mentioned approaches, which focus on the graph structure itself, we now address approaches that make use of external information for the sake of completeness.

To incorporate external knowledge, many of the following methods rely on semantic similarity or semantic relatedness of terms. Typically, labels (if available [49]) are used to map between Linked Data entities and their textual representation in external data sources.

WordNet

To approximate semantic similarity or relatedness of two concepts, many approaches are based on the aforementioned *WordNet* [52, 125] and use features of its hierarchical structure, such as the length of shortest paths between concepts or the overlap of synsets. An extensive evaluation of WordNet-based semantic relatedness measures can be found in [32]. The main disadvantage of using WordNet as an external resource is that despite its size and quality it is far from complete and quickly becomes outdated (long gone trend words such as "iPod" are still missing).

WikiRelate

Other approaches, such as *WikiRelate* [165], try to overcome these issues by relying on Wikipedia and often focus on textual features of the articles, their overlap, structural Wikipedia features such as disambiguation pages, the hierarchy of categories, listings, and Wikilinks.

Some of the aforementioned approaches, especially those depending on WordNet and Wikipedia, nowadays can actually be performed on their Linked Data pendants. In contrast to our approach and the ones mentioned in the previous section, they however use very specific knowledge about these datasets (and their mappings to Linked Data), which makes them hard to adapt to other use-cases.

Another large group of similarity measures focuses on distributional aspects of words and their co-occurrence in large text corpora (e.g., online documents) or social online platforms. The underlying

idea here is that similar words appear in similar contexts, also known as *Harris' Distributional Hypothesis* [76, 77]). Approaches in this group are typically based on the count of contexts in which both terms co-occur, compared to the counts of contexts in which they occur independently and then try to estimate the significance of the co-occurrences. An extensive overview about such approaches can be found in [51]. Examples for these similarity measures include the aforementioned *Normalised Google Distance* [38] (often applied to other search engines as well, such as in the aforementioned DBpediaNYD) and tag relatedness in social bookmarking systems [37].

Harris' Distributional Hypothesis

Normalised Google Distance

Apart from approaches that can clearly be assigned into one of our categories, mixed approaches exist. An example for approaches that combine co-occurrence based measures with WordNet-based ones can be found in [2]. Another example for such a mixed approach is *DBpediaRanking* [127], which makes use of graph intrinsic features from DBpedia, but at the same time also uses external information. DBpediaRanking finds semantically related terms for a given DBpedia resource by exploiting the graph-based nature for a limited depth-first search restricted to predefined properties (`skos:subject` and `skos:broader`). The discovered nodes are then compared to the root node by a scoring mechanism that includes similarity measures derived from co-occurrences of both `rdfs:label`s in web documents by querying search engines such as Google and Yahoo and online bookmarking services such as Delicious. The scoring mechanism also ranks nodes higher which have bidirectional `dbo:wikiPageWikiLink`s with the root node (an idea which can also be found in [178]), and scores nodes higher which have bidirectional `dbo:abstract` inclusions of their `rdfs:label`s with the root node. DBpediaRanking thereby achieves promising evaluation results.

DBpediaRanking

We want to conclude our listing of related ranking approaches with a mention of recent approaches to rank type like relations [15, 16], such as the recent WSDMcup [90] winning (hybrid) approach [45], that combines many of the above mentioned ideas via an ensemble learning approach.

Due to the apparent successes of hybrid approaches, we consider an analogous combination of our graph-based learning approach with external metrics and features as promising future work, as discussed in Chapter 13.

3.2.2 *Embeddings & Vectorisation*

As graph knowledge on its own is incompatible with most traditional vector based machine learning approaches, in recent years a variety of approaches have been developed to convert graph structures into vector form, often called vectorisation, embedding learning or propositionalisation. Once generated, the feature-vector representation can

be fed into traditional machine learning algorithms and used for supervised machine learning tasks (e.g., classification, regression) or unsupervised ones (e.g., clustering).

While approaches exist that are applicable to graphs in general (e.g., node2vec [69]), for this work techniques that focus on RDF and knowledge graphs are of more interest, such as RDF2Vec [150, 152], Global RDF Vector Space Embeddings [39] or NASARI [33, 34, 96].

Word2Vec

RDF2Vec is based on *Word2Vec* [124], a method originally designed to learn word embeddings from large text corpora: Each word w of the corpus is represented by a vector v_w of length n (typically $100 \leqslant n \leqslant 1000$) that after training is located in the proximity of words that have similar context in the text corpus. Typically during Word2Vec training, the context is formed by a continuous bag of words (CBOW) or skip gram model operating on a sliding window around the word in question. In case of the CBOW, the bag of words is used to try and predict the current word, while in the skip gram model the current word is used to try and predict the surrounding context.

RDF2Vec

RDF2Vec [150, 152] makes use of Word2Vec's training approaches after first constructing the corresponding text documents (sentences) from RDF by localised graph walks (breadth first) or graph kernels (Weisfeiler-Lehman Subtrees) around each of the entities in question.

In contrast to preceding LSA [108], PLSA [94] and LDA [28] families of approaches, Word2Vec's vector space representations are also shown to capture *analogies* surprisingly[4] well and transform them into simple vector arithmetic. For example, the analogy that "King" behaves to "Queen" as "Man" behaves to "Woman" can be expressed a trained Word2Vec model as:

analogies

$$v_{\text{King}} - v_{\text{Queen}} \approx v_{\text{Man}} - v_{\text{Woman}}$$

and reformulated to:

$$v_{\text{King}} - v_{\text{Man}} + v_{\text{Woman}} \approx v_{\text{Queen}}$$

This compositionality aspect and especially the latter formulation makes Word2Vec based approaches such as RDF2Vec interesting for this work.

Another embedding technique shown to perform well on such analogy tasks is *GloVe* [139]. In contrast to Word2Vec, GloVe does not focus on word co-occurrences from a local window, but on a global factorisation of non-zero elements in a word-word co-occurrence matrix.

GloVe

GloRDFVe

The *Global RDF Vector Space Embeddings approach* [39] extends GloVe to work on RDF graphs. Unlike RDF2Vec, it does not rely on first transforming the existing graph structure into localised textual representations, but instead applies a fast personalised PageRank approximation approach and 12 different weighting strategies to directly form a

4 Word2Vec's formality and the lack of understanding why it works so well is not entirely undisputed in the NLP community (cf. [67, 113, 139]).

sparse co-occurrence matrix from the RDF graph structure on which GloVe's training is executed. Many Global RDF Vector Space Embeddings and RDF2Vec models trained on DBpedia entities are available online[5]. We use them as baselines for this work, as shown in Section 10.4.2.

Other embedding approaches of RDF knowledge exist, such as *NASARI* [33, 34] and SensEmbed [96] which are trained on different text corpora, while relying on the aforementioned BabelNet and its synsets as dimensionality reduction technique. The vocabulary mismatch between the available trained NASARI[6] and SensEmbed[7] models and DBpedia URIs, as well as their inconsistent format make it impractical to compare with our approach. In Section 10.4.2 we will however use one of their models as baseline.

NASARI

Many other approaches exist which learn vector representations from knowledge graphs. Extensive overviews about such algorithms can be found in [133, 148, 151]. However, one common disadvantage of many vectorisation approaches is their lack of explainability. In contrast to our approach, the individual dimensions of an embedding rarely have an assignable, human understandable meaning, even before applying machine learning algorithms on top of them.

3.2.3 *SPARQL Query Learning*

In contrast to the aforementioned vectorisation approaches, our algorithm learns SPARQL BGP queries directly from a given SPARQL endpoint for a given training list of source-target node-pairs. In comparison to embedding approaches, one advantage of SPARQL queries is that their components (in our case mostly triples with variables) are comparably easy to *explain*. In most cases, a simple textual representation via `rdfs:label` is already understandable and the underlying classes and predicates are often semantically defined in form of an ontology.

explainability

To the best of our knowledge our algorithm (cf. Part III) is unique in that it learns an ensemble of SPARQL queries modelling a list of source-target node-pairs with the goal of node prediction. However, other approaches exist that help to discover relationships between entities from SPARQL endpoints and to formulate SPARQL queries, which we will detail in the following.

The first approach we want to mention is the *DBpedia Relationship Finder* [112]. Amongst others, it pre-processes the DBpedia RDF dumps, loads the results into on a SQL database and uses SQL queries with n self joins to find paths of a given length n. During preprocessing each triple is duplicated and swapped in directionality (subject

DBpedia Relationship Finder

5 http://data.dws.informatik.uni-mannheim.de/rdf2vec/
6 http://lcl.uniroma1.it/nasari
7 http://lcl.uniroma1.it/sensembed/

and object swapped), to simplify the generated SQL query and increase efficiency. While the approach uses smart lower and upper bounds for the path length n from further pre-processing computations, the algorithm is designed to find (shortest) paths only, excluding loops. Further, for efficiency reasons the approach uses a configurable ignore list of predicates and objects. After entering two nodes, between which to find relations, the identified paths are visualised in an interactive interface.

RelFinder

Basing on ideas from the DBpedia Relationship Finder, the more generic *RelFinder* [88, 89] was developed with an even higher focus on an interactive user interface. Given two entities it shows "interesting" paths between them. In later versions, more than two entities can be given, in which case all paths between pairwise combinations are visualised at once. Like the *DBpedia Relationship Finder*, the approach only finds paths and uses a configurable ignore list to exclude paths with certain predicates (by default `rdf:type`, `rdfs:subClassOf`, `dbo:wikiPageWikiLink`) and objects. Further, directionality constraints are imposed on the paths, so that only paths are returned, in which the edge direction alters once at most.

Edge directionality

While both aforementioned approaches find "interesting" paths between entity pairs, they rely on extensive pre-processing and impose heavy restrictions on the edge directionality and properties used, in order to drastically reduce the search space and thereby complexity. Furthermore, being an "interesting" path according to these approaches does not mean that the path (or its corresponding SPARQL query) is a good predictor for our use-case and according to our definition of a good pattern in Section 7.3.1. The approaches treat both nodes between which to find paths as equals, and unlike us do not try to predict target nodes given a source node. Further, we will find that many of the best patterns are neither paths (they include loops or further edges apart from the path connecting `?source` and `?target`) and involve properties that are ignored for being too unspecific (e. g., `dbo:wikiPageWikiLink`) or lead to "uninteresting" connections (e. g., `rdf:type`).

AutoSPARQL

Unlike the previous approaches, that find paths between entity pairs, other approaches exist which try to learn a SPARQL query for a given flat list of entities. One of these approaches is *AutoSPARQL* [111], which is an iterative active learning variant of the Query Tree Learning (QTL) algorithm. The QTL algorithm is a supervised machine learning algorithm that learns a SPARQL query resembling a forward oriented tree with a root variable `?x0`. After learning, the bindings of `?x0` should resemble the given (positive) list of entities (i. e., a `SELECT ?x0` will return all entities from the positive list). Additionally, a negative list of entities can be given, of which none should be returned by the learned query. The algorithm proceeds by finding the Least General Generalisation of all positive sample resource

query trees and uses Negative Based Reduction to remove constraints until negative examples are matched. While the performance of the algorithm is impressive, it only works on forward oriented edges, excluding loops, and is neither designed to work on a list of source-target node-pairs nor to deal with cases, in which there is no single query that fulfils all positive samples. As many of the proofs of QTL's properties rely on the forward oriented tree structure of the learned SPARQL query, an extension of the algorithm to allow for loops, reverse edge following and to learn SPARQL queries that are good predictors for target nodes given a source node, seems far from trivial.

KRETR

Another flat list learning approach that has been developed in parallel to our graph pattern learner is *KRETR* [140]. KRETR, also called Learning To Query (LTQ), is an active learning approach that takes a list of positive and negative example entities and tries to learn a SPARQL query that returns all positive entities and none of the negative ones. The approach resembles a best first search of possible refinements of the current query `q` (containing a variable or node `x`) by adding an additional outgoing (`x ?p ?o`) or incoming (`?o ?p x`) triple and selecting bindings for `?p` or `?p` and `?o` in order of descending F_1 measure. While ideas such as adding of a random variable triple or variable instantiation in order of descending F_1 measure can also be found in our graph pattern learner (cf. Section 8.5), KRETR similarly to AutoSPARQL is not designed to learn SPARQL queries for source-target node pairs or deal cases in which no single query can fulfil all positive samples. Further, KRETR cannot learn SPARQL queries that have edge variables or contain loops that include node variables.

RapidMiner

Other tools such as the RapidMiner LOD extension [149] make it easy to incorporate LOD knowledge in a machine learning framework. SPARQL queries are however used in a rather manual way and mostly for simple pre- or user-defined data access, instead of learning the queries themselves.

This shortcoming is addressed by another RapidMiner extension for pattern based feature construction [109], that focuses on learning SPARQL patterns to use them as features for binary classification tasks of entities. It allows to answer questions such as: "Does an entity belong to a predefined class?" In contrast to this, our approach focuses on learning patterns between a list of source-target pairs for entity prediction: given a source entity the goal is to predict target entities. While it might in theory be possible to simulate target entity prediction for a single given source entity with binary classification, one would need to train one classifier for each possible target, making the approach computationally infeasible for our scenario.

With this we conclude our related work section and introduction part. In Part II, we will present our approaches to generate high-quality semantic association datasets, before presenting our evolutionary graph pattern learning algorithm in Part III.

Part II

DATASET GENERATION

In order to train a machine learning approach to simulate human associations with Linked Data, a ground truth dataset of *semantic associations* is needed. As no such dataset existed before this work, we explored three different approaches (two games in Chapter 4 and one semi-automatic mapping approach in Chapter 5) to generate large, high-quality ground truth datasets of semantic associations.

BetterRelations

The first of these approaches, called *BetterRelations* (cf. Section 4.2), is a browser game that collects human ranking preferences about existing facts (RDF triples). In each round, the players are presented with the textual representation of two facts about an entity and asked which one of the facts will come to their partner's mind first (e.g., for an entity like `dbr:Facebook` the game could ask to decide between "key person Chris Hughes" and "has subject Online social networking"). On agreement, the players earn points. Behind the scenes, the game uses TSSort [79], a noise resistant sorting algorithm which we developed to efficiently rank existing facts based on pair-wise user preferences.

Knowledge Test Game

As BetterRelations can only help to rank existing facts, we created a second game called *Knowledge Test Game* (cf. Section 4.3). This game resembles a Family Feud like setting: The players are shown a screen telling them that we asked 100 people to name something associated with a semantic entity (e.g., `dbr:Egypt`). The players are asked to guess what they said. The answers are entered via a text input field, which live disambiguates the entered string to semantic entities from DBpedia, involving Google and Bing searches, then lets the user select the entity they meant. If the same entity was selected by others, they are rewarded with points. By allowing users to freely enter any text, but forcing them to select an existing semantic entity, the game quickly collects high-quality semantic associations between (previously potentially not connected) stimulus and response entities.

semi-automatic mapping approach

While games allow us to make data-entry as much fun as possible, they still involve a lot of human effort. Hence, in a third approach (cf. Chapter 5), we focused on mapping an existing psychological human association dataset, the Edinburgh Associative Thesaurus [101] (EAT), to DBpedia entities. As mentioned before, EAT is a collection of ~ 788k free-text associations (e.g., "cat - dog") that were collected directly from students in the 1970s. With our *semi-automatic mapping approach* we were able to map about 1000 distinct most frequently occurring stimulus-response pairs to their corresponding DBpedia entity pairs. After each mapping was manually verified by 3 reviewers, we were able to collect a ground truth of 727 semantic associations corresponding to about 25.5k raw associations.

In the following, we will describe our three approaches in more detail, before concluding this part with an analysis of the results in Chapter 6.

4

GAMES WITH A PURPOSE (GWAP)

In this chapter, we describe two games that we developed to collect a human association ground truth between semantic entities: *BetterRelations* and the *Knowledge Test Game*. Parts of this chapter have already been published in [83–86].

As illustrated before (in Section 3.2.1), despite being a huge success for the Semantic Web, the increasing amount of available Linked Data also creates new challenges. Simple queries that lack specificity frequently return thousands of facts. Widely known examples of such queries are SPARQL *`DESCRIBE` queries*. For a given concept `:c` of interest on many SPARQL endpoints a `DESCRIBE :c` just returns the union of all outgoing `:c ?p ?o` and incoming `?s ?p :c` triples. The same holds true for the majority of resolvable URIs. Most ill-conceived, the results are often sorted alphabetically and often even truncated arbitrarily to limit bandwidth consumption.

DESCRIBE query

While this behaviour is acceptable for debugging, it most certainly is not what should be happening in productive systems which try to use the gathered information and in the end present the results to users. When simply asked about a URI, servers should return useful information, opposed to all information[1] they know, as already mentioned in the Linked Data Design Issues by Berners-Lee [21]. The problem with this rule is that without a given context, it is unclear which information is useful for a client. We can however observe that clients who are in a specific context typically have a specific information need and are able to formulate more specific SPARQL queries than a `DESCRIBE` query or resolving URIs. In other less specific cases, and especially when considering to truncate the results, we propose to rank facts according to human association strengths between entities, as human associations play a key role in human thinking, leading us from one thought to the next. This means that for an entity such as `dbr:Germany` which is strongly associated to `dbr:Berlin` we would like to rank facts between these two entities higher than facts connecting `dbr:Germany` and `dbr:Kaiserslautern`. Further, even if there is no fact already existing, we would like to create facts linking entities that are strongly associated by humans.

Useful information

However, to the best of our knowledge, before our work, no heuristic for or a dataset of human association strengths between Linked Data entities existed. Furthermore, collecting such a dataset is prone to subjectivity and changing context, it is monotonous and tedious to

1 Or worse: blindly biasing the returned facts towards those with URIs that occur first when sorted alphabetically.

collect and it is difficult for humans to reliably and objectively assess the absolute association strength between two entities in comparison to many others. Additionally, the immense amount of Linked Data would cause great expenses if a substantial amount of it had to be ranked or newly linked by associations with a traditional experimental set-up.

Challenges

With these challenges in mind, we designed the two games described in this chapter, following the Games With A Purpose (GWAPs) approach by von Ahn and Dabbish [5]. For a given Linked Data entity the games collect ranking preferences between facts or associated Linked Data entities directly from humans. Both games are designed to work in spite of subjectivity and efficiently aggregate a large amount of ranking or linking information into high-quality ground truth datasets. The generated ground truth datasets can be used to benchmark existing ranking techniques, or to develop completely novel approaches to simulate human associating on Linked Data, as will be presented in Part III.

GWAPs

The remainder of this chapter is organised as follows: We will first describe other related games (with a purpose) in Section 4.1, before describing BetterRelations in Section 4.2 and the Knowledge Test Game in Section 4.3.

4.1 RELATED GAMES

While many approaches to rank Linked Data exist, as mentioned in Section 3.2.1, we are not aware of any approach to collect or approximate human association strengths between Linked Data entities which also distinguishes them from similarity and relatedness. As BetterRelations and the Knowledge Test Game are two GWAPs to collect such data, we will discuss related games in the following.

GWAPs gained publicity with the introduction of the ESP Game [4], which turns the tedious process of labelling images into a fun game. In subsequent years, many more GWAPs were developed and their game design principles summarised in [5]. In terms of these design principles, BetterRelations and the Knowledge Test Game are symmetric output-agreement games.

With respect to game design, BetterRelations is most similar to Matchin. *Matchin* [73] is a two player web-game, which confronts pairs of players with two pictures, asking them which one they prefer. If the preferences of both players match, they are rewarded with points and an increasingly higher bonus. In case of a mismatch, they are not rewarded with points and the bonus (streak) is reset to 0. In the background, Matchin records the pairwise user preferences and uses them to compute a global rating of the played images. In contrast to Matchin, instead of letting the players choose between images, BetterRelations presents two textual facts corresponding to Linked Data

BetterRelations' role model: Matchin

triples about one topic. Furthermore, where Matchin creates a single globally ranked list of items, BetterRelations computes a ranking for each topic and its related facts. Therefore, the rating algorithm which transforms the pairwise user preferences into global ratings does not deal with one large list but multiple significantly smaller lists in BetterRelations. As detailed in Section 4.2.1, BetterRelations includes several additional features in order to make Linked Data issues such as noise or unknown facts tractable.

Knowledge Test Game's role models: Common Consensus & Family Feud

The Knowledge Test Game can be seen as a successor of our *Associator* [87] which is a pair-game to collect free-text associations for given topics. Its gaming principles are also inspired by *Common Consensus* [114], a Family Feud like web-game which asks its players to name common sense goals (e. g., "What can you do to watch TV?"). In contrast to Associator and Common Consensus, the Knowledge Test Game maps the entered answers back to existing Linked Data entities with its suggestion-box: With the help of parallel Google and Bing searches, the players' answers are immediately disambiguated and mapped to DBpedia entities. With this mapping, the game overcomes restrictions to previously existing facts and noise issues that pose a problem for BetterRelations. Players can even introduce new entities, should they be missing.

Beyond this, BetterRelations and the Knowledge Test Game are related to other GWAPs for the Semantic Web [162].

OntoGame

OntoGame [161] was the first and most prominent game with a purpose focusing on Linked Data. Its players are asked to decide if a Wikipedia topic is a class or an instance, aiming at creating a taxonomy of Wikipedia.

WhoKnows?

WhoKnows? [105], a single player game, judges whether an existing Linked Data triple is known by testing players with (amongst others) a multiple choice test or a hangman game. WhoKnows uses a limited fraction of the DBpedia dataset and excludes triples not matched by a predefined domain ontology in a preprocessing step. While this greatly increases fun by reducing noise issues, it also reduces the possibility to collect user feedback about the quality of excluded triples and to detect problems in the extraction process. In a variant *WhoKnows? Movies!* [168], a ground truth for entity summarisation in the movie domain is generated basing on facts from Freebase and compared with two entity summarisation approaches.

WhoKnows? Movies!

RISQ

RISQ! [181], a Jeopardy like single player game that generates questions from DBpedia, restricts itself to the domain of people after excluding non-sense facts in a preprocessing step. It then rates the remaining facts by using predefined templates to generate questions (clues) about subjects and tests if they are correctly recognised from a list of alternatives.

Typewriter

Other collaborative approaches exist that use input methods resembling games. For example, the two Freebase [30] apps *Typewriter*[2] and

Genderizer

Genderizer[3] used gamification to turn data input into a competition with global leader boards and high-scores. Answers taken from users in these interfaces were directly converted into statements (e. g., "... is female.") issued by the user and added to the knowledge base, taking them out of the list of items which lack information. In contrast to BetterRelations and the Knowledge Test Game, such input methods typically do not contain any means of filtering (possibly intentional) disruptive user input and do not provide edge weights.

Summarising, BetterRelations and the Knowledge Test Game are unique in that they collect human associations strengths between semantic entities. In contrast to many other games, they try to reduce the user's choice as little as possible and use aggregation and multiplayer agreement as filters against noise, subjectivity and spam. While BetterRelations collects ratings for existing facts only, the Knowledge Test Game allows its players to freely associate existing entities (independent of connecting facts) and even to introduce new ones.

4.2 BETTER RELATIONS

After the related games in the previous section, we will describe *BetterRelations*, a GWAP to rank facts by association strengths, in this section. We will introduce the game (Section 4.2.1) and what happens behind the scenes (Section 4.2.2), evaluate the game and the quality of its output (Section 4.2.3), before discussing our findings (Section 4.2.4) and our conclusion (Section 4.2.5).

The contents of this section have already been published in [84–86].

4.2.1 *The Game*

topic of interest

related triples

A simple approach to collect association strengths for Linked Data triples could look like this: First, we select a Linked Data resource of interest (e. g., `dbr:Facebook`). We call this a *topic of interest* or simply *topic*. We then show randomly shuffled lists of all "related triples" to test persons and ask them to order the triples by decreasing association strength. In the context of this game, given a topic `:t`, we define *related triples* to be the collection of (subject, predicate, object)-triples where the topic is the subject, so `:t ?p ?o`.[4]

2 https://wayback.archive.org/web/20111227045846/http://typewriter.freebaseapps.com:80/

3 https://wayback.archive.org/web/20110711055209/http://genderizer.freebaseapps.com/

4 Extending the list to triples where the topic is the object (incoming links), so `?s ?p :t`, typically imports a large number of unimportant facts for the topic. In Wikipedia and thus analogously in DBpedia one would expect to learn about Facebook by visiting the page about it, not by reading through all the pages linking to its page.

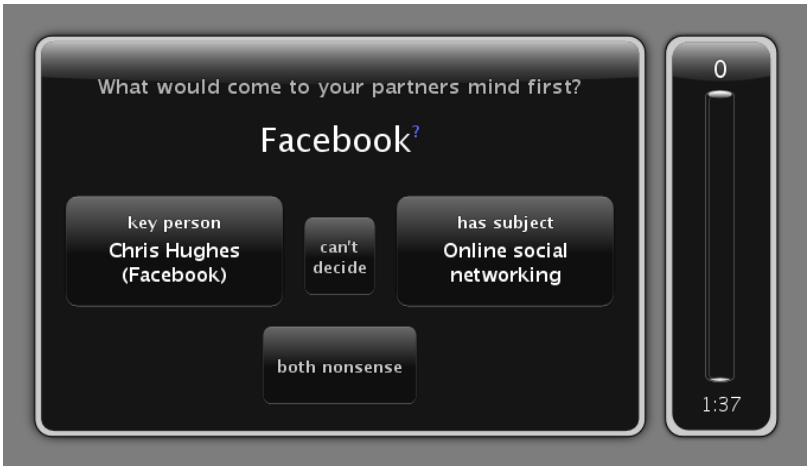


Figure 4.1: In a game round, choosing phase.

However, the aforementioned approach suffers from the problem that the outcome of each of these experiments, which is a user centric ranking, is not only highly subjective, but also unstable for one person over time. Furthermore, our early experiments showed that humans have problems with sorting lengthy lists reliably. In order to overcome such difficulties, we can instead ask for atomic relative comparisons of two facts about one topic and then use an objective rating algorithm to generate an absolute ranking of the topic's related facts. This leads us to the idea behind BetterRelations.

BetterRelations is a symmetric two player output (decision) agreement web-game in terms of von Ahn and Dabbish's design principles for GWAPs [5]: A player starting to play the game is randomly matched with some other player for a predefined timespan (e. g., 2 minutes). In every *round* (see Figure 4.1) both players are presented with a *topic*, which actually is a Linked Data resource's symbol (e. g., *Facebook*, the `rdfs:label` for `dbr:Facebook`), and two *items*, which are symbolic forms of facts about the topic (e. g., *key person Chris Hughes (Facebook)* and *has subject Online social networking*). As in Matchin [73] the facts are presented to the players in randomised order to counter easy cheating attempts.

BetterRelations

round

topic

items

Both players are asked to select the fact that their partner will have thought of first. This question formulation prefers less subjective associations, as it indicates that even if a person has some personal preference, they should pick the one they think others would prefer. In case a player does not know the topic, a quick information popup showing the article's abstract can be requested by clicking on the question mark appended to the topic. Doing so will internally mark the player's decision as influenced and the partner's as still to validate. To decide, each player can either click on the more important fact's button or on two additional buttons in case the player can't

decide between the alternatives or thinks that both alternatives are nonsense / noise.

As in Matchin, BetterRelations rewards agreements between both players with points and punishes disagreements without subtracting points, in order to increase game fun. The *scoring function* bases on the number of successive agreements (streak) in the current and preceding rounds: Players are rewarded with 0, 5, 10, 25, 50, 75, 90, 95, 98, 99, 99, 99, 100, . . ., 100 additional points for a streak of $0, 1, 2, 3, \ldots$ agreements, slightly resembling a sigmoid like curve. In contrast to Matchin (where the streak is reset to 0 on a mismatch), in BetterRelations a mismatch will only decrease the streak by 2 and does not reward the current round with additional points, which we found to be more motivating in case of high noise levels.

scoring function

BetterRelations includes two more buttons than Matchin: "can't decide" and "both nonsense". Hence, the scoring function was changed in order to counter easy cheating strategies such as always selecting the "can't decide" button. In terms of the scoring function, both middle buttons are the same button (it counts as an agreement if one player selects "can't decide" and the other "both nonsense") and an agreement on the middle buttons will not be rewarded with additional points, but will sustain the accumulated streak. Furthermore, a player who requested a quick info will not gain points in the current round.

After rewarding the players with points, the next round starts until the game runs out of time. The next topic is chosen by selecting the topic least often played by both players from a list of topics currently open for playing. In the end of a game, both players see a summary of their performance showing the amount of points gained in this game, the longest streak and their total game score in BetterRelations.

Topic selection

In case no partner can be found or the partner leaves the game, BetterRelations also provides a *single player mode*, which will either replay rounds with previous still to validate decisions or replay previous two player games if no still to validate decisions are left. As the latter replays usually waste human decisions, the single player mode can also be configured to initiate two player games with a certain probability and fake the (dis-)agreement by chance, based on the player's historical rate of agreements. The results of such rounds again provide new still to validate decisions, subsequently used by other single players.

single player mode

From a Human Computation point of view, BetterRelations can be interpreted as a framework to sort lists of facts related to topics by human association strength. BetterRelations controls the selection of topics and facts to play and outsources each atomic decision, which fact is preferable over another to human players. In this process, all player actions are recorded.

4.2.2 *Behind the Scenes*

On the server side, the game records a large amount of relative decisions between pairs of items, filtered by a partner and uses them to upgrade ratings in case of agreements. A both nonsense agreement will mark both items as nonsense and exclude them from future games in order to quickly reduce the amount of facts perceived as noise and reducing fun. Generating an absolute ranking from such results can be compared to chess rating systems, where based on the outcomes of atomic competitions (player p_1 won against p_2), a global ranking is calculated [73], just that in this case it is not players competing, but facts. In contrast to Matchin, we developed a TrueSkill [91] based algorithm called TSSort [79]. BetterRelations internally uses it to update fact ratings after each agreement and to select the next fact pair for a topic in a way to minimise the overall needed amount of decisions. We stop sorting lists with n facts after $n \cdot \log_2(n)$ updates, determined to be a good threshold by simulations.

Chess rating like

As BetterRelations is designed to rank multiple lists of triples related to one topic each, we initially have to decide which topics to present to players. Topics should be well-known to most players and interesting, in order to receive valuable feedback and provide an entertaining game. Additionally, each of the topics should have associated Linked Data triples. Hence, BetterRelations selects topics (Linked Data URIs) corresponding to the most often accessed Wikipedia article pages[5], which include articles such as Wiki, United States, Facebook, and Google. Every time the game needs a new game topic and its related triples (e. g., because an existing topic's facts were sorted), it loads the corresponding triples for the next topmost Wikipedia topic from a local DBpedia mirror, which also includes all used vocabularies such as `rdf`, `rdfs`, `foaf`.

Topic selection

As showing URIs to the end-users mostly confuses them, the users will always see the corresponding `rdfs:label`s instead. Hence, for each URI in the list of related triples of a topic, all English or non language tagged `rdfs:label`s are acquired. For URIs with multiple labels a best label is selected following a heuristic preferring language tagged literals and such which are similar to the URI's last part if still in doubt. We call this the *symbolic form* of a triple. Triples having the same symbolic form are merged from a game's point of view and such with missing labels for predicate or object are excluded from the game.

symbolic form

Finally labels and corresponding triples are excluded, which (due to long string length) don't fit into the game's window, end with suspicious file endings (e. g., .jpeg) or which have an object label equal to the topic's label ("Facebook label Facebook").

5 Statistics were aggregated from raw access logs, formerly available at `http://dom.as/wikistats/`, nowadays at `https://dumps.wikimedia.org/other/analytics/`.

4.2.3 *Evaluation*

After the previous sections described BetterRelations' concept, data acquisition and preprocessing, we will now provide a detailed evaluation of the game itself and of the generated output, followed by an evaluation of the achieved output quality.

4.2.3.1 *Measurements, Estimates and Questionnaire*

First, the game's concept and its realisation are evaluated by summarising measurements and derived estimates as common for GWAPs. Afterwards, the outcomes of a questionnaire are provided which was presented to players of the game.

MEASUREMENTS AND ESTIMATES In an 18 day period, the game was played by 359 Users (re-identified by cookies if possible). In this timespan, 1041 games were played, out of which 431 were two-player and 610 were single-player games.

The players played a total amount of 12k rounds submitting 14.7k decisions, out of which they selected 11.2k times an item, 2k times "can't decide" and 1.5k times "both nonsense". This led to an amount of 3.8k mismatches, 4.7k matches, including 3.8k item matches, and 840 non-item matches.

The total amount of time all players together played the game was 42 hours (rounds without any decisions were not counted, they summed up to 5 hours, 46 minutes, e. g., idle browser windows). With this, we can calculate the average time a decision takes to be 10.3 seconds. The *throughput*[6] of BetterRelations hence is 350 decisions per human hour of gaming. With the given numbers, we can also find out the *average lifetime play*, so the time an average player plays the game, to be about 7 minutes. Multiplication of both numbers gives us an *expected contribution* of 41 decisions per human.

throughput

average lifetime play

expected contribution

When repeating the above calculations for matches (agreements) instead of raw decisions, we arrive at a *throughput* of 112 matches per human hour of gaming, and an *expected contribution* of 13 matches per human.

Knowing that the 1000 most viewed Wikipedia articles and their corresponding topics contain 56k game items, and taking into account the observed nonsense ratio of $\frac{1}{10}$, we can estimate that in order to sort the facts known about the top 1000 Wikipedia topics, we would need 313k matches. In terms of players, this means that with the current implementation we would need about 23.9k players to sort the top 1000 Wikipedia topics or 24 players per topic.

QUESTIONNAIRE Aside from these measurements and estimates, we wanted to know if the game was fun to play and to collect feed-

6 For definitions of *throughput*, *average lifetime play* and *expected contribution* see [5].

Statement	μ	σ
The gaming principle was easy	4.43	0.77
I knew all topics	3.11	1.04
I knew all items	2.54	0.91
Too much nonsense	3.68	1.23
Too many irrelevant facts	3.57	1.13
The game was fun	2.66	1.04
I will play it again	2.34	1.29
Played online games before	4.20	1.33
Age	27.68	6.76

Table 4.1: Results of an online survey answered by 35 game players. Except from *Age* users could select answers from a 5-point Likert scale: 1 (Strongly disagree), 2 (Disagree), 3 (Neutral), 4 (Agree), 5 (Strongly agree).

back for possible future enhancements. For this, an online questionnaire survey was conducted among players of the game. The questionnaire was completed by 35 participants, mainly German (32) computer science students (23) or researchers (8), 31 male and 4 female.

Apart from background questions, the questionnaire consisted of a series of 5-point Likert scale items that are listed in Table 4.1 and comment fields asking what the participants liked, disliked and what they were missing. The summarised results in Table 4.1 show that most of the players were between 21 and 33 years old and had played online games before.

Limited fun

Unknown topics and items

Noise

The main result from the conducted survey is that the game in its current version is of limited fun and that the majority of people do not plan to play it again. From the collected numerical data we can also see that on average the participants did not know all the topics and knew even less of the game items. At the same time, most of the participants agreed that the game contained too much nonsense and too many irrelevant facts.

Apart from these numerical results, a view of the collected comments yields many common aspects. Many users mentioned that they liked the idea of creating a game to collect scientific data and the design of the game. In accordance with the numerical results, most users mentioned that they disliked the high amount of nonsense, consisting for example of unknown or cryptic abbreviations. Many participants also mentioned that they disliked the formatting of dates and often were confronted with facts they did not know anything about. Some of the participants also disliked the waiting period in the beginning of the game and complained about the mixture of German and English facts.

Noise

rank	sum	ns	predicate	object
0.0	14.0		has subject	Wikis
1.0	26.0		has subject	Social information processing
2.0	28.0		has subject	Self-organization
3.0	30.0		has subject	Hypertext
4.0	42.5		has subject	Human-computer interaction
5.0	47.5		has subject	Internet history
6.5	74.0	x	Jahr	2007
6.5	74.0	x	tag	10

rating	ns	predicate	object
19.41		has subject	Self-organization
18.33		has subject	Social information processing
15.78		has subject	Human-computer interaction
9.15		has subject	Wikis
5.34		has subject	Hypertext
-1.63		has subject	Internet history
4.24	x	Jahr	2007
4.21	x	tag	10

Table 4.2: Example Gold Standard (left) and Game Output (right) lists for topic *Wiki*. In this case, *predicate* and *object* are the symbolic forms of the corresponding triples from DBpedia.

Improvement ideas

Many of the participants also mentioned that they were missing a button "I don't know any of these" or an initial selection of own interests, so they would not be asked things they did not know that often. Many users requested a way to know who they were playing with and even suggested to make it possible to explicitly select a partner to play with. Some of the participants also suggested showing a high-score screen at the end of the game and including user accounts to save their own score and a recap phase after the game listing the questions and selected answers, showing their outcomes and providing more exploratory features.

4.2.3.2 *Output Quality*

Besides evaluating the game itself as a vehicle to collect data, the quality of generated results is of special interest in this work. As mentioned in the previous sections, the game calculates rating scores for the facts in each of the *topics' related triples* lists. The rating score can be used to order each of these lists, generating ordered output rankings. In the testing period, the game completed the generation of 12 such lists ordered by importance ratings.

Gold Standard list

In order to assess the quality of these lists, a *Gold Standard list* was generated for each of these 12 topics. The Gold Standard lists were generated by a test group consisting of 11 people who had played the game before. Each candidate was asked to manually sort each of the 12 randomly shuffled lists of related facts by importance after excluding facts that the candidate identified as nonsense. For each of the topics, the manually sorted lists were aggregated by summing up the ranks for each fact and afterwards sorting ascending by rank sum (Borda Count), forming the Gold Standard list. In this process, nonsense facts were appended to each list's end and given a rank according to the barycentre of all nonsense items in that list. In the aggregated list, a fact is said to be nonsense if the majority of test persons considered it as nonsense. An example of such a manually generated Gold Standard list can be seen in Table 4.2 (left).

Once a Gold Standard list is generated, the Mean Squared Errors (MSEs) can be calculated for each of the individual manually generated ranked lists. The MSE is computed as the average sum of squared rank differences of each fact in the list and can be seen as blue histogram bars in Figure 4.2.

Calculating the average of these MSEs (so the average error an individual human makes when compared to the Gold Standard) and the deviation thereof (seen as red dashed and dotted lines in Figure 4.2), we can compare the human results with the game's result (which is shown as green vertical bar).

Even though the statistics in Figure 4.2 should be interpreted with care due to the small sample size, we can observe that the game's result are within the 1σ interval of manually created lists in 9 out of 12 cases. In 3 cases (*ISBN*, *Halloween* and *Harry Potter*), the game's results are more than 1σ below average individual human performance, but in 6 cases better.

After this description of the game's evaluation and its generated output rankings, the results will be discussed in the next section.

4.2.4 *Discussion*

One of the main concerns when designing BetterRelations was the desired high-quality of its generated output ratings. This task was considerably complicated by the high amount of facts considered as noise by players. Nevertheless, the results of the evaluation show that the game's outputs are about as good as those of an average human in 9 out of 12 cases and even better than an average human in half of the cases.

High quality rating

We notice that even for the 3 less successful results, which correspond to topics *ISBN*, *Halloween* and *Harry Potter*, a randomly ordered list would have resulted in expected MSEs of 28, 42.5 and 104 respectively, which is much worse than the 19.62, 36.03 and 34.56 achieved by our approach.

We were however interested in what could have caused the problems. An investigation of the topic *Harry Potter* revealed that while the game item ((p,o) pair) "image caption · Complete set of the seven books" was marked as nonsense in the Gold Standard list, it is ranked as top item by the game, indicating that many players preferred it over other game items. A possible explanation for this is that players of the game had limited time for their decisions and maybe overlooked the erroneous predicate label in a rush, and their association was likely dominated by the more prominent and very useful object label. At the same time, the participants of our Gold Standard test group had no time restriction to select items they regarded as nonsense. This single misplaced item accounts for a large amount of the game's calculated MSE ($\approx$ 15), probably making the result much

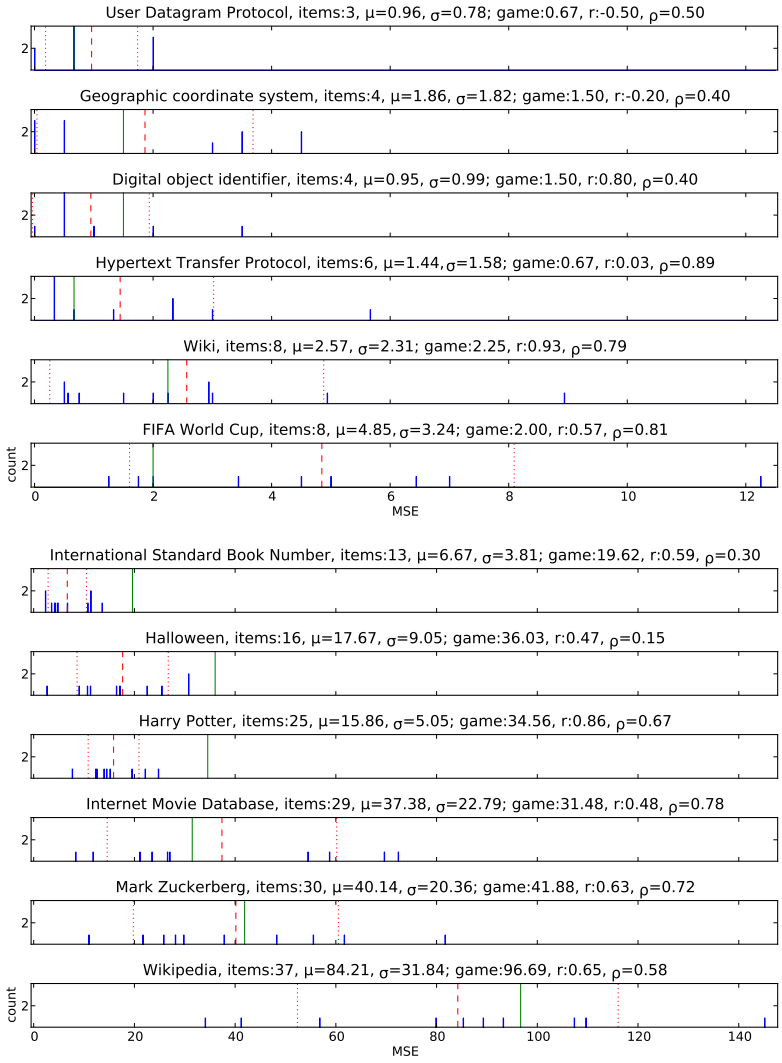


Figure 4.2: Comparison of Gold Standard and game output on 12 topics' item lists. Blue histogram bars show the MSEs of each manually generated lists, their mean μ is shown as a red dashed line, their standard deviation σ as red dotted lines. The game's MSE error is shown as a green line. The titles also include the Pearson product-moment correlation coefficient r and Spearman's rank correlation coefficient ρ of the Gold Standard List and the game's output.

worse than it is. In the results of *Halloween*, we noticed that the facts "has subject · Irish folklore", "has subject · Irish culture" and "has subject · Scottish folklore" were marked to be nonsense in the game results. Nevertheless, these game items receive suspiciously high ratings for nonsense items which, if they were not reordered to the end of the list as done in each of the human-generated lists, would have caused a much lower MSE value. Hence, we suggest to trigger a review in cases of such discrepancies between current rating and nonsense flagging in future versions. In the third of these lists for topic *ISBN*, we could not identify an obvious reason for the discrepancy.

Taking these considerations into account, we are confident that the game, already in its current version, generates good output ratings from pairwise comparisons of items.

Aside from the quality of the generated ratings, we also evaluated the game itself. The questionnaire reveals that the game principle was easy and straightforward and the majority of topics was known. However, problems related to fun and playability were also mentioned. An investigation of the given comments revealed that the primarily impairing factors were the presence of many cryptic abbreviations, strange formatting of numbers and dates, and the mixture of English and German facts. Since improvements of the game's fun factor would further decrease the amount of 24 players needed to sort the facts known about one Wikipedia topic, we performed further analysis on the reported problems. It turned out that many of them originated from errors in the DBpedia 3.5.1 dataset (e. g., German labels which had missing or incorrect language tags), and have been resolved in the more recent datasets. With progressing improvement of DBpedia's extraction templates, we expect more and more of these problems to be addressed. Such an enhanced quality of the underlying datasets has the dual effect of reducing the amount of (erroneous) triples to sort and at the same time increasing the fun of the game.

High noise levels reduced fun

4.2.5 *Conclusion & Outlook*

In this section, we presented a game called BetterRelations as well as a detailed evaluation of our implementation.

Our evaluation shows very promising results in terms of the desired and achieved high-quality of the generated collection of importance ratings. However, the low average lifetime play indicates a problem with the game's fun, which appears to be mainly caused by the high amount of noise in the underlying Linked Data triples. As even slight improvements of the average lifetime play could already drastically reduce the number of players needed to sort the facts known about a popular Wikipedia topic, and further, as DBpedia's extraction mappings are constantly evolving, attempts to re-run the game with a slightly more aggressive exclusion of irrelevant facts might be

interesting for future work. For this work however, we decided to concentrate our efforts on an orthogonal approach that focuses more on the players themselves and less on the existing facts in DBpedia, as described in the next section.

4.3 KNOWLEDGE TEST GAME

After BetterRelations in the previous section, we will now describe a second game, called the *Knowledge Test Game*. The game was developed to collect *semantic associations*, so human associations between semantic entities, without introducing a bias towards already existing facts in DBpedia. The game is also designed to work around the high amount of existing facts perceived as noise by players, which reduced the fun while playing BetterRelations.

After presenting the game in Section 4.3.1, what happens behind the scenes in Section 4.2.2 and its evaluation in Section 4.3.3, we will discuss our findings in Section 4.3.4, before concluding this section in Section 4.3.5.

Parts of this section have already been published in [83].

4.3.1 *The Game*

Knowledge Test Game

topic of interest

The *Knowledge Test Game* is available as standalone web-game[7] and as a Facebook Game[8]. Each game of the Knowledge Test Game has 2 to 10 players, all seeing the same *topic of interest* (or simply *topic*), which is a label of a Linked Data entity (in our case a DBpedia entity) for which we would like to collect associations. Players can play anonymously as guests or authenticate themselves by logging in using their Google or Facebook accounts. From the Knowledge Test Game website, players can choose to directly play a game or go through a *How to play* interactive tutorial.

game

When a player chooses to join a game, she either directly joins a random already running *game* or creates a new one and waits for at least one other player. A player can only join games that have less than 10 players, and have not been running for more than 70 % of their time. Additionally, the topic of the game being joined must be suitable for the player according to the topic restrictions (as will be explained in Section 4.3.2.2).

Once a player joins a game (see Figure 4.3), she is presented with the statement: "We asked 100 people to name something associated with **Egypt**. Try to guess what they said!", where "Egypt" is the current game's topic. The mention of 100 people is a white lie in order to remind of the well-known Family Feud TV show. Similarly to Bet-

7 http://www.knowledgetestgame.org
8 http://apps.facebook.com/knowledgetestgame/

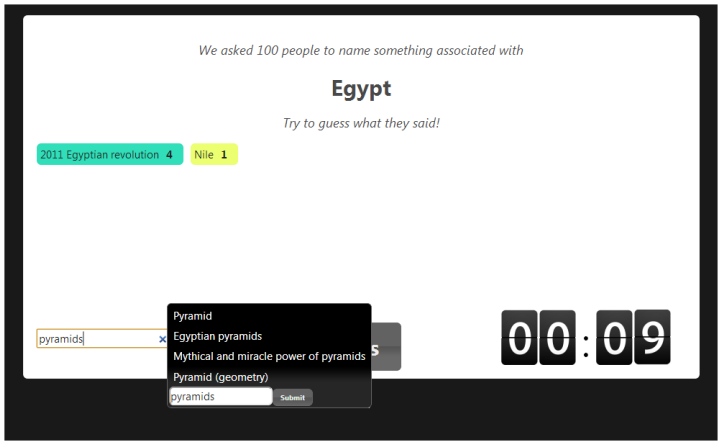


Figure 4.3: A player has submitted 2 associations already for the topic "Egypt" (scored 4 points for the first one and 1 point for the other), and is now viewing the suggestions after guessing "pyramids".

terRelations, the question formulation leads to a preference for more objective associations that are acceptable by the general public.

During a timespan of 45 seconds, indicated by a well visible timer, the players are able to submit guesses resembling what they think is associated to the topic. For each entered *guess*, the player gets a list of suggestions from which she can select the one most relevant to what she had in mind. The selected suggestion is then submitted as a *semantic association* to the topic. If none of the suggestions are satisfactory, the player's guess can also be submitted as it is. The process of displaying relevant suggestions is managed by the suggestions-box, which is further explained in Section 4.3.2.1.

guess

semantic association

Throughout the game, each player can see their own submitted semantic associations along with the score of each. The scores are increased dynamically when other players have submitted the same semantic association. Along with the question formulation, this motivates players to enter associations that others would agree on, consequently countering the subjective nature of the players' inputs.

When the game's time is elapsed, the players are forwarded to the recap page, where they can see the associations submitted by all other players, as well as their scores. Players can decide to join the next game with the same players or join a new game.

4.3.2 *Behind the Scenes*

Behind the scenes the Knowledge Test Game collects semantic associations for the given topics directly from human players.

4.3.2.1 *The Suggestions-Box*

suggestions-box

The primary interaction element for players is the *suggestions-box*. Its primary task is the immediate disambiguation of free-text user input to relevant semantic entities and submit the user's selection. In our case, both, the topics and suggestions, are `rdfs:label`s for DBpedia entities, allowing us to collect semantic associations directly from human players.

Disambiguation helps players to get points

Further, the suggestions-box makes it easier to match guesses. The disambiguation process is in our interest as well as the players', since we will be getting more useful information, and the players will be getting more matches (e. g., despite differences in spelling) and consequently better scores.

The Knowledge Test Game uses the submitted guess as a clue to suggest relevant semantic entities, and then stores the selected semantic association. For example, if the current topic is "Egypt", and three different players submitted "pyramids", "the pyramids" and "Egyptian pyramids". It would be challenging to detect a match, although they could have meant the same thing. Once the suggestions-box displays the suggestions for each of these guesses, the players would eventually pick the association that they meant, which could be "Egyptian pyramids", realising that it best matches what they had in mind. Consequently, the three players will get matches and therefore bonus points, and the game will give the association (`dbr:Egypt`, `dbr:Egyptian_pyramids`) a higher rank than for example (`dbr:Egypt`, `dbr:Pyramid_(Geometry)`).

Disambiguation approaches

As the primary goal of the game is high-quality data-collection, not the development of an own information retrieval approach, we decided to make use of existing techniques to realise the suggestions-box and make the user-experience and fun in the game as pleasant as possible. We compared several approaches, including SPARQL searches on the public DBpedia endpoint[9], DBpedia Spotlight [123], the search APIs of Freebase [30], Wikipedia[10], as well as the two commercial search engines Bing[11] and Google[12]. To return DBpedia entities in the latter two, all search phrases were extended with the word "Wikipedia" and the results filtered for Wikipedia URIs. Unsurprisingly, speed- and quality-wise the commercial search engines outperformed all other approaches, leading us to base the suggestions-box on a combination of Google and Bing query results. For an entered guess in each search engine 3 queries are performed in parallel: one consisting of the topic and the guess, one of the guess only (boosting result ranks) and one of the topic only (damping result ranks). After fusion of the results, the suggestions are shown to the players.

9 http://dbpedia.org/sparql
10 https://www.mediawiki.org/wiki/API:Search
11 http://www.bing.com
12 http://www.google.com

Further details about the suggestions-box and the various evaluated approaches can be found in [83].

Additionally, basing on Google and Bing allowed us to let our players benefit of many of their features as well, including implicit autocorrection and high tolerance against different representations of the same word. For example, submitting the British "organisation" and the American "organization" will result in two very similar, if not identical, suggestion lists. Players can even enter hints to the association instead of an exact association name. For example, a player can submit "c inventor" as a guess for "Deaths in 2011", and get a suggestion list that includes "Dennis Richie", who died in 2011, and who is also the inventor of C. Furthermore, the suggestions-box supports inputs in multiple languages, including complex ones such as Arabic, or even transliteration of Arabic words in English literals, and still yields relevant results. Regardless of the used text-input language, the resulting suggestions always correspond to English DBpedia entities.

Additional features

If despite these efforts, the player does not find a suitable association, they are additionally allowed to submit their guess as it is, by using the *other box* at the bottom of each suggestion list. Submitting a guess this way allows the player to come up with own associations which are not well represented or outside the scope of Wikipedia (and thereby DBpedia), at the expense of making it harder to match with other players. In order to get bonus points for an association submitted using the *other box*, other players have to submit the exact same string. In case the player uses the other box, the game creates new Linked Data entities with URIs of the form `ktg:<topic>/association/<association>` (for a discussion of this see Section 4.3.4).

other box

4.3.2.2 *Topic Selection*

Presenting players with topics that they are familiar with further increases the fun factor of the game, as well as the validity of the results, since users with interest in a topic are more qualified to provide valid associations. Similarly to BetterRelations, in order to focus on topics that are likely to be known, we collected the top most visited 10k Wikipedia articles in 2011[13]. Knowing that each of these articles corresponds to a DBpedia entity, the topics are randomly selected from their titles.

During topic selection further, player based restrictions apply. For example, a topic will not be presented to a player more than once, in order to exclude possible influences from earlier games. Additionally, as the Knowledge Test Game is also available on Facebook, for players logged in via their Facebook account, it will use their Facebook likes

Topic restrictions & preference

13 Statistics available at `http://dumps.wikimedia.org/`.

to prefer topics the users are likely to know more about due to their interest.

topic done

After 50 unique players provided associations to a topic, the topic is marked as *done*, and can be prevented from appearing in further games. This allows us to analyse the collected associations, and automatically re-focuses the collective efforts on other topics. In the process, the topic selection algorithm tries to close topics as early as possible, meaning that if there are several topics available for a game (following the mentioned restrictions), the one that was already played most is preferred.

4.3.2.3 *Generated Dataset*

Most of the data logged during the game is made available online[14]. The main components of interest are the players' guesses and submitted semantic associations. For every guess, the entered string provided by the player is stored along with the list of suggestions that were shown afterwards. We also log the game's ID, the topic's label and URI, as well the player's ID and account type (the ID hides all potentially personal information about the player).

When the player selects an association from the suggestion list, the record is updated to further include the association's URI and its index with respect to the suggestion list. The time of submitting the guess, and the time of choosing the association are both stored. We also keep track of the time taken by the player, in milliseconds, to choose the association from the list. The aggregated number of occurrences and the score of the association across the game are also logged. Furthermore, each record holds "nth guess" and "nth association" which show the record's submission order as a guess and its order as an association by that player in the given game.

4.3.3 *Evaluation*

After the previous sections focused on the game, its suggestion box and topic selection, we will now provide a detailed evaluation of the game and the generated output.

4.3.3.1 *Measurements, Estimates and Questionnaire*

First, the game's concept and its realisation are evaluated by summarising measurements and derived estimates as common for GWAPs. Afterwards, the outcomes of a questionnaire, which was presented to players of the game, are provided.

The game was run in several focused experiments, that added up to 26.6 hours of game-play time by humans. In these experiments, the

14 http://knowledgetestgame.org/export

game was played by 267 different players who played a total of 1046 games together collecting 6882 ranked associations.

Using these numbers, we can evaluate the game w. r. t. the *throughput*, *average lifetime play* and *expected contribution* metrics for Games with a Purpose defined by von Ahn and Dabbish [5].

throughput

The *throughput* is calculated by dividing the collected data (6882 ranked associations) by the total human game-play time (26.6 hours), resulting in ~ 259 ranked fully disambiguated semantic associations per human hour.

average lifetime play

We can also compute the *average lifetime play* by dividing the total game-play time (26.6 hours) by the number of players (267), resulting in an average lifetime play of ~ 6 minutes per player, which is equivalent to the time needed for ~ 8 games .

expected contribution

Finally, we can calculate the *expected contribution* by multiplying the average lifetime play with the throughput, resulting in an expected contribution of ~ 25.78 ranked associations per player.

Questionnaire

Apart from these measurements, we conducted an online survey which was filled out by 21 players after playing the game. Most of the participants were students from Egypt and Germany, between 20 and 25 years old, had a computer science or engineering background, had played web games before and described their English skills as fluent. Besides these demographic questions, the survey consisted of 3 open and 13 5-point Likert scale questions. The 3 open questions were asked beforehand in order not to influence participants. The text of the questions was: "What did you like about the game?", "What did you dislike about the game?" and "What would you improve?".

Fun, challenging

Summarising, most players liked the idea of the game and described it as fun, mentally challenging and interesting to compare their own thoughts to those of others. Many participants mentioned that they enjoyed the topic mix and were positively surprised by the quality of the suggestions-box:

> It is very challenging, not only are you challenging the other players, but also your own knowledge. The topics are very good. The recommended words are very good, Ex. I got the topic "Princess Diana" and I wanted to add the name of the man she was with in the car accident but I couldn't remember his name, I just know he was Egyptian, I wrote down "Egypt" and I found "Dodi Al Fayed"... very cool! :)

Unknown topics

In response to the dislike question, it was mentioned that some topics were too vague or unknown, that the suggestions-box sometimes was slow and that the 45 seconds per round were not sufficient to enter all associations in some cases. Also some participants complained about the little information they got about other players which was also mentioned in the following improvements question.

Table 4.3: Results of an online survey answered by 21 game players. Users could select answers from a 5-point Likert scale. If not indicated otherwise the options were: 1 (Strongly disagree), 2 (Disagree), 3 (Neutral), 4 (Agree), 5 (Strongly agree).

Statement	μ	σ
The game rules and concept were direct and straight forward.	4.5	0.8
The How To Play tutorial was... (useless ... useful)	4.8	0.4
45 seconds for the game were... (too short ... too long)	2.6	0.7
The topics were clear and know to me.	4.0	0.8
The suggestions were relevant to what I had in mind.	4.1	1.0
The suggestions that I got for a guess influenced my following guesses.	4.0	0.9
15 seconds for the recap page were... (too short ... too long)	3.1	0.6
I understood the recap page.	4.6	0.6
I was interested in reading the scores in the recap page.	4.5	0.7
Seeing my partner's answers influenced my guesses in the following games.	3.2	1.3
I enjoyed the game.	4.5	0.7
I would play it again	4.3	1.2
I played web games before.	4.0	1.2

Improvement ideas

Many participants mentioned that they would like to know more about the people they are playing with and suggested to introduce a chat after the game in the recap phase. Others want to be able to play with their friends. Moreover, participants mentioned that they would want to see global high-scores after the round and live statistics of other players in their game during the game, so they don't have to wait for the recap page to see their own performance. Furthermore, it was suggested to provide the ability to select categories of topics to play, to show photos for the topic or for vague topics to provide hints by showing some of the most often entered associations.

The findings from the open questions were refined by 13 questions in which participants could select numerical values between 1 and 5 (5-point Likert scale). The results are summarised in Table 4.3. In general, we can see that the game concept was easy to understand, people found the tutorial useful, knew the topics, found the suggestions relevant to what they had in mind, understood the recap page, were interested in it and that most people enjoyed the game and would play it again. The timing restrictions of 45 seconds per round was perceived as slightly too short, but 15 seconds for the recap page were just right.

Influence by suggestions

The questionnaire identified a key problem, namely that many participants had the feeling the suggestions-box influenced their following guesses. This effect was later mitigated by reducing the suggestions from ten to four (see Section 4.3.4). The effect seems to be less pronounced for the recap page.

Before discussing these findings and possible solutions, we first want to present our evaluation of the data collected.

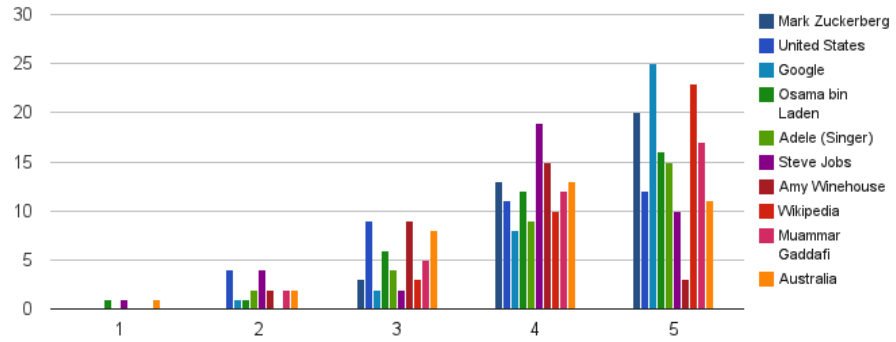


Figure 4.4: Histogram of ratings for the ordered lists of associations. For each topic the participants could chose on a scale from 1 (Makes no sense at all) to 5 (Makes perfect sense).

4.3.3.2 *Data Quality*

In order to assess the quality of the collected data, we aggregated the associations collected by the game for each topic. Focusing on topics for which the most associations were submitted by players, we counted the number of occurrences of each association and ordered them descending by counts. In this process, we excluded associations which were submitted by less than two players as a provisional filter against noise.

After the first major experiment, the resulting ordered lists of associations for the 10 topics which were played most often were generated. With these lists we conducted another online questionnaire with 36 participants out of which 19 had played the game. The participants' demographics resembled those of the game players: they mainly were computer science students from Egypt and Germany, between 20 and 25 years old and described their own English skills as fluent. In the questionnaire, for each of the topics we asked the participants to rate the ordering of the list of associations on a scale from 1 (Makes no sense at all) to 5 (Makes perfect sense). The histogram of the ratings can be found in Figure 4.4 and with an average over all ratings $\mu = 4.2$ ($\sigma = 0.9$) indicates that the majority of participants were very satisfied with the presented associations and their ordering.

Questionnaire

After a second large experiment, we chose another form to evaluate the generated association lists (an example can be seen in Table 4.4). We again conducted an online survey, this time with 17 participants, who were asked to rank randomised lists of the top associations for the (at the time) most often played topics. By then, we had 15 *done* topics (i.e., played by more than 45 players). Out of these 15 topics the 9 lists summarised in Table 4.5 were picked to form a ground truth, as they had been ordered manually by more than 5 participants. The *ground truth* was formed by averaging the individual ranks of the manually ordered lists of the participants and sorting the associations accordingly. Afterwards, the Normalised Discounted Cumu-

ground truth

Table 4.4: The most frequently submitted associations for the topic Mark Zuckerberg

Association	Times mentioned
Facebook	50
The Social Network	15
Chief Executive Officer	12
Rich	8
Millionaire	7
Social Network	6
Entrepreneur	5

Table 4.5: The 9 most often played topics whose top n collected associations were ranked by $p > 5$ participants. During the questionnaire the participants were presented with randomly shuffled lists and asked to reorder them. The resulting aggregated ranks were then compared with the NDCG to those generated by the Normalised Google Distance (NGD) and the game.

	Top n	Sorted by	NDCG	
Topic	associations	p part.	NGD	Game
dbr:Charlie_Sheen	8	7	0.860	0.969
dbr:Eminem	11	14	0.870	0.931
dbr:Lady_Gaga	18	9	0.806	0.924
dbr:Mark_Zuckerberg	7	15	0.895	0.954
dbr:Osama_bin_Laden	12	7	0.814	0.835
dbr:Transformers:_Dark_of_the_Moon	18	6	0.768	0.926
dbr:United_Kingdom	14	7	0.806	0.873
dbr:World_War_II	17	17	0.876	0.953
dbr:YouTube	10	17	0.927	0.928
		μ	0.847	0.921
		σ	0.051	0.042

lative Gain (NDCG) was calculated to compare the manually ranked ground truth association lists with those retrieved by the game. As a relevance metric, we used a linear mapping of the top element to a relevance of 1 down to the last element with a relevance of $\frac{1}{n}$.

In order to differentiate our game's results from simple corpus based similarity metrics, we also re-ranked the ground truth lists according to the popular Normalised Google Distance (NGD) [38]. As the NGD calculates a similarity between pairs of entities only and cannot trivially be used to find the top candidates for a given topic we artificially enhanced the method by only focusing on the top-20 candidates in the ground truth. The corresponding NDCGs for the NGD can be found in Table 4.5 as well.

4.3.4 *Discussion*

After detailing our evaluation in the previous section, we will now discuss our findings. In summary, we are very satisfied with the results of our evaluations, as the game was well perceived and fun for the players and also collected associations of high-quality.

We consider the achieved throughput of 259 associations per human hour as quite satisfactory, as it means that on average less than 14 seconds were spent for typing in a guess string, waiting for the suggestions-box and selecting one of the alternatives. As many players complained that the suggestions-box was slow we investigated our server logs to find that under high load it seems our requests to Google were rate limited, resulting in an average response time of the suggestions-box of ~ 2.3 s. At the same time, all 3 requests to Bing on average return within 250 ms. As we also got a lot of feedback that the quality of the suggestions-box is astonishing, we would like to keep using the merged results of Google and Bing. A possible solution to decrease delays might consist of more aggressive caching.

High throughput, suggestions-box rate limited

In order to solve ambiguity issues of the strings displayed in the suggestion list, another idea would be to extend the display with a `rdfs:comment` or a useful `rdf:type` from DBpedia. At the same time, a `foaf:depiction` could be shown to make the suggestions visually recognisable.

The evaluation also revealed the problem that later guesses were likely to be influenced by the displayed suggestion lists for preceding guesses. Throughout the experiments we therefore collected the index of the association selected from the suggestion list. As on average the second suggestion was selected with a standard deviation of 1.7 in the first major test, we reduced the amount of shown suggestions to 4 to mitigate the influence. Another alternative would be further reducing the amount of suggestions and providing a "more" button.

Reduced influencing

We were very pleased with the evaluation of the data quality, as the game shows a high average NDCG of 0.921 (Table 4.5) in comparison to the ground truth. The comparison to a popular corpus based technique shows that even when enhanced with an oracle that only suggested the associations we consider correct, the corpus based technique still was not able to rank the associations as well as the game (average NDCG of 0.847).

Last but not least, we investigated a potential design issue of our approach, resulting from its design to link the topic to other (but not the same) Linked Data entities. Our approach thereby neglects the possibility that people could want to associate a Linked Data entity with one of its alternative labels. Hence, we studied the list of all associations which were submitted with the "other" option of the suggestions-box and all guesses for which no association was selected, coming to the conclusion that not a single one of them corre-

Rare use of other box

sponded to a desired but missing alternative label in the suggestion lists. Also we were surprised how seldom players seemed to have missed an association target. From this we conclude that even though theoretically possible it seems to be very rare that people would want to associate an entity with one of its alternative labels or cannot find a desired association target in the domain of Wikipedia.

4.3.5 *Conclusion & Outlook*

In this section, we presented our Knowledge Test Game, a GWAP collecting semantic associations between Linked Data entities directly from human players.

Our evaluations show good results w. r. t. throughput and perceived fun of the game, as well as the quality of the collected data.

Apart from the potential improvements mentioned in Section 4.3.4 one of the main challenges is making it more desirable for players to stay in the game in order to be able to collect more data. This can be tackled, e. g., by providing a chat on the recap page, global high-scores, an exponential scoring scheme, player ranks and permissions (such as reporting cheaters). Another idea is to experiment with social gaming aspects such as team games by taking more advantage of the Facebook integration. Furthermore, a transparent single-player mode in which players play against recorded sessions of other players could help to reduce waiting times, validate existing data and detect cheaters.

Last but not least, while for this work a simple aggregation of the association counts is sufficient, future researchers might be interested in aggregation methods that for example take the submission order and timings of the associations into account. For this reason, the published data[15] includes not only the aggregates, but the raw individual players' guesses and selections, their order as well as all timings.

15 http://knowledgetestgame.org/export

5 MAPPING OF EXISTING DATASETS

After describing two attempts to collect semantic associations from scratch in the previous chapter, in this chapter, we will present an approach to re-use existing (psychological) free-text association datasets and semi-automatically map them to semantic entities from DBpedia in the centre of the LOD Cloud.

One such dataset, which has already briefly been introduced in Section 3.1, is the Edinburgh Associative Thesaurus [101] (EAT). We will extend our description of it in Section 5.1, before explaining our association vocabulary and how EAT can be transformed into an RDF Dataset in Section 5.2. In Section 5.3 we will then describe a semi-automatic mapping approach from EAT's free-text associations to semantic associations between DBpedia entities. We conclude this chapter in Section 5.4.

Parts of this chapter have already been published in [81].

5.1 EDINBURGH ASSOCIATIVE THESAURUS (EAT)

The Edinburgh Associative Thesaurus [101] (EAT) was created in the 1970s and is a dataset of single free-text *associations* collected directly from humans. The associations were collected in several rounds, starting from a seed list of common words as *stimuli*. In the following rounds, frequent *responses* of the previous rounds became the new stimuli. Over all rounds, a total of ~ 8200 stimuli were each presented to 100 participants (mainly students from Edinburgh) in randomised order. Each participant was allowed to write down one response for each stimulus, hence also called *primary word associations*. An example of the aggregated responses for 8 different stimuli can be found in Table 5.1.

stimulus

response

primary word associations

By this, Kiss et al. managed to create a well-connected network of ~ 788k *raw associations*, which form ~ 326k *unique associations* (unique stimulus-response pairs) between 8200 unique stimuli and ~ 22700 unique responses. About 5000 unique associations occur more than 20 times (167k raw associations). In the remainder of this work, we will refer to them as *strong associations* (associations with responses that occur in $\geqslant$ 20% of the cases). An example for such a strong association is the one between the stimulus "dog" and response "cat" which occurred 57 out of 100 times.

raw associations

unique associations

strong associations

Table 5.1: Example of EAT associations. Responses are aggregated for 8 different stimuli. The counts denote how many participants gave the response. We only show responses given by at least 2 participants. (As each stimulus was presented to 100 participants, the counts can also be interpreted as percentages.)

stimulus	dog		cat		man		woman	
responses and their counts	cat	57	dog	49	woman	66	man	59
	collar	5	mouse	8	strong	3	sex	5
	bark	3	black	4	human	2	girl	5
	leg	2	mat	3	hole	2	female	3
			tom	2	boy	2	child	2
			kitten	2				
			gut	2				
			eyes	2				
			animal	2				

stimulus	day		night		golf		dentist	
responses and their counts	night	52	day	52	ball	28	teeth	30
	light	13	time	10	club	26	pain	14
	time	8	dark	8	tee	5	chair	11
	dream	3	sleep	4	clubs	5	tooth	9
	sun	2	ride	2	balls	4	drill	8
	nigh	2	black	2	stick	2	doctor	7
	long	2	bed	2	grass	2	white	2
	life	2			game	2	waiting	2
	break	2					surgery	2

5.2 ASSOCIATION VOCABULARY AND RDF VERSION OF EAT

The original EAT dataset could be downloaded on the project's website[1] till late 2016, but was not available as RDF. Hence, we will now describe how the dataset can be transformed into RDF.

We can formally model EAT as a multi-set of raw associations. Each raw association $a \in EAT$ is a free-text stimulus-response pair: $a = (s, r), s \in S, r \in R$. The union of all stimuli S and responses R forms the set of terms $T = S \cup R$. While the original EAT corpus capitalises all terms, we will write an association $(\text{PUPIL}, \text{EYE})$ as "pupil - eye" for easier readability.

occurrences

Further, we can define the count $c_{s,r}$ as the number of *occurrences* of the raw association

$$c_{s,r} = |\{(s, r) \in EAT\}|$$

frequency

and the relative *frequency* $f_{s,r}$ as the relative count of response r with

1 http://www.eat.rl.ac.uk/, thankfully archived by the web archive: http://web.archive.org/web/20161030032628/http://www.eat.rl.ac.uk:80/

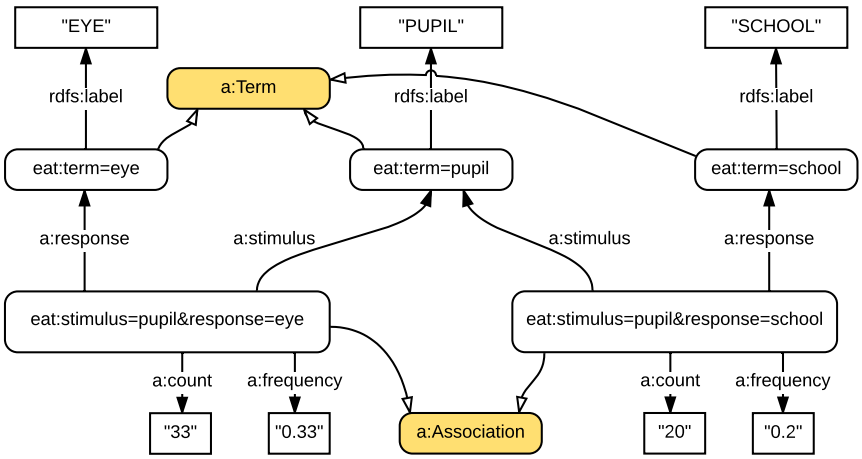


Figure 5.1: Example of the two EAT associations "pupil - school" (right) and "pupil - eye" (left) transformed into RDF. Notice how the stimulus "pupil" has two meanings in the associations: student (right) and part of the eye (left). (Unlabelled edges are `rdf:type`.)

respect to a fixed stimulus s over all responses to that stimulus:

$$f_{s,r} = c_{s,r} / \sum_{r' \in R} c_{s,r'}.$$

As nearly all stimuli were presented to 100 participants, the above nearly always collapses to $f_{s,r} = c_{s,r}/100$, allowing us to interpret the counts $c_{s,r}$ as percentages.

An example for the transformation of the two associations "pupil - eye" and "pupil - school" into RDF can be found in Figure 5.1.

As EAT consists of free-text associations, we modelled each of its terms t as an RDF literal, keeping its capitalisation as found in the original dataset. Further, as RDF does not allow making statements about literals in the subject position, we also minted a URI for each such literal in the domain of the original project's website, for example `eat:term=eye` as shown in Figure 5.1. This will for example allow us to add additional labels (e.g., other capitalisation) to the terms in the future.

term URI

Similar to the terms, we also minted a URI pointing back to the original project's website for each unique association $(s, r) \in \text{EAT}$, for example `eat:stimulus=pupil&response=eye` and linked its corresponding stimulus, response, count and frequency with the properties defined in our freely available association vocabulary `a`: `http://w3id.org/associations/vocab#`, to which (to avoid ambiguity) we will often refer to as `assoc` in the following.

association URI

Further, we assert that each term is an `assoc:Term` and each association an `assoc:Association`.

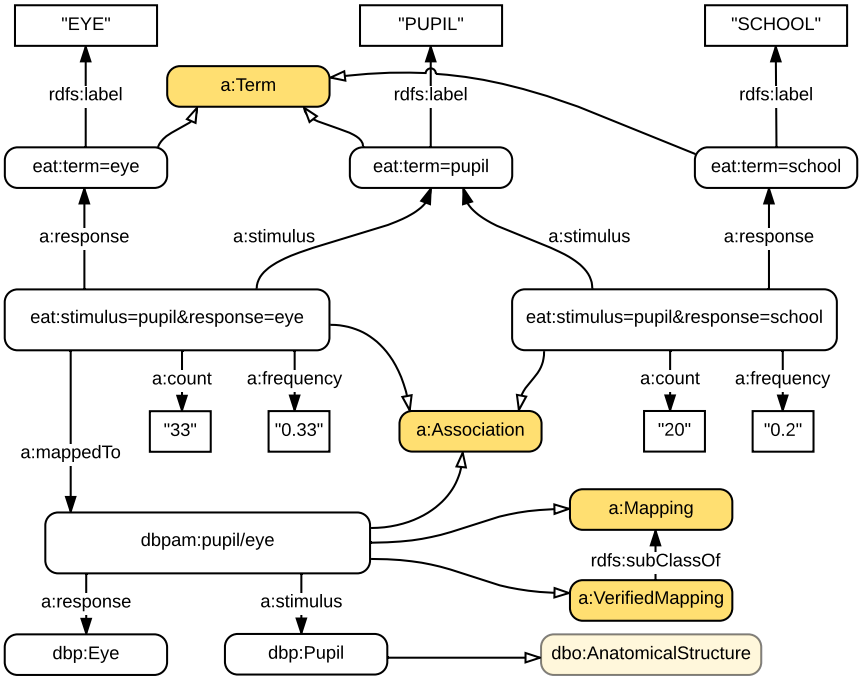


Figure 5.2: Example mapping from EAT to DBpedia for the association "pupil - eye" (left). As `dbr:Pupil` conveys a different meaning than the association "pupil - school" (right), we refrained from creating a wrong semantic association and mapping.

The resulting transformation of EAT into RDF consists of 1 674 376 triples and is provided independently of the following as free download in form of a compressed N-Triples dump.[2].

5.3 MAPPING EAT TO DBPEDIA

After the transformation of the EAT dataset into RDF in the previous section, this section describes the process of mapping associations from EAT to equivalent semantic associations between DBpedia entities.

More precisely, we want to find a mapping of the two terms in each of the EAT associations to two different semantic entities in the DBpedia, such that they convey the same meaning. If we find such two entities, we call the relation between them a *semantic association*.

semantic association

For example, let's focus on the association "pupil - eye", with URI `eat:stimulus=pupil&response=eye` in Figure 5.2. We can identify two DBpedia entities, namely `dbr:Pupil` and `dbr:Eye` with the intended meaning of the association and create a new semantic association

2 http://w3id.org/associations/eat.nt.gz

`dbpam:pupil/eye` with the corresponding links as further detailed in Section 5.3.3.

For the association "pupil - school" however, we find that "pupil" conveys a different meaning than `dbr:Pupil`. Instead of a part of the eye it is used synonymous with student. Hence, in this case we do not create a semantic association that would falsely connect `dbr:Pupil` (the body part) and `dbr:School`.

Meaning of an association

Other negative examples include associations that form composite phrases and describe just one semantic entity. For example, the association "michael - jackson" describes the semantic entity `dbr:Michael_Jackson` instead of two. Thereby it does not fall into our definition of a semantic association, which refers to a pair of distinct semantic entities between which an association relation exists.

For the mapping we will focus on the ~ 5000 unique strong associations occurring more than 20 times (167k raw associations), as they are most undisputed and robust w. r. t. subjectivity, location and time dependency.

In the following, we will first describe some systematic mapping challenges we identified, before detailing our semi-automatic mapping approach and its results.

5.3.1 *Expected Quantities and Identified Challenges*

As the examples above already show, the mapping process is not straightforward for at least some of the associations. In order to estimate what could be expected from a completely manual mapping, which would involve a lot of human work, we decided to randomly sample 50 out of the ~ 5000 unique strong associations and asked two test persons to manually map the stimuli and responses to their corresponding Wikipedia Articles.

The somewhat surprising outcome of this small experiment was that the test persons were only able to manually map 14 of the 50 associations to corresponding semantic associations between DBpedia entities. Out of these 14 the testers reported that 6 could be matched following very simple rules. The remaining 8 required human knowledge and understanding, for example to pick a synonymous term from a list of alternatives on a disambiguation page.

Mapping challenges

Because of this, our expectation for any at least partially automated mapping process is that we will only be able to achieve a successful mapping for about 6/50 to 14/50 (12 - 28 %) of the cases, as even humans cannot do better. While this indicates that DBpedia might not be the best mapping target, it is the primary choice for this work due to its lasting centrality in the LOD Cloud. Also, starting from ~ 167k raw associations, this means that we can still expect to generate a mapping for 20k to 46.8k of them.

We also asked the testers to collect notes about the associations they could not map or had difficulties with. The notes can be summarised as follows (including overlaps):

- **Composite phrases**: In 11/50 cases the association formed a composite phrase (e.g., "identical - twins"), which is just a single semantic entity in DBpedia (not a semantic association).
- **Synonyms**: In 9/50 cases the stimulus and response were synonyms (e.g., "children - kids") leading to the same semantic entity in DBpedia (not a semantic association).
- **Adjectives / Verbs**: In 11/50 cases at least one of the terms was an adjective (e.g., "hot - cold"), in 3/50 one was a verb (e.g., "ring - bell"). As Wikipedia is an encyclopaedia, there is a bias towards substantives, often making it harder to indisputably map adjectives or verbs to a semantic entity in DBpedia.
- **False friends**: In 4/50 cases a simple lookup of the stimulus or response works, but leads to a wrong entity (e.g., "sharpen - knife"[3]).

Further inspection revealed that in 10/50 cases at least one of the terms was a plural word (e.g., "colours - red"), but can mostly be handled without problems via Wikipedia redirect pages.

5.3.2 *Semi-Automatic Mapping Approach*

Using the observations from the manual mapping attempt we developed the semi-automatic mapping approach described in this section. Our approach aims to find high-quality mappings from strong EAT associations to semantic associations between DBpedia entities, while reducing the amount of necessary human work.

In order to achieve this, we use a two-step process: First, we perform an automatic mapping, employing a scoring component which focuses on the identified mapping challenges. Afterwards, we let humans verify the highest scoring mappings with a web application to guarantee high precision of the generated mappings.

5.3.2.1 *Automatic Mapping with Scoring Component*

The automatic mapping uses the Wikipedia API[4] to perform simple searches (following redirect pages) for the stimulus and response in article titles and full texts in order to generate candidate mappings.

The scoring component then assigns scores to these candidate mappings, mostly by trying to identify the potential problems mentioned

3 "Sharpen" describes an Eclipse (IDE) plugin: http://en.wikipedia.org/w/index.php?title=Sharpen&oldid=629433221

4 http://www.mediawiki.org/wiki/API:Main_page

in Section 5.3.1, helping us to focus on the least disputable mapping candidates first:

- **Composite phrases** (e.g., "port - wine"): As a composite phrase is a name for a single semantic entity it is a bad candidate for a semantic association (between two different semantic entities). Hence, if searching for Wikipedia articles (or redirect pages) containing both estimulus and response in their title is successful, the mapping's score receives a strong punishment.
- **Reflexive mappings / synonyms** (e.g., "child - children"): If the mapping of both the stimulus and the response result in the same semantic entity, the score is strongly punished.
- **Adjectives & verbs vs. nouns** (e.g., "unbound - free"): Due to Wikipedia's nature of being an encyclopaedia, adjectives and verbs are under-represented in contrast to nouns. To identify such cases, the stimulus and response are searched in WordNet [52], potentially resulting in multiple synset candidates for each. Mappings containing only synset candidates with the given type "noun" are slightly rewarded. The more synset candidates with types unequal to "noun" are found, the stronger the punishment for the mapping's score.
- **Plural words** (e.g., "thumbs - fingers"): A simple stemming approach is used to compare the stimulus/response to the identified Wikipedia article titles after following redirects. If the match is close to perfect and only differs in singular/plural, the score only receives a slight punishment. Matches with higher edit distances receive a stronger punishment. Perfect matches of stimulus/response with the article title are rewarded.
- **Disambiguation pages** (e.g., "pod - pea"): If the mappings of stimulus or response result in a Wikipedia disambiguation page, the mapping's score receives a strong punishment.

After applying the automatic mapping component with this scoring mechanism to the ~ 5000 strong associations, 1066 promising semantic association candidates (corresponding to ~ 34.2k raw associations) remained for human verification.

5.3.2.2 *Manual Verification*

In order to quickly verify the mapping candidates from the previous section, we developed a small mapping verification web application that shows the textual association from EAT on top (stimulus - response) and both mapped Wikipedia articles below (featuring their abstracts). After a tutorial explaining the purpose, the user is asked if both stimulus and response are correctly mapped to Wikipedia pages.

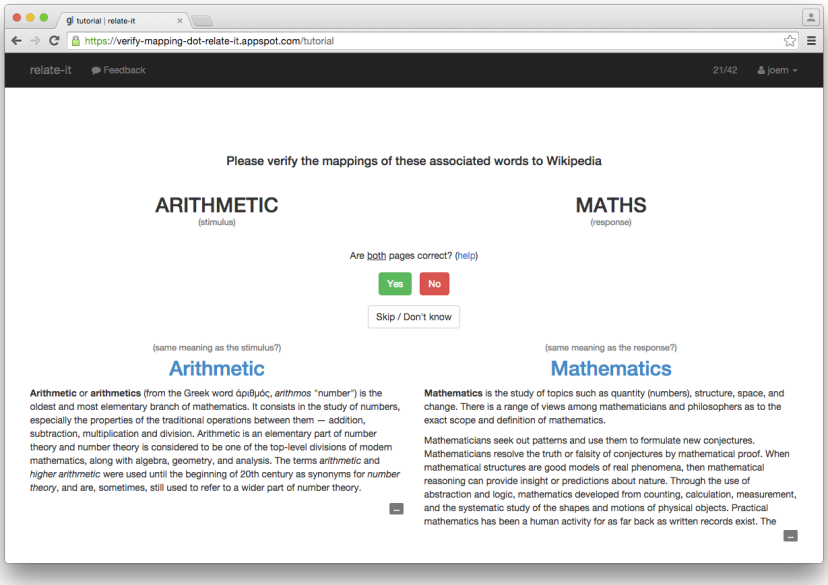


Figure 5.3: DBpedia Mapping Verification Web Application. On top the original EAT association can be seen. Below the two identified corresponding Wikipedia articles and their abstracts are shown.

Possible answers are "Yes", "No" or "Skip / Don't know", as can be seen in Figure 5.3.

The mapping candidates are presented in randomised order and at most once to each user. Candidates that receive a "No" or "Skip / Don't know" rating are immediately excluded from further verifications. After receiving 3 "Yes" ratings from different users, mapping candidates are marked as *valid* and excluded from further evaluation in order to focus efforts on the remaining candidates.

valid mapping

The web application was used by 10 reviewers and quickly allowed the manual verification of the 1066 promising semantic association candidates from the previous step. Out of these, 790 were marked as *valid* by 3 different reviewers. An excerpt of the valid verified mappings can be found in Table 5.2. A list of all 790 verified result mappings and their corresponding Wikipedia article URIs can be found in the appendix in Table A.1.

Results

Notice how multiple different stimulus-response pairs can lead to mappings with the same stimulus and response URI pair. For example, the three associations "casks - beer", "barrels - beer" and "barrel - beer" all lead to the DBpedia stimulus `wiki:Barrel` and response `wiki:Beer`. Out of the 790 mappings, 63 are redundant, leaving 727 that lead to a distinct stimulus-response pair of Wikipedia articles.

Further, we want to present 4 interesting exemplary mapping candidates that our automatic mapping component suggested, but were rejected in the manual review process:

Table 5.2: Verified Semantic Association Mappings Excerpt of 10 stimulus-response pairs, their counts $c_{s,r}$ as defined in Section 5.2 and the corresponding mapped and verified Wikipedia article URIs.

stimulus	response	$c_{s,r}$	stimulus URI	response URI
armour	knight	21	`wiki:Armour`	`wiki:Knight`
bacon	egg	24	`wiki:Bacon`	`wiki:Egg`
barrel	beer	51	`wiki:Barrel`	`wiki:Beer`
barrels	beer	58	`wiki:Barrel`	`wiki:Beer`
casks	beer	23	`wiki:Barrel`	`wiki:Beer`
dentist	teeth	30	`wiki:Dentist`	`wiki:Tooth`
ford	car	35	`wiki:Ford_Motor_Company`	`wiki:Automobile`
pupil	eye	33	`wiki:Pupil`	`wiki:Eye`
pupils	eyes	37	`wiki:Pupil`	`wiki:Eye`
snoring	sleep	29	`wiki:Snoring`	`wiki:Sleep`

1. "pupil - school", mapped to (`wiki:Pupil`, `wiki:School`). As explained above, `wiki:Pupil` refers to the part of the eye, while it is meant as synonym for student in the association.
2. "stocks - shares", mapped to (`wiki:Stocks`, `wiki:Share_(finance)`). The Wikipedia article for stocks refers to wooden medieval punishment devices instead of the `wiki:Stock` in the financial meaning.
3. "germanium - flower", mapped to (`wiki:Germanium`, `wiki:Flower`). Germanium is a chemical element. The EAT participants probably misread it as geranium.
4. "corral - reef", mapped to (`wiki:Pen_(enclosure)`, `wiki:Reef`). Corral is a synonym for pen and enclosure. The EAT participants probably misread it as coral.

The 4 examples illustrate that mappings can be very wrong despite perfect string matches, either caused by ambiguity or even by participants reading a rare textual stimulus wrong as a much more common one.

5.3.3 *Mapping Results and Mapping RDF Dataset*

After manual verification, for each of the 790 verified mapped associations an additional mapping URI was created in the `dbpam` name space (e.g., `dbpam:pupil/eye` in Figure 5.2) and linked from the EAT association with the `assoc:mappedTo` property. The mapping URI is also linked to the DBpedia stimulus and response entities accordingly, as well as typed as an `assoc:Association`, a `assoc:Mapping` and a `assoc:VerifiedMapping`. To generate the DBpedia URIs, we used the fact that they have a 1:1 correspondence to the Wikipedia articles, allowing us to simply replace the `wiki` prefix with `dbr`.

The resulting mapping dataset consisting of 4740 triples can be downloaded[5] or simply dereferenced.

As mentioned before, out of the 790 resulting mappings, 727 lead to a distinct DBpedia stimulus-response pair. In order not to skew our results towards such multiple mappings, we will mostly focus on the distinct 727 stimulus-response pairs of DBpedia entities in the remainder of this work and call them DBpedia associations or simply *semantic associations*.

5.4 CONCLUSION & OUTLOOK

In this chapter, we presented a transformation of 788k free-text associations from the Edinburgh Associative Thesaurus into a RDF dataset, making it easily accessible and mappable to other datasets in the Linked Data community.

Further, we provided a first such mapping to semantic associations between DBpedia entities. After identifying systematic mapping challenges such as composite phrases, synonyms or perfect string matches leading to false friends, we developed a semi-automatic mapping approach. Our approach allows us to quickly find good candidates for indisputable, high precision mappings, that can be manually verified by a web application with little effort. Applying our approach to the EAT dataset resulted in 790 manually verified mappings corresponding to ~ 25.5k raw associations, leading us to 727 semantic associations between distinct DBpedia entity pairs. All generated datasets are publicly available[6].

As mentioned in Section 3.1, in this work we focus on the Edinburgh Associative Thesaurus [101] (EAT) dataset. However, in future, we look forward to apply our semi-automatic mapping approach to map further association datasets, such as the University of South Florida free association, rhyme, and word fragment norms [131, 132] (USFA) to DBpedia or other Wikipedia related RDF datasets such as Wikidata [174].

5 http://w3id.org/associations/mapping_eat_dbpedia.nt.gz
6 https://w3id.org/associations

6 DATASET GENERATION RESULTS

After the description of our three approaches to collect ground truth datasets of semantic associations in the previous chapters, we will in this chapter provide a comparison of the approaches (Section 6.1), a decision which collected dataset forms the semantic association ground truth for the remainder of this work (Section 6.2) and a first analysis of the semantic associations in DBpedia in (Section 6.3). Afterwards, we will conclude Part II in Section 6.4.

6.1 COMPARISON OF DATASET GENERATION APPROACHES

The three approaches presented in Chapter 4 and Chapter 5 consist of the two GWAPs *BetterRelations* and the *Knowledge Test Game*, as well as a semi-automatic mapping approach that we used to map EAT to DBpedia.

The approaches differ in the kind of data collected, the amount of recorded human actions backing the aggregated results and the type of aggregation results. An overview of the differences can be found in Table 6.1.

Table 6.1: Comparison of Dataset Generation Approaches

Approach	Raw Data	Result Type	Aggregated Result
Better-Relations	4.7k	Triples per topic ranked by association strengths	12 completed topics (183 ranked triples)
Knowledge Test Game	6.9k	Ranked semantic associations per topic	62 distinct semantic associations (14 primary)
EAT mapped to DBpedia	25.5k	Primary word associations mapped to semantic associations	727 distinct primary semantic associations

BetterRelations focuses on existing triples for a given topic (stimulus) and tries to score them by human association strengths. The approach aims at a high-quality scoring of each individual list of triples, deriving the scores from pairwise comparisons. Overall 4.7k matching comparisons could be collected, resulting in 12 ordered topic lists with a total of 183 ranked triples by association strengths. Sadly, the game suffered from high amounts of noise, and the number of topic lists (and thereby different topics), which the game determined to be fully ordered, is very limited.

BetterRelations

Knowledge Test Game

The *Knowledge Test Game* tries to overcome the issues with noise and biasing towards existing triples by allowing free-text input and immediately disambiguating the entered response strings to response DBpedia entities. It allowed us to collect 6.9k semantic raw associations from players. After aggregation of all submissions (independent of their order) from games that were played by at least 20 players, 62 associations (between 24 distinct stimuli and 54 distinct responses) were mentioned by at least 10 players in total. Only two of these associations ((`dbr:Mark_Zuckerberg`, `dbr:Facebook`) and (`dbr:YouTube`, `dbr:Video`)) also were mentioned by at least 20% of all corresponding players. When only focusing on the first guesses (primary associations) of each player, we are left with 19 primary semantic associations (between 14 distinct stimuli and 18 distinct responses) mentioned by at least 10 players. Out of those, 14 primary semantic associations were further mentioned as first submission by at least 20% of the players.

semi-automatic mapping approach

In contrast to our games, which despite being fun still require a lot of human effort, our *semi-automatic mapping approach* allows us to benefit from the huge amount of work that has already been put into existing psychological association collections such as the EAT dataset. After manual review, the approach allowed us to quickly collect 727 distinct semantic associations between DBpedia entities, corresponding to ~ 25.5k primary word associations. Unlike our games, which were designed to validate all collected user input and hence properly rank associations (responses) for relatively few topics (stimuli), the semantic association mappings based on the EAT dataset are not only backed by orders of magnitude more humans, but they also lead to a much larger variety of different stimuli: In total, the 727 distinct semantic associations are composed of 685 distinct stimulus and 346 distinct response nodes.

Table 6.2: Semantic Association Ground Truth Excerpt

stimulus entity	response entity	train	test
`dbr:Armour`	`dbr:Knight`	✓	
`dbr:Bacon`	`dbr:Egg`	✓	
`dbr:Barrel`	`dbr:Beer`	✓	
`dbr:Beach`	`dbr:Sand`	✓	
`dbr:Dentist`	`dbr:Tooth`		✓
`dbr:Ford_Motor_Company`	`dbr:Automobile`	✓	
`dbr:Pupil`	`dbr:Eye`	✓	
`dbr:Puppy`	`dbr:Dog`	✓	
`dbr:Snoring`	`dbr:Sleep`	✓	

6.2 SEMANTIC ASSOCIATION GROUND TRUTH

After the comparison of approaches in the previous section, we see that the semi-automatic mapping approach was by far the most successful in collecting a large variety of semantic associations. Also, unlike the gaming approaches it does neither introduce a strong bias towards only existing facts nor rely on closed source external search engines for the live disambiguation of response string inputs to response entities.

Hence, for the remainder of this work, we decided to focus on the 727 distinct semantic associations generated by our semi-automatic mapping approach of EAT's strong primary word associations to DBpedia entity pairs. We will also call this the *ground truth* $\mathcal{GT}$ list of semantic associations. For the development of our algorithm in Part III we also performed a 9:1 random training-test set split: All development and training was performed on the training set of 655 ground truth pairs. An excerpt of the ground truth list can be seen in Table 6.2, the full list in Table B.1.

ground truth $\mathcal{GT}$

6.3 FIRST ANALYSIS OF SEMANTIC ASSOCIATIONS IN DBPEDIA

Before concluding Part II in the following section, we want to provide a first analysis of the generated semantic association ground truth dataset in this section. We start with statistics about the involved DBpedia entities, before performing an analysis of the distances and linkage patterns in DBpedia.

Out of the 727 unique semantic associations in DBpedia, there are 685 distinct stimulus and 346 distinct response nodes, totalling in 955 distinct nodes. None of the stimuli occur more than twice, but some of the responses occur more frequently, such as `dbr:Money` or `dbr:Bird`, as can be seen in Table 6.3.

Table 6.3: Most Frequent Response Nodes

Response	Count	Response	Count
`dbr:Money`	19	`dbr:Water`	9
`dbr:Bird`	15	`dbr:Army`	8
`dbr:Horse`	14	`dbr:Beer`	8
`dbr:Automobile`	13	`dbr:Death`	7
`dbr:Flower`	12	`dbr:Fish`	7
`dbr:Music`	12	`dbr:Bed`	7
`dbr:Tree`	11	`dbr:Ship`	7
`dbr:Sea`	11	`dbr:Red`	6
`dbr:Dog`	9	`dbr:Gun`	6
`dbr:Food`	9	`dbr:Hair`	6

core dataset

extended dataset

To analyse the node degrees, distances and linkage patterns in DBpedia, we used a local Virtuoso 7.2[1] mirror of the DBpedia 2015-04[2] core and extended datasets. The *core dataset*[3] includes the ~ 412M triples which are loaded on the public DBpedia *SPARQL endpoint*[4]. Additionally, we loaded all further available datasets for the English DBpedia[5] including the nearly 159M Wikilinks (`dbo:wikiPageWikiLink`), to which we refer as the *extended dataset*.

Table 6.4: Top-20 degrees of the 955 investigated association nodes in the core (left) and extended (right) datasets.

Node	Degree	Node	Degree
`dbr:Animal`	237855	`dbr:Animal`	445324
`dbr:Insect`	118589	`dbr:Village`	344264
`dbr:France`	94826	`dbr:Insect`	239032
`dbr:India`	85386	`dbr:France`	234700
`dbr:Plant`	79062	`dbr:India`	196686
`dbr:Italy`	55966	`dbr:Plant`	149369
`dbr:Village`	54082	`dbr:Italy`	143942
`dbr:Beetle`	43739	`dbr:Town`	85994
`dbr:Scotland`	27607	`dbr:Beetle`	83109
`dbr:Bird`	25933	`dbr:Scotland`	73312
`dbr:Switzerland`	19874	`dbr:Paris`	66504
`dbr:City`	18030	`dbr:Switzerland`	61214
`dbr:Paris`	17362	`dbr:City`	53008
`dbr:Wales`	14605	`dbr:Bird`	50332
`dbr:Town`	13301	`dbr:Ireland`	40592
`dbr:Ireland`	11340	`dbr:Marriage`	38643
`dbr:Rome`	10344	`dbr:Rome`	38611
`dbr:Fly`	10299	`dbr:Wales`	38532
`dbr:Mayor`	9812	`dbr:School`	32824
`dbr:Reptile`	9595	`dbr:Novel`	32193

Degrees

In order to analyse the differences between the core and extended dataset, we first computed the degrees of all DBpedia association nodes. As expected, the node degrees in the extended dataset are much larger than the ones in the core dataset (avg. ~ 4650 extended vs. ~ 1240), as can be seen in Table 6.4. Nevertheless, we can observe that even without Wikilinks, some of the nodes, such as `dbr:Animal`, `dbr:Insect`, `dbr:France` have a very high degree. Investigations re-

1 https://github.com/openlink/virtuoso-opensource
2 http://wiki.dbpedia.org/dbpedia-data-set-2015-04
3 http://downloads.dbpedia.org/2015-04/core/
4 http://dbpedia.org/sparql
5 http://downloads.dbpedia.org/2015-04/core-i18n/en/

vealed that such high node degrees are mostly originating from incoming edges such as dbo:kingdom, dbo:class, dbo:country, dbo:type, dbo:order, or dbo:birthPlace in the core dataset. In the extended dataset, they unsurprisingly mostly originate from incoming edges of the property dbo:wikiPageWikiLink, but also from gold:hypernym from the Linked Hypernym Datasets [103].

Paths

Next, we analysed the paths up to a length[6] of 2 between stimulus and response of the DBpedia associations. In the core dataset, only 34 (< 5%) of the 727 DBpedia associations are directly connected (24 forward, 12 backward, 2 bi-directionally) and still only 417 (57.4%) via another node (path of length 2)[7]. In contrast to this, in the extended dataset 547 (75.2%) of the 727 DBpedia associations are directly connected (445 forward, 413 backward and even 311 bi-directionally) and 726 (99.9%) via another node (path of length 2).

Connecting properties

For paths of length 1, we also analysed which properties frequently link the stimulus and response nodes. In the core dataset for the 34 associations, these properties are mostly rdfs:seeAlso, dbo:class, dbprop:classis, dbo:kingdom, dbo:country, and dbo:ingredient (unidirectional). In the extended dataset for the 547 associations, we additionally find many dbo:wikiPageWikiLink and gold:hypernym edges.

Wikilinks

Furthermore, in more than 60% an existing Wikilink connecting a DBpedia association is bi-directional, in contrast to globally only ~ 7% of all Wikilinks. This could be a good signal and indicator for a semantic association from the dataset of Wikilinks, which is otherwise often discarded as difficult to use due to its size and weak semantics.

Connecting properties

Connecting nodes

Finally, we also analysed the properties and connecting nodes for paths of length 2. In the core dataset, the majority of connecting properties consists of dcterms:subject, rdf:type, rdfs:seeAlso, dbo:product, and dbo:class. Connecting nodes are unsurprisingly owl:Thing, but also umbel:EukaryoticCell, umbel:BiologicalLivingObject, umbel:Bird, umbel:Animal, and dbr:Category:Plant_morphology. In the extended set, the connecting properties are again led by dbo:wikiPageWikiLink and gold:hypernym, followed by dbprop:wikiPageUsesTemplate and the ones from the core dataset. The connecting nodes are additionally led by nodes such as dbr:Template:Reflist, dbo:Article, but also dbr:QI_(L_series), dbr:List_of_Latin_words_with_English_derivatives, and dbr:Bird.

Wikilinks

Again, we can see that a lot of information seems to be hidden in the dbo:wikiPageWikiLinks. For example, despite their weak semantics, they allow us to filter for many common words by connecting to

6 Length is here defined as the number of triples.

7 This is actually surprising, as one would expect every entity to be an owl:Thing in DBpedia and hence to always find an undirected path of length 2 of the form: `?source a owl:Thing. ?target a owl:Thing`. In DBpedia 2015-04 this was however not always the case, e. g., there were no triples `?s rdf:type owl:Thing` for `?s = dbr:Snoring` or `?s = dbr:City`. The bug was reported and fixed in later releases.

nodes such as dbr:QI_(L_series)[8] and suddenly become a good indicator for a potential semantic association between DBpedia entities in the extended dataset. In the core dataset, such weak signals seem to be lost.

6.4 CONCLUSION

With this, we conclude Part II. In chapters 4 and 5, we presented three approaches to collect semantic associations and compared them in Section 6.1. The resulting ground truth dataset of 727 semantic associations is described in Section 6.2 and made available online[9] (amongst others) as RDF.

By size, desired quality and future availability the dataset fulfils the first goal of this work (cf. Section 1.2) and forms our semantic association ground truth dataset used for our graph pattern learning algorithm in Part III.

Further, we already presented the results of a first analysis of the distances and linkage patterns of semantic associations in DBpedia in Section 6.3. We note significant differences between the DBpedia core and extended datasets, mainly with respect to the Wikipedia page links (dbo:wikiPageWikiLink). As we will for example see in Section 10.4.4, our graph pattern learner will indeed make use of the information hidden behind Wikipedia page links.

8 QI is a TV game show featuring many common words.
9 https://w3id.org/associations

Part III

PATTERN LEARNING FROM LINKED DATA

7 LEARNING APPROACH INTRODUCTION

7.1 LEARNING APPROACH OVERVIEW

In this part, we present a machine learning algorithm which was developed to make use of the generated semantic association ground truth dataset described in Part II and can learn patterns to simulate human associations. Parts of this chapter have already been published in [80] and [82].

Input

A high level overview diagram of our algorithm (called the "Graph Pattern Learner") can be seen in Figure 7.1. The inputs for our algorithm's training phase are a training list of source-target node-pairs and a SPARQL endpoint. In our primary use-case, this list of source-target pairs consists of the semantic association ground truth described in Section 6.2.

Pattern learner & fusion component

From the inputs, our graph pattern learner forms a trained model. Given a new source node, the model can predict target nodes in a way that resembles the relations behind the given training list of source-target node-pairs. The full algorithm consists of two main components, the pattern learner and fusion component. The first component is an evolutionary algorithm that learns an ensemble of graph patterns for the given list of source-target pairs, as described in Chapter 8. Afterwards, in Chapter 9, we will explain how the learned patterns can be used for prediction of target candidates given a new source node, and how the fusion component is trained and used to fuse all candidates into a single ranked list.

Following the presentation of our algorithm, we present several evaluations of our algorithm in Chapter 10. We will conclude this part with Chapter 11, in which we show how our algorithm is applicable to other scenarios than our primary use-case of semantic associations.

However, before jumping into the details of our algorithm, we use the remainder of this chapter to mention the high-level design goals (Section 7.2) and introduce basics and definitions (Section 7.3).

7.2 DESIGN GOALS

The main goal of our machine learning approach is the simulation of human associations with Linked Data. Influenced by our first analysis of the semantic association ground truth in Section 6.3, we define the following design goals for our algorithm:

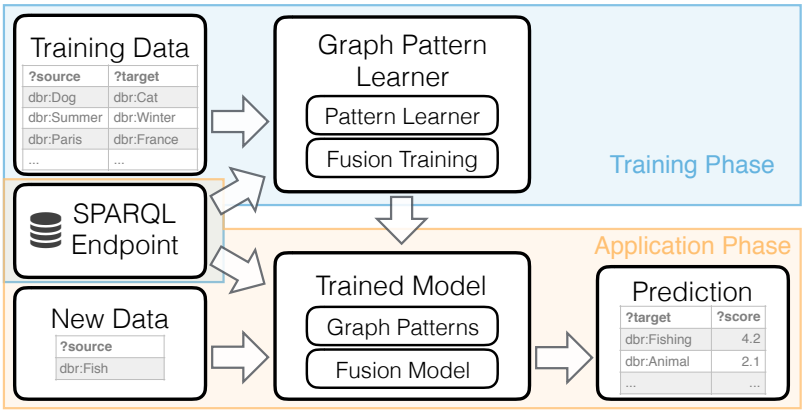


Figure 7.1: Graph Pattern Learner System Overview

- **Direct learning from a SPARQL endpoint**

  Linked Data is available in a large variety of formats. Accessing it via a *SPARQL endpoint* using the standardised SPARQL protocol allows us to directly evaluate SPARQL queries in a future proof and interoperable way. Furthermore, it allows us to benefit from many years of optimisation efforts that has been put into the development of state of the art endpoints.

- **No assumptions about modelling**

  When expressing knowledge in RDF, a lot of design decisions are made, e. g., about which vocabularies to use or the directionality of properties. We do not want our algorithm to be tailored towards just one way of modelling, for example by only using certain types of relations or only following them in a forward fashion. Instead, our algorithm should be more generic and able to follow relations in a forward and backward manner.

- **Scalable**

  As we use DBpedia as the mapping target for our semantic associations, our algorithm should be able to deal with at least 1G triple being loaded on the endpoint and the resulting large search spaces, mainly originating from the observed high node degrees and large direct neighbourhoods.

- **Efficiency**

  As we expect to perform many SPARQL queries during training, the algorithm should be able to efficiently cancel runaway queries with timeouts and limits. It is also desirable to batch and parallelise many of the computations.

- **Noise & Failure Resistance**

  When performing millions of queries on current SPARQL endpoints, the chance of encountering errors (e. g., due to bugs,

timeouts, congestion, dropped requests) becomes non-negligible. Furthermore, endpoints such as Virtuoso provide (non-deterministic) partial results when running into timeouts. Our algorithm should be able to deal with such noise and uncertainty.

- **End-to-end learning**

  As the first analysis of our ground truth dataset already showed, there is no single already existing property in DBpedia that perfectly models semantic associations. Rather than relying on manual selection (or exclusion) of such properties or other manual feature engineering, we would like our algorithm to learn useful features on its own in an end-to-end fashion. This means that, given a training list of source-target node-pairs, the algorithm should be designed to learn suitable features and from them construct high-quality predictors without the need for further human intervention.

- **Ensemble Learning**

  Further, as we do not expect the relations behind the training list of source-target node-pairs to be reducible to a single feature (e. g., in the case of semantic associations there seems to be a relation that connects capitals to their countries and another that connects hyponyms to hypernyms), we would like our algorithm to be able to find an ensemble of features instead of just one dominant feature.

- **Explainable**

  Last but not least, we would like the feature representation of our algorithm to be explainable to humans. As our algorithm tries to learn features from a large knowledge graph, we would like the feature representation to be graph patterns in form of simple SPARQL queries (more precisely SPARQL Basic Graph Pattern [75, 156] (BGP)).

7.3 BASICS AND DEFINITIONS

After listing our design goals, we now want to introduce some basic definitions for the following chapters.

Formally, our goal is to develop a graph pattern learning algorithm that can help to identify SPARQL queries for a *relation* $\mathcal{R}$ between node pairs $(s, t) \in \mathcal{R}$ in a given *knowledge graph* G[1], where s is a source node and t a target node. $\mathcal{R}$ can for example be a simple relation such as "given a capital s return its country t" (called $\mathcal{R}_{cc}$ in the following), or a complex one such as in our main use-case "given a stimulus s return a response t that a human would associate" ($\mathcal{R}_{ha}$).

relation $\mathcal{R}$
knowledge graph G
source node s
target node t

1 For our purpose G is a set of RDF triples, typically accessible via a given SPARQL endpoint.

ground truth $\mathcal{GT}$

To learn queries for $\mathcal{R}$ from G without any prior knowledge about the modelling of $\mathcal{R}$ in G, we allow users to compile a *ground truth set*[2] of training source-target pairs $\mathcal{GT} \subseteq \mathcal{R}$ as input for our algorithm. For example, for relation $\mathcal{R}_{cc}$ between capital cities and their countries, the user could generate a ground truth set $\mathcal{GT} = \{$(`dbr:Berlin`, `dbr:Germany`), (`dbr:Paris`, `dbr:France`), (`dbr:Oslo`, `dbr:Norway`)$\}$. Given $\mathcal{GT}$ and the DBpedia SPARQL endpoint[3], our graph pattern learner could then learn a set of graph patterns such as:

gp_1 :
```
?source dbo:country ?target.
```

gp_2 :
```
?target dbo:capital ?source. ?target a dbo:Country.
```

graph pattern gp

Refining the terminology from Section 1.3.4, we define a *graph pattern* $gp \in GP$, where GP is the infinite set of SPARQL Basic Graph Patterns [75, 156] (BGPs). We can model GP as the power-set of triples over terms (*URIs* U, *BNodes* B and *Literals* L) in G and *Variables* V:

URIs U
BNodes B
Literals L
Variables V

$$GP = \mathcal{P}((U \cup B \cup V) \times (U \cup V) \times (U \cup B \cup L \cup V)) \setminus \{\}$$

$gpl(\mathcal{GT}, G)$

Our task is to find a finite subset of "good" (as detailed below) patterns $gpl(\mathcal{GT}, G) \subset GP$.

ASK, SELECT

As each gp is a BGP, we can easily form corresponding SPARQL queries from it, such as *ASK* and *SELECT queries*, as already mentioned in Section 1.3.4 and Section 2.1.1.2. We denote their execution against G as $ASK(gp, G)$ or $SELECT(gp, G)$. For simplicity, we will omit the static G in the following, and simply write $ASK(gp)$ and $SELECT(gp)$. The graph patterns can contain SPARQL variables,

?source *?target*

out of which we reserve *?source* and *?target* as special variables for source nodes s and target nodes t. For example, the query corresponding to $SELECT(gp_1)$ for gp_1 from above would look like this:

```
SELECT DISTINCT * WHERE {
  ?source dbo:country ?target.
}
```

Its execution generates a (potentially empty) set μ of result bindings with instantiations of `?source` and `?target`. For example, in the above case, (`dbr:London`, `dbr:United_Kingdom`) $\in \mu$.

A mapping ϕ can be used to (partially) bind variables in gp before execution, which we typically realise by inserting a corresponding `VALUES` clause into the SPARQL query. Using such a binding, a pattern

prediction

gp can be used to *predict* target candidates by selecting `?target` after binding a source node s, which we will denote as:

$$\text{prediction}_{gp}(s) = \underset{?target}{SELECT}(\phi_{?source:=s}(gp))$$

For example, by binding `?source` to `dbr:London` the query corresponding to pattern gp_1 looks as follows:

2 We usually refer to this set as "list", as its order is used by our algorithm to perform a reproducible pseudo-random split of the training and test set with a static seed.

3 http://dbpedia.org/sparql

```
SELECT DISTINCT ?target WHERE {
  VALUES (?source) { (dbr:London) }
  ?source dbo:country ?target.
}
```

and on execution predicts the target candidate `dbr:United_Kingdom` $\in$ prediction$_{gp_1}$(`dbr:London`).

To simply check if a graph pattern gp *fulfils* a source-target pair (s_i, t_i), we can also execute the corresponding SPARQL ASK query:

gp *fulfils* (s_i, t_i)

$$\text{ASK}(\phi_{\texttt{?source}:=s_i,\texttt{?target}:=t_i}(gp))$$

which looks as follows for gp_1 and the source-target pair (`dbr:London`, `dbr:United_Kingdom`):

```
ASK {
  VALUES (?source ?target) {
    (dbr:London dbr:United_Kingdom)
  }
  ?source dbo:country ?target.
}
```

We also say that the graph pattern gp *covers* or *models* (s_i, t_i) if the corresponding ASK query returns true.

covers
models

7.3.1 *Good Patterns*

Given these basics, intuitively, a good pattern should maximise the coverage of source-target pairs, while at the same time minimise the amount of (false positive) target candidates. It should also try to minimise complexity of the pattern w. r. t. query size and query time. A graphical representation of this intuition can be found in Figure 7.2.

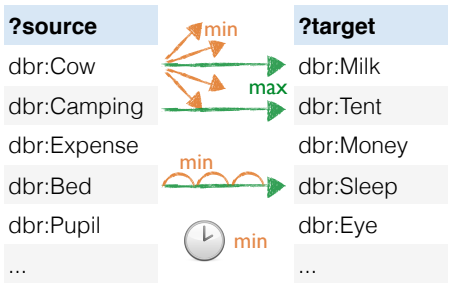


Figure 7.2: Conceptual Visualisation of a Good Pattern

The following definitions help us to formalise the above intuition of a good pattern and will ultimately lead to the definition of our fitness function in Section 8.3:

recall

- High *recall*:
  A good pattern gp fulfils as many of the given ground truth pairs $(s_i, t_i) \in \mathcal{GT}$ as possible:

$$\text{gt matches}_{gp} = \left| \left\{ (s_i, t_i) \in \mathcal{GT} \middle| \text{ASK} \left(\underset{\substack{?source:=s_i \\ ?target:=t_i}}{\phi} (gp) \right) \right\} \right|$$

$$\text{recall}_{gp} = \frac{\text{gt matches}_{gp}}{|\mathcal{GT}|}$$

precision

- High *precision*:
  A good pattern should also be precise. For every individual ground truth pair $(s_i, t_i) \in \mathcal{GT}$ we define the precision as:

$$\text{precision}_{gp}((s_i, t_i)) = \begin{cases} \frac{1}{|\text{prediction}_{gp}(s_i)|} & \text{if } t_i \in \text{prediction}_{gp}(s_i) \\ 0 & \text{otherwise} \end{cases}$$

  The expected target t_i should be in the list of predicted target candidates and if possible nothing else; we do not look for patterns returning many potentially wrong targets for a given source.

  Over all ground truth pairs, we can define the precision for gp as the inverse of the average result lengths:

$$\text{avg result len}_{gp} = \underset{(s_i,t_i)\in\mathcal{GT}}{\text{avg}} |\text{prediction}_{gp}(s_i)|$$

$$\text{precision}_{gp} = \begin{cases} \frac{1}{\text{avg result len}_{gp}}, \text{if avg result len}_{gp} > 0 \\ 0, \text{otherwise} \end{cases}$$

gain

- High *gain*:
  As mentioned above, one of the design goals of our algorithm is to learn an ensemble of graph patterns. Given a set of other patterns GP_e in the ensemble, a pattern $gp \notin GP_e$ is better (w. r. t. information gain) if it covers those ground truth pairs $gtp \in \mathcal{GT}$ that aren't covered with high precisions by other patterns of the ensemble $gp_e \in GP_e$ already:

$$\text{gain}_{GP_e,gp} = \sum_{gtp\in\mathcal{GT}} \max \left\{ 0, \text{precision}_{gp}(gtp) - \max_{gp_e\in GP_e} \text{precision}_{gp_e}(gtp) \right\}$$

  Similarly, given an ensemble of graph patterns, the potentially remaining gain can be computed as:

$$\text{remains}_{GP_e} = \sum_{gtp\in\mathcal{GT}} \left(1 - \max_{gp_e\in GP_e} \text{precision}_{gp_e}(gtp) \right)$$

- No *over-fitting*: *over-fitting*
  While precision is to be maximised, a good pattern should not over-fit to a single source or target from the training input.

- Short *pattern length* and low *variable count*: *pattern length* *variable count*
  Between two patterns which perform similarly otherwise, the one being more concise should be favoured.

  The pattern length of a pattern is defined as the number of triples in the pattern. The variable count of a pattern is the number of distinct variables in the pattern.

  However, we consider this a low priority dimension: A good pattern is not necessarily restricted to a shortest path between `?source` and `?target`. For example, good patterns can have additional edges off the connecting path, such as gp_3:

  gp_3 : `?target ?p ?source. ?target a dbo:Country.`

  gp_3 is an example of a pattern of length 2 with variable count 3. While the first part of the pattern `?target ?p ?source` describes a shortest path between `?source` and `?target`, the pattern might perform much better in terms of precision with the additional restriction `?target a dbo:Country`.

- Low *execution time* & *timeout*: *execution time* *timeout* *time*
  Last but not least, to have any practical relevance, good patterns should be executable in a short *time*. Especially during the training phase, many graph patterns might be encountered that cause excessively long query-times. We need to make sure to terminate such queries early and efficiently on both sides: the graph pattern learner and the endpoint (cf. Section 8.8). In case the query representing the pattern was aborted due to a *timeout* and none or only a partial result was obtained, the pattern is not considered practically applicable.

7.3.2 *Search Space Complexity*

Before describing our machine learning approach to find an ensemble of good patterns, it is helpful to briefly think about the search space of the problem.

Using the results of our first analysis in Section 6.3, the observed high average node degrees are alarming. With average node degrees in the order of 1k (in the core dataset, or 4k in the extended), we very quickly face combinatorial explosion. We can estimate that with 2 steps in our graph, we reach about 1M nodes, with 3 about 1G. The latter would already exceed the amount of nodes contained in DBpedia (and all other datasets loaded on our local endpoint, described *neighbourhood*

in Section 10.2). While this estimate makes several simplifying assumptions, such as the node degree distributions staying constant after each hop, it is not unrealistic that the longest shortest path between any two nodes in our training set is very short, even without relying on RDF specific super-nodes such as `owl:Thing`.

Patterns based on knowledge base

However, the above estimate only looks at the connectivity of existing triples in the knowledge base and not at how many graph patterns can be formed from them. For a knowledge base G with $|G|$ triples, a simplistic upper-boundary for this is the number of subgraphs, so the power-set $\mathcal{P}(G)$ of size $2^{|G|}$: for each triple we can decide whether to use it in a graph pattern or not. For knowledge bases in the order of 1G triples, this means an upper boundary of $2^{1\,000\,000\,000} \approx 10^{300\,000\,000}$ possible graph patterns, which are fully instantiated. This simplistic upper boundary however neither takes into account that we usually search for connected graph patterns, nor that each term could also be replaced with a variable.

Patterns based on neighbourhood

To form a closer upper bound, we can look at the number of connected graph patterns of length n that we can draw from the neighbourhoods of our ground truth pairs. Even under the simplifying assumption that all triples are drawn from the 1-neighbourhood of a center node with 1k connections, we have $\binom{1000}{n}$ possibilities. For $n \in \{1, 2, 3, 4, 5\}$ this leaves us with 1k, 500k, 166M, 41G and 8T patterns. Despite drastic simplification, we can see that full enumeration based on the given instantiations, even of relatively short patterns, is infeasible.

Variable-only patterns

As graph patterns can also include variables, we can look at the search space of our problem from another angle, which allows us to abstract away from combinatorial explosions arising from the instantiation in our given knowledge base. For this, we can focus on graph patterns only consisting of variables. As the amount of variables is infinite, it follows that there are infinitely many graph patterns only consisting of variables.

However, to be connected, a pattern with n triples can contain at most $2n+1$ distinct variables[4]: 3 in the first triple and 2 in any further triple (the remaining variable connecting the triple to a variable already appearing in one of the previous triples). In the following, we will without loss of generality restrict the amount of variables and their names to the finite set of $V_{2n+1} = \{?v_1, \ldots, ?v_{2n+1}\} \subset V$.

Simple enumeration

In an attempt to enumerate all possible variable-only patterns of length n, we can first generate all possible triples T over the $2n+1$ variables: $T = V_{2n+1}^3$. Finally, to generate a pattern, we can draw n out of these triples arriving at $\binom{|T|}{n} = \binom{(2n+1)^3}{n}$ combinations. For $n \in \{1, 2, 3, 4\}$ this leads to 27, 7750, 6.7M and 11.6G different candidates

4 Note that this is a necessary but not sufficient condition. For example, the pattern `?v1 ?v2 ?v3. ?v3 ?v2 ?v1. ?v4 ?v5 ?v6.` has $6 < 2 \cdot 3 + 1 = 7$ variables.

of variable-only graph patterns. An example for a pattern of length 3 looks like this:

$$gp_4 : \quad \texttt{?v1 ?v2 ?v3. ?v1 ?v4 ?v5. ?v3 ?v4 ?v5.}$$

However, these pattern candidates will also include many atypical patterns and triples for RDF knowledge bases:

- Triples forming *self loops*, for example: *self loops*

  $$\texttt{?v1 ?v2 ?v1}$$

  Patterns with self loops contain a triple with subject and object being the same:

  $$\exists i, (s_i, p_i, o_i) \in gp : s_i = o_i$$

- Patterns that are *disconnected*, for example: *disconnected*

  $$\texttt{?v1 ?v2 ?v3. ?v3 ?v2 ?v1. ?v4 ?v5 ?v6.}$$

  A pattern is *connected* if an ordering of its triples $t_i \in gp, 1 \leqslant i \leqslant n$ exists, such that for each but the first triple at least one of the variables or terms x in it occurs in a previous triple already: *connected*

  $$\forall j \exists i, 1 \leqslant i < j \leqslant n : \exists x : t_i \ni x \in t_j$$

  For a connected pattern such an ordering can for example be generated with a breadth first enumeration of its triples.

  Disconnected patterns are rarely helpful for prediction and put a lot of stress on the SPARQL endpoint: they cause a Cartesian product between the results for each disjoint component. For example, when selecting `?source` and `?target` from the following disconnected pattern: *Disconnected patterns cause problems*

  $$\texttt{?source ?v2 ?v3. ?v3 ?v2 ?source. ?v4 ?v5 ?target.}$$

  the endpoint has to combine all results for `?source` in:

  $$\texttt{?source ?v2 ?v3. ?v3 ?v2 ?source.}$$

  with those for `?target` in:

  $$\texttt{?v4 ?v5 ?target.}$$

  In such cases, it is more efficient and reliable to learn the two (incomplete) patterns independently.

- Patterns that are *edge-only connected*, for example: *edge-only connected*

  $$\texttt{?v1 ?v2 ?v3. ?v4 ?v2 ?v5.}$$

node connected

A pattern is edge-only connected, if it is connected (as above), but not node connected: We call a pattern *node connected*, if it is connected via its subjects and objects:

$$\forall j \exists i, 1 \leqslant i < j \leqslant n, (s_i, p_i, o_i) \in gp, (s_j, p_j, o_j) \in gp : \exists x : \{s_i, o_i\} \ni x \in \{s_j, o_j\}$$

Similar to disconnected patterns, edge-only connected patterns are rarely helpful for prediction. After instantiation, the pattern above could for example look like this:

```
?source rdf:type dbo:Animal. ?target rdf:type dbo:Town.
```

and only be connected via their edges `rdf:type`. In contrast, the following pattern is also node connected:

```
?source rdf:type dbo:Animal. ?target rdf:type dbo:Animal.
```

node-edge joint

- Patterns that are *node-edge joint*, for example:

  ```
  ?v1 ?v2 ?v3. ?v2 ?v4 ?v5.
  ```

  A pattern is node-edge joint, if any of its predicate variables or terms also appears in subject or object position:

  $$\exists i \exists j : (s_i, p_i, o_i) \in gp, (s_j, p_j, o_j) \in gp : p_i \in \{s_j, o_j\}$$

  From the atypical patterns, this is the most useful one, as it can be used to model complex relations or even simulate `rdfs:subPropertyOf` reasoning:

  ```
  ?source ?v2 ?target. ?v2 rdfs:subPropertyOf ?v5.
  ```

  However, in most cases, due to materialisation and reasoning support of endpoints, we can rely on simpler, more direct (inferred) patterns, such as:

  ```
  ?source ?v5 ?target
  ```

Further, the generated pattern candidates will include many patterns that are structurally similar, but have different syntactical string representations. For example, compare the two patterns gp_4 and gp_5:

gp_4 : `?v1 ?v2 ?v3. ?v1 ?v4 ?v3. ?v3 ?v4 ?v5.`

gp_5 : `?v1 ?v4 ?v5. ?v1 ?v2 ?v5. ?v5 ?v2 ?v3.`

isomorphic graph patterns

By swapping the names for `?v2` with `?v4` and `?v3` with `?v5` we can convert the string representation of gp_4 to the one of gp_5. We call such structurally similar patterns *isomorphic*, as will be explained in more depth in Section 7.3.3.

Excluding the above atypical patterns from our enumeration of all variable patterns of length n, inserting `?source` and `?target` as two

special variables (that are distinguishable) and afterwards only leaving one pattern per isomorphism class, for $n \in \{1, 2, 3, 4\}$, we are left with 2, 28, 486 and 10k structurally different and relevant patterns for our use-case. Currently, the enumeration of such patterns for $n > 4$ seems to be computationally infeasible.

Enumeration of pattern isomorphism classes

With this, we have found a lower bound for the structurally different and relevant variable only graph patterns which we would like our algorithm to be able to find. The actual number of interesting graph patterns is however drastically increased by potential instantiations of each of the variables from the given knowledge base.

7.3.3 *Canonical Form of Isomorphic Graph Patterns*

As briefly mentioned above, graph patterns with different string representations can have the same meaning for our purpose. We call such graph patterns isomorphic. Besides for enumeration, we are strongly interested in finding a canonical representation for all isomorphic patterns in order to avoid unnecessary computations. For example, a canonical representation of patterns allows us to efficiently cache query results, as will be described in Section 8.8.5.

A simplistic example of two isomorphic patterns with different string representations are the following gp_6 and gp_7, which only differ in the ordering of their triples:

gp_6 : `?source ?p ?target. ?target a dbo:Town.`

gp_7 : `?target a dbo:Town. ?source ?p ?target.`

As BGPs (and thereby our graph patterns) typically use set semantics, we can avoid this simplistic problem by ordering the triples of graph patterns alphanumerically in our string representation.

However, as variable names (except for `?source` and `?target`) can be chosen arbitrarily, simply sorting their triples does not help us to identify that the two patterns gp_8 and gp_9 have the same meaning for our purposes:

gp_8 : `?source ?v1 ?target. ?source ?v2 ?v3.`

gp_9 : `?source ?v2 ?target. ?source ?v1 ?v3.`

In general, we can define pattern gp_i to be homomorph to gp_j if there is a *homomorphism* function h such that $h(gp_i) = gp_j$. In our case, h is a function that provides a mapping between variable names of gp_i to those of gp_j, excluding `?source` and `?target`. Formally, we can define $\text{vars}(gp)$ as the set of all variables except for `?source` and `?target` in gp:

homomorphism

$$\text{vars}(gp) = (\{v \mid v \in t \in gp\} \cap V) \setminus \{\texttt{?source}, \texttt{?target}\}$$

$h : \text{vars}(gp_i) \to \text{vars}(gp_j)$ then is a mapping of the variable names from gp_i to those of gp_j. We denote the replacement of the variables in a graph pattern as $h(gp)$.

isomorphism

If $\exists h : h(gp_i) = gp_j$ and h is bijective ($h^{-1}(gp_j) = gp_i$), we further call it an *isomorphism* and $gp_i \cong gp_j$ isomorphic. As $\cong$ forms an equivalence relation we also say that gp_i and gp_j are in the same equivalence or *isomorphism class*.

isomorphism class

canonical form of a pattern $\text{canon}(gp)$

With the notion of an isomorphism class, we can define the *canonical form of a pattern* $\text{canon}(gp)$ as the representative of its isomorphism class. We select this representative $\text{canon}(gp)$ to be the alphanumerically smallest pattern in the isomorphism class of gp, after sorting the triples of each pattern:

$$\text{canon}(gp) = \min_{\forall gp_i : gp_i \cong gp} \text{sorted}(\{t \in gp_i\})$$

Generating canonical form

One simple approach to find this canonical form of a graph pattern $\text{canon}(gp)$ is: change its variable names to $?v_1, ?v_2, \ldots, ?v_k$, generate all permutations of these new variable names and keep the alphanumerically smallest of the arising patterns after sorting each of their triples. The approach is however not very efficient, as it is in $\mathcal{O}(k!)$.

Complexity

In general, graph isomorphism and graph canonicalisation are challenging problems in NP, but proving more precise bounds of their complexities is still the subject of active research [10, 11, 121]. Graph isomorphism is known to be in NP, but it is unknown whether it is also in P or NP-complete [121] (in contrast to subgraph isomorphism, which is known to be NP-complete [41]). Graph canonicalisation is known to be NP-hard and at least as difficult as graph isomorphism, but it is unknown if they are polynomial time equivalent [8, 12].

Reduction to RDF graph canonicalisation

To still efficiently generate a canonical form for BGPs, we can reduce the problem to the more common RDF graph isomorphism and BNode canonicalisation problem. Two RDF graphs are isomorphic if a bijective mapping between their BNodes can be found, which transforms one graph into the other. A canonical graph representation, assigning each BNode a unique string representation, can for example be computed with the RDF Graph Digest Algorithm 1 (RGDA1) [120], which achieves good practical run-times.

GP canonicalisation approach

Our reduction approach works as shown in Listing 7.1: For a given BGP gp we generate a graph g. For each triple $t \in gp$, we generate a new reification BNode `triple_bnode` in g, that we connect with its s, p and o with `rdf:subject`, `rdf:predicate` and `rdf:object` edges. In the process, we also replace all variables with corresponding BNodes, except for `?source` and `?target` which are replaced with special URIs. We then use the RGDA1 algorithm on g to replace all BNodes with their canonical representations, resulting in a canonicalised graph cg. From this we re-extract each `triple_bnode` and reconstruct the corresponding triple via the used reification. We also convert the special URIs back to `?source` and `?target` and all encountered BNodes back into variables using their now canonicalised string representations as variable names. After sorting the returned triples alphanumerically, we are left with the canonical graph pattern $cgp = canon(gp)$.

Listing 7.1: SPARQL BGP Canonicalisation via RDF Graph Canonicalisation with RGDA1

```
def sparql_bgp_canonicalisation(gp):
    g = Graph()
    for t in gp:
        triple_bnode = BNode()
        triple = []
        for i in t:
            if isinstance(i, Variable):
                if i in {SOURCE_VAR, TARGET_VAR}:
                    triple.append(URIRef(PREFIX + i))
                else:
                    triple.append(BNode(i))
            else:
                triple.append(i)
        s, p, o = triple
        g.add((triple_bnode, RDF['type'], RDF['Statement']))
        g.add((triple_bnode, RDF['subject'], s))
        g.add((triple_bnode, RDF['predicate'], p))
        g.add((triple_bnode, RDF['object'], o))
    cg = RGDA1.to_canonical_graph(g)
    cgp = sorted(extract_vars_from_bnodes(cg))
    return cgp
```

The generated canonical form of a graph pattern allows us to reduce isomorphism checks between two patterns to a simple string comparison. It also allows us to efficiently store (and compare) many patterns in hash-based data-structures, for example for the enumeration of structurally different patterns in the previous Section 7.3.2 or for caching purposes, as we will see in Section 8.8.5. Due to these beneficial properties, we will in general store patterns in their canonical form.

8 PATTERN LEARNING ALGORITHM

8.1 EVOLUTIONARY ALGORITHM OVERVIEW

After introducing the design goals and basics in Chapter 7, in this chapter, we will present the core of our pattern learning approach. Parts of this chapter have already been published in [80].

The algorithm is designed to solve the aforementioned learning problem: Given a ground truth list of source-target pairs $\mathcal{GT}$ and a knowledge base G in form of a SPARQL endpoint, our graph pattern learner shall learn a set of "good" graph patterns $\text{gpl}(\mathcal{GT}, G) \subset GP$, with GP the set of SPARQL BGPs. As mentioned before, the primary ground truth dataset is formed by the semantic associations described in Chapter 6.

$\text{gpl}(\mathcal{GT}, G)$

As common in machine learning, our goal is to inductively learn a model from the ground truth examples during training. The model should be able to replicate the shown behaviour and after training be able to transfer it to new inputs, even such which have never been seen during training. Applied to our primary use-case, this means that during training our algorithm should find a model consisting of patterns for semantic associations, which can after training be used to simulate human associations.

Before detailing the individual components of our approach, we give a brief overview. The outline of our graph pattern learner is similar to the generic outline of evolutionary algorithms. It consists of *individuals*, which are evaluated to calculate their *fitness*. In our case, the individuals are graph patterns $gp_i \in GP$ (BGPs) with at least a `?source` and `?target` variable. Their fitness is evaluated against a given SPARQL endpoint by performing a series of queries. In all brevity, patterns are the fitter, the more ground truth source-target pairs they cover with high precision and low query evaluation times.

individuals, fitness

The fitter an individual is, the higher is its chance to survive and reach the next *generation* (often also called an evolution step). Together, we also refer to all individuals of a generation as a *population*.

generation
population

In each generation, there is a chance for its individuals to *mate* by exchanging triples, and *mutate* by introducing, deleting or flipping triples, as well as replacing variables and entities from the SPARQL endpoint with each other. A population can contain the same individual (graph pattern) several times, causing fitter individuals to have a higher chance to mate and mutate over several generations.

mate
mutate

As the given ground truth list of source-target pairs is unlikely to be modelled with a single pattern in the given knowledge base, our

runs

algorithm performs several *runs*. In each run, it re-focuses on the parts of the ground truth training list that are not already covered well yet. This allows our algorithm to reach good overall coverage over the whole ground truth, independent of the modelling in the knowledge base.

Furthermore, in order to reduce the computation times, our algorithm uses techniques, such as query timeouts as a proxy for complexity, batching and caching.

In the following, we will walk through the main building blocks of our algorithm: Runs & coverage (Section 8.2), fitness & evaluation (Section 8.3), mating (Section 8.4), mutation (Section 8.5), initial population (Section 8.6), and selection of the next generation (Section 8.7). We will also address practical issues (Section 8.8) and introduce an interactive visualisation we developed to be able to observe each step of the learning process (Section 8.9), before we will use the resulting graph patterns for prediction in Chapter 9.

We implemented our graph pattern learner with help of the Distributed Evolutionary Algorithms in Python (DEAP) framework [61]. The full source-code, interactive visualisation and other complementary material can also be found on our website[1].

8.2 RUNS & COVERAGE

As mentioned before, our algorithm is not limited to learn a single best pattern for a list of ground truth pairs, but it can learn an ensemble of multiple patterns which together cover the list, as explained in Section 7.3.

We realise this by an adaptive fitness function (as defined in the next section) and invoking our evolutionary algorithm in several *runs*[2]. In each run, a full evolutionary algorithm is executed (with all its generations). After each run, the resulting best patterns are added to a global list of results. In the following runs, all ground truth pairs which are already covered by the patterns from previous runs become less rewarding for a newly learnt pattern to cover (cf. the gain dimension of the fitness in the following section). Over its runs our algorithm will thereby re-focus on the left-overs, which allows us to maximise the coverage of all ground truth pairs with good graph patterns.

runs

For the re-focusing in run r we will rely on the $\text{gain}_{\text{run}_{r-1},gp}$ and $\text{remains}_{\text{run}_{r-1}}$ defined in Section 7.3.1, with run_{r-1} consisting of all patterns discovered in all previous runs: $\text{run}_{r-1} = \bigcup_{0<q<r} \text{GP}_q$.

1 https://w3id.org/associations

2 We decided against picking the name "epoch", as it is already frequently used in other machine learning algorithms and could lead to the impression that a single run only sees all training data once, which would be wrong: In our algorithm, each individual fitness evaluation sees all training data.

8.3 FITNESS & EVALUATION

An important aspect of every evolutionary algorithm is the representation of its individuals and their *fitness*. As already mentioned before, the individuals of our algorithm are SPARQL BGPs and an individual $gp \in GP$ is in general represented internally by its canonical form (cf. Section 7.3.3). *fitness*

The goal of our algorithm is to find graph patterns that are good graph patterns, incorporating the various dimensions of "good" described in Section 7.3.1. In order to capture these considerations in an evaluable order, we define the fitness of an individual graph pattern gp as a 10-tuple of real numbers with the following optimisation directions. When comparing the fitness of two patterns, the fitness tuples are (currently) compared lexicographically. This means that the earlier dimensions have a higher priority than the later ones.

1. **Remains** (max): Remaining precision sum $\text{remains}_{\text{run}_{r-1}}$ in the current run r (see Section 8.2). Patterns found in earlier runs are considered better. This component is a constant from the view-point of the evolutionary algorithm in a single run for now, which allows us to later easily re-construct the learning order and to normalise the gain. In the first run, this is equal to the length of the training ground truth pair list. In later runs, remains is reduced by the sum of max precisions for each training pair.

2. **Score** (max): A derived attribute multiplying the gain attribute with a configurable punishment p_o (default: 0.25) for over-fitting patterns and a multiplicative punishment p_t for timeouts (see 9. Timeout): $\text{score} = \text{gain} \cdot p_o \cdot p_t$. For most good patterns this means that score = gain, but a pattern only matching a single ground truth pair for example, will be punished by score = gain · 0.25 by default.

3. **Gain** (max): The summed gained precision over the remains of the current run r: $\text{gain}_{\text{run}_{r-1},gp}$.

4. **F_1-measure** (max): F_1-measure for precision_{gp} and recall_{gp} of this pattern. In case of incomplete patterns (lacking `?source` or `?target`), this is set to 0. In case of timeouts, the resulting value is multiplied with the timeout punishment term (see 9. Timeout).

5. **Average Result Lengths** (min): avg result len_{gp}.

6. **Recall (Ground Truth Matches)** (max): gt matches_{gp}.

7. **Pattern Length** (min): The number of triples $|\{t \mid t \in gp\}|$ this pattern contains.

8. **Variables Count** (min): The number of distinct variables this pattern contains: $|\{i \mid i \in t \in gp\} \cap V|$.

9. **Timeout** (min): 0 if no timeout occurred, 0.5 in case of a soft timeout (a result was generated only in between 75% and 100% of the specified maximal query time) and 1 in case of a hard timeout (the result was not generated within the maximal query time). The timeout punishment term p_t that is used in score and F_1-measure is: $1 - \text{timeout}$. Also see Section 8.8.

10. **Query Time** (min): The evaluation time in seconds. As the last dimension, this attribute is mostly effective indirectly via the timeout attribute and its p_t in score. The raw query time however is informative as it hints at the real complexity of the pattern for the endpoint. A pattern may have a small number of triples and variables, but its evaluation could involve a large portion of the dataset.

8.4 MATING

In each generation there is a configurable chance for two patterns to mate in order to exchange information. In our algorithm this is implemented in a common way for evolutionary algorithms: Mating always creates two children, replacing their two parents, keeping the population size stable.

For one child, one parent is the dominant and the other the recessive parent. For the other child it is vice versa.

Each child will contain all triples occurring in the intersection of both parents. Additionally, there is a high chance p_d (typically 80 %) to select each of the remaining triples from the *dominant parent* and a low chance p_r (20 %) to select each of the remaining triples from the *recessive parent*.

dominant parent

recessive parent

The probabilities $p_d + p_r$ are chosen to add up to 1, meaning that the children have the same expected length as their parents. Thereby, the expected length of the patterns in any generation is not affected by the mating process. If $p_d + p_r \neq 1$, we would either encounter exponential growth (> 1) or exponential decrease (< 1) of the pattern lengths over the generations by the mating process.

Balanced probabilities

Furthermore, as variables from the recessive parent could accidentally match variables already in the child, and this can be beneficial or not, we add a 50 % chance to rename such variables before adding the triples.

8.5 MUTATION

Besides mating, which exchanges information between two individuals, information can also be gained by mutation. Each individual

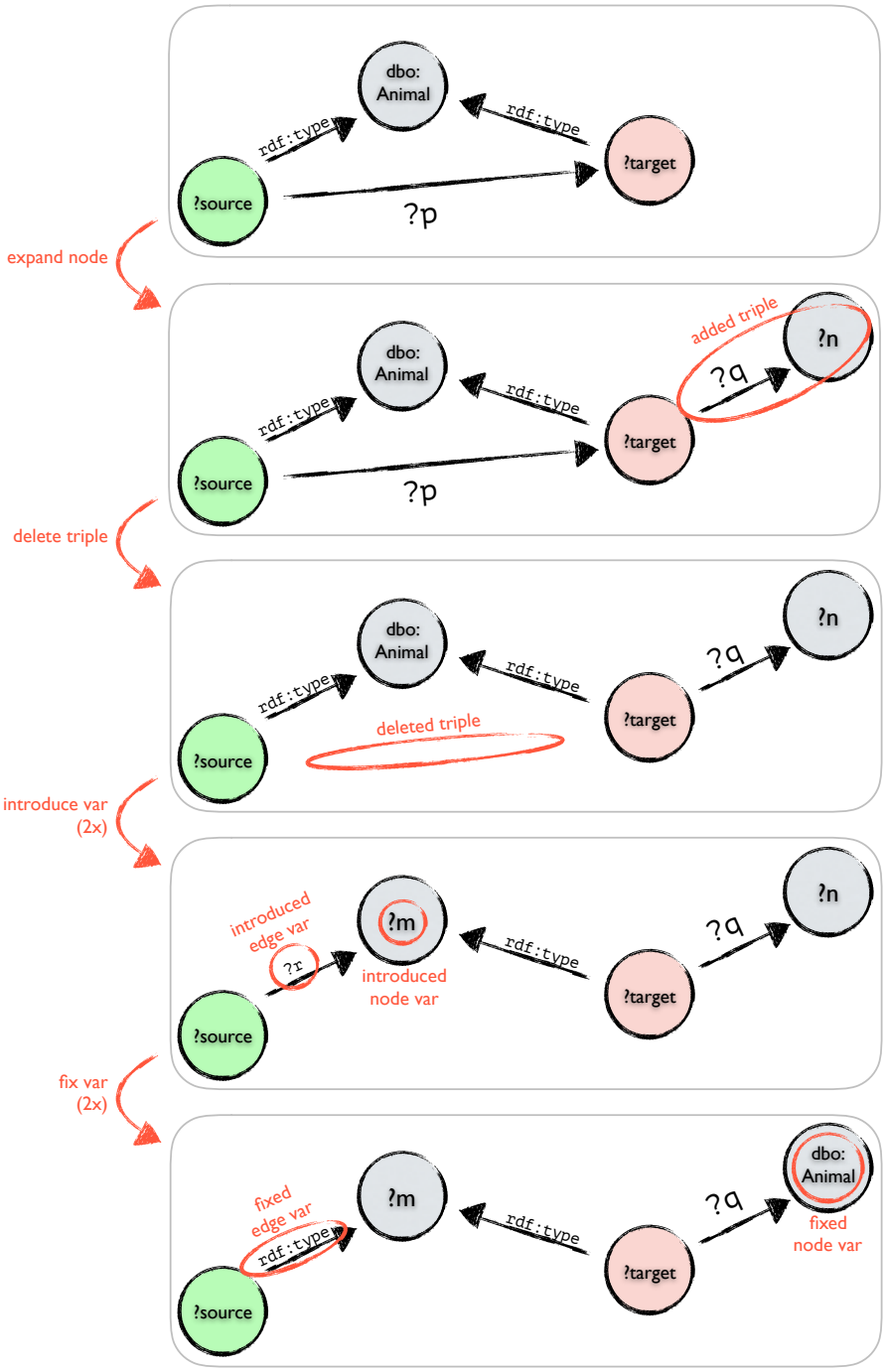


Figure 8.1: Mutation Examples

in a population has a configurable chance to mutate by the following (non exclusive) mutation strategies. A visualisation of the most fundamental mutations can be found in Figure 8.1. Currently, all but one of the mutation operations can be performed on the pattern itself (local) without issuing additional SPARQL queries (except for the re-evaluation of the pattern in the next generation). The mutation operations also have different effects on the pattern itself (grow, shrink pattern size) and on its result size (harden, loosen pattern constraints).

- **expand node**: select a node, and add a triple with the node and two new variables (grow, harden) (local)
- **add edge**: select two nodes, add an edge with a new variable in between (grow, harden) (local)
- **delete triple**: delete a triple statement (shrink, loosen) (local)
- **introduce var**: select a component (node or edge) and convert it into a variable (loosen) (local)
- **split var**: select a variable and randomly split it into two variables (grow, loosen) (local)
- **merge var**: select two variables and merge them (shrink, harden) (local)
- **increase dist**: increase distance between source and target by moving them one a hop further apart (grow) (local)
- **simplify pattern**: simplify the pattern, deleting unnecessary triples (shrink) (local) (cf. Section 8.8.7)
- **fix var**: select a variable and instantiate it with an IRI, BNode or Literal from the knowledge base, that can take its place (harden) (SPARQL) (see below)

In a single generation, sequential mutation (by different strategies in the order as above) is possible.

We can generally say that introducing a variable loosens a pattern and fixing a variable hardens it. Patterns which are too loose will generate a lot of candidates and take a long time to evaluate. Patterns which are too hard will generate too few solutions, if any at all. Very big patterns, even though very specific can also exceed reasonable query and evaluation times.

Given only the expand node, add edge, delete triple and merge var mutations, it is already possible to generate all possible isomorphism classes of variable-only patterns (cf. Section 7.3.3). With help of the fix var mutation, the algorithm can further generate all possible partial instantiations of these patterns from the given knowledge base.

8.5.0.1 *Fix Var Mutation*

Unlike the other mutations, the fix var mutation is the only one which makes use of the underlying dataset via the SPARQL endpoint G, in order to instantiate variables with an IRI, BNode or Literal. As it is one of the most important mutations and also because performing SPARQL queries is expensive, it is implemented to immediately return several mutated children.

For a given pattern gp we randomly select one of its variables `?v` (excluding `?source` and `?target`). Additionally, we sample up to a defined number of source-target pairs from the ground truth training list which are not well covered yet (high potential gain). For each of these sampled pairs (s, t), we issue a SPARQL Select query of the form:

```
SELECT DISTINCT ?v {
  VALUES (?source ?target) { (s, t) }
  ...gp...
}
```

Instantiation of a variable

We collect the possible instantiations for `?v`, and count them over all queries. Afterwards, we randomly select (with probabilities according to their frequencies) up to a configurable number of them. Each of the selected instantiations forms a separate child by replacing `?v` in the current pattern. While this temporarily grows the population, the population will be controlled to remain within the desired limits, as will be explained in Section 8.7.

8.6 INITIAL POPULATION

In order to start any evolutionary algorithm, an initial population needs to be generated. The main objective of the first population is to form a starting point from which the whole search space is reachable via mutations and mating over the generations. While the initial population is not meant to immediately solve the whole problem, a poorly chosen initial population results in a lot of wasted computation time.

Initial population: path patterns

For prediction capabilities, we are searching graph patterns which connect `?source` and `?target`. Hence, our algorithm fills the initial population (consisting of 200 individuals by default) with patterns of varying length l forming a simple path between `?source` and `?target`.

Length $l \in \mathbb{N}_0$ is drawn randomly according to a beta distribution between 0 and the configurable maximum pattern length (default: 15) with configurable α and β parameters. As longer patterns are less desirable, the default values for $\alpha = 5$ and $\beta = 30$ are selected in a way that cause 48% of the initially generated patterns to have a length $l \leqslant 1$, 84% a length $l \leqslant 2$ and 99.9% a length $l \leqslant 5$.

A special case are paths of length $l = 0$. In this case, the pattern consists of a single triple containing only either `?source` or a `?target` variable. These patterns are of one of the four following forms:

```
?source ?p1 ?v1.
```

```
?v1 ?p1 ?source.
```

```
?target ?p1 ?v1.
```

```
?v1 ?p1 ?target.
```

While having a low chance of survival (direct evaluation would typically yield bad fitness), such patterns can re-combine (see mating in Section 8.4) with other patterns to form good and complete patterns in later generations.

For $l > 0$, such a path pattern solely consists of variables and is initially directed from source to target:

`?source` $?p_1$ $?n_1$ . ... $?n_i$ $?p_{i+1}$ $?n_{i+1}$ . ... $?n_{l-1}$ $?p_l$ `?target.`

For example, a pattern of desired length of $l = 3$ looks like this:

```
?source ?p1 ?n1. ?n1 ?p2 ?n2. ?n2 ?p3 ?target.
```

After the patterns are generated, we randomly flip each of their edges with a 50% chance, to explore edges in any direction.

In order to reduce the high complexity and noise introduced by patterns only consisting of variables, we immediately subject them to the fix variable mutation (cf. Section 8.5) with a high chance.

8.7 NEXT GENERATION & POPULATION CONTROL

After each generation, the next generation is formed by the surviving individuals from n tournaments of k randomly sampled individuals (with repetition) from the previous generation (defaults: $n = 200$ and $k = 3$). For each tournament of k individuals, the fittest is the one with the lexicographically highest ranked fitness vector (see Section 8.3). Only the individual with the highest fitness from each of the n tournaments is allowed to proceed to the next generation.

Selection tournament

Population control

Additionally, we employ two techniques to counter population degeneration (e. g., all patterns becoming too complex) and make our algorithm more robust (even against non-optimal parameters):

- In each generation, we re-introduce a small number (by default 5%) of newly generated initial population patterns (see Section 8.6).
- Each generation updates a hall of fame, which will preserve the best (by default 100) patterns ever encountered over the generations. In each generation a small fraction (by default 5%) of the best of these all-time best patterns is re-introduced.

8.8 PRACTICAL CONSIDERATIONS

In the following, we will briefly discuss practical challenges that we encountered during the development of our approach and the solutions and optimisation techniques we used to overcome them.

8.8.1 *Batching*

The most important optimisation of our algorithm lies in the reduction of the amount of queries issued by making use of batch queries. This mostly applies to the queries for fitness evaluation. As mentioned in Section 8.3, to evaluate the fitness of one graph pattern we need the *ground truth matches* and *avg result length*. A straight forward implementation of this would perform $2 * |\mathcal{GT}|$ queries (one *ASK* and one *COUNT query* for each of the training pairs) to evaluate the fitness of a single individual. However, for our primary use-case, with an estimated 100ms SPARQL query response time, this wastes more than 2 minutes per evaluation mostly with unnecessary connection overhead.

Connection overhead

It is a lot more efficient to run several sub-queries in one big query and to only transport the training ground truth pairs to the endpoint once (via a `VALUES` clause) than to ask for each result separately. For example, we can combine the ASK and COUNT queries for several hundred batched training pairs like this:

Batching with `VALUES`

```
SELECT ?source ?target ?ask ?count WHERE {
 VALUES (?source ?target) {
  (dbr:Berlin dbr:Germany)
  (dbr:Amnesia dbr:Memory)
  (dbr:Paris dbr:France)
  (dbr:Rome dbr:Egypt)
  ... long list ...
 }
 BIND(EXISTS{
    ?source dbo:wikiPageWikiLink ?target .
    ?source a dbo:PopulatedPlace .
    ?target a schema:Country .
 } AS ?ask)
 OPTIONAL {
  {
   SELECT ?source COUNT(DISTINCT ?target) as ?count WHERE {
    ?source dbo:wikiPageWikiLink ?target .
    ?source a dbo:PopulatedPlace .
    ?target a schema:Country .
   }
  }
 }
}
```

The main cost of such an approach is a drastic increase of the client side implementation complexity for re-assembling the results and proper error-recursion, as a timeout or error in a single query might otherwise quickly cause incorrect results for another query.

Also note that there are query size limits on most SPARQL endpoints. To reduce query size, we hence strip unnecessary white-space and shorten the URIs by using prefixes where possible.

8.8.2 *Limits and Timeouts as a Proxy for Complexity*

Another mandatory optimisation involves the use of limits and timeouts for all queries, even if they usually only return very few results in a short time. We found that a few run-away queries can quickly lead to congestion of the whole endpoint and block much simpler queries.

Timeouts are also especially useful as a reliable proxy to exclude too complicated graph patterns. Even seemingly simple patterns can take a very long time to evaluate depending on the underlying dataset and its distribution. `?source a ?v1 . ?target a ?v1` is an example of such a pattern. On most RDF knowledge bases, the pattern will cause a denial of service on the endpoint by requiring it to iterate over all contained nodes that are an `owl:Thing`.

8.8.3 *Fit To Live Filter*

Apart from timeouts we use a filter which checks if mutants and children are actually desirable, meaning fit to live, even before evaluating them. If not, the respective parent takes their place in the new population, allowing for a much larger part of the population to be viable and a lot of SPARQL endpoint resources freed up for meaningful queries.

Pattern constraints

The filter asserts that a pattern does not exceed a certain maximum size (amount of variables, triples as well as bytes), that it has at least a `?source` or `?target` variable and that it does not contain very long literals. Additionally, it asserts that each of the patterns is connected, as mutations and mating can create disconnected patterns with two or more components. Such disconnected patterns lead to especially stressing SPARQL queries, as their disconnected nature asks the SPARQL endpoint to create the Cartesian product over all solutions of each disconnected component.

8.8.4 *Parallelisation*

Evolutionary algorithms are parallelisable via parallel evaluation of all individuals, but in our case the SPARQL endpoint quickly becomes the bottleneck. Ignoring the limits of the queried endpoint will

resemble a denial of service attack. For most of our experiments we hence use an internal Linked Data mirror with exclusive access for our learning algorithm, as described in Section 10.2. In case the algorithm is run against public endpoints, we suggest to only use a single thread in order to not disturb their service (fair use).

8.8.5 *Caching*

Client side caching further helps to reduce the time spent on evaluating graph patterns, by only evaluating them once, even if the same pattern is generated by different sequences of mutation and mating operations. To identify equivalent patterns despite different syntactic surface forms, we had to solve the SPARQL BGP canonicalisation problem (finding a canonical labelling for variables). We were able to achieve this by reducing the problem to RDF graph canonicalisation, as mentioned in Section 7.3.3, which allows us to apply the RDF Graph Digest Algorithm 1 (RGDA1) [120] with good practical runtime.

BGP canonicalisation

8.8.6 *Non-Deterministic SPARQL Results*

In the context of caching, one other important finding is that many SPARQL endpoints (especially the widely used OpenLink Virtuoso) often return incomplete and thereby non-deterministic results by default. Unlike many other search algorithms, an evolutionary algorithm has the benefit that it can cope well with such non-determinism. Hence, when caching is used, it is helpful to reduce, but not completely remove redundant queries.

8.8.7 *Pattern Simplification*

Last but not least, through several generations of mutations and matings, our algorithm can create patterns that are unnecessarily complex. Several forms of this are shown in Figure 8.2. A pattern can for example contain redundant parallel variable edges, edges between fixed nodes (IRIs), edges behind fixed nodes and completely unrestricting leaf branches. While many of these redundancies are helpful during exploration (e. g., an unrestricted leaf can in the next generation be subjected to a fix var mutation that instantiates the leaf), having too many of these unnecessarily complex patterns in a population or returning them as results, makes our algorithm less efficient.

Remove redundancies

Hence, we introduced a so called simplify pattern mutation (cf. Section 8.5) and always simplify the result patterns of a full run: Given a complicated graph pattern gp_c with one or multiple of the above flaws, our pattern simplification algorithm finds a smaller equivalent

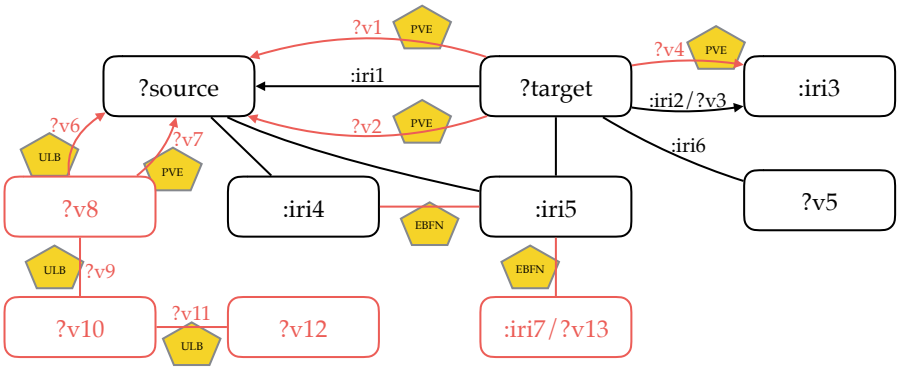


Figure 8.2: Shown is a complex graph pattern gp_c. Unnecessary edges and nodes are coloured in red. The black part forms the simplified graph pattern gp_s. SPARQL syntax is used for node and edge labels. Unlabelled edges can be picked to either be a URI or Variable. Where not denoted with an arrow head, directionality of the edge is arbitrary.
Possible **simplifications** are annotated with yellow background: **PVE**: parallel variable edge, **ULB**: unrestricting leaf branch, **EBFN**: edge between/behind fixed nodes (IRIs and Literals).

pattern gp_s with the same result set w. r. t. the `?source` and `?target` variables by removing unnecessary triples.

8.9 VISUALISATION

After presenting the main components of our evolutionary algorithm in the previous sections, we will now briefly present an interactive visualisation[3]. As the learning of our evolutionary algorithm can produce many graph patterns, the visualisation allows to quickly get an overview of the resulting patterns in different stages of the algorithm. By this, the visualisation not only allowed us to find meaningful defaults for the multitude of parameters and probabilities (e. g., for mating and mutations), but also to quickly identify problems during the development of our algorithm.

Figure 8.3 (top) shows a screen shot of the visualisation of a single learned graph pattern. In the sidebar (Figure 8.3 (bottom left)), the user can select between individual generations, the results of a whole run or the overall results (default) to inspect the outcomes at various stages of the algorithm. After a stage is selected for analysis, the user can for example select an individual graph pattern. Below these selection options the user can inspect statistics about the selected graph pattern including its fitness (Figure 8.3 (bottom middle)), a list of matching training ground truth pairs, and the corresponding SPARQL *SELECT query* of the pattern (Figure 8.3 (bottom

3 Interactively available at `https://w3id.org/associations`.

right)). Links are provided to perform live queries on the SPARQL endpoint.

At each of the stages, the user can also get an overview of the precision coverage of a single pattern (as can be seen in Figure 8.4 (top)) or the accumulated coverage over all patterns (Figure 8.4 (bottom)).

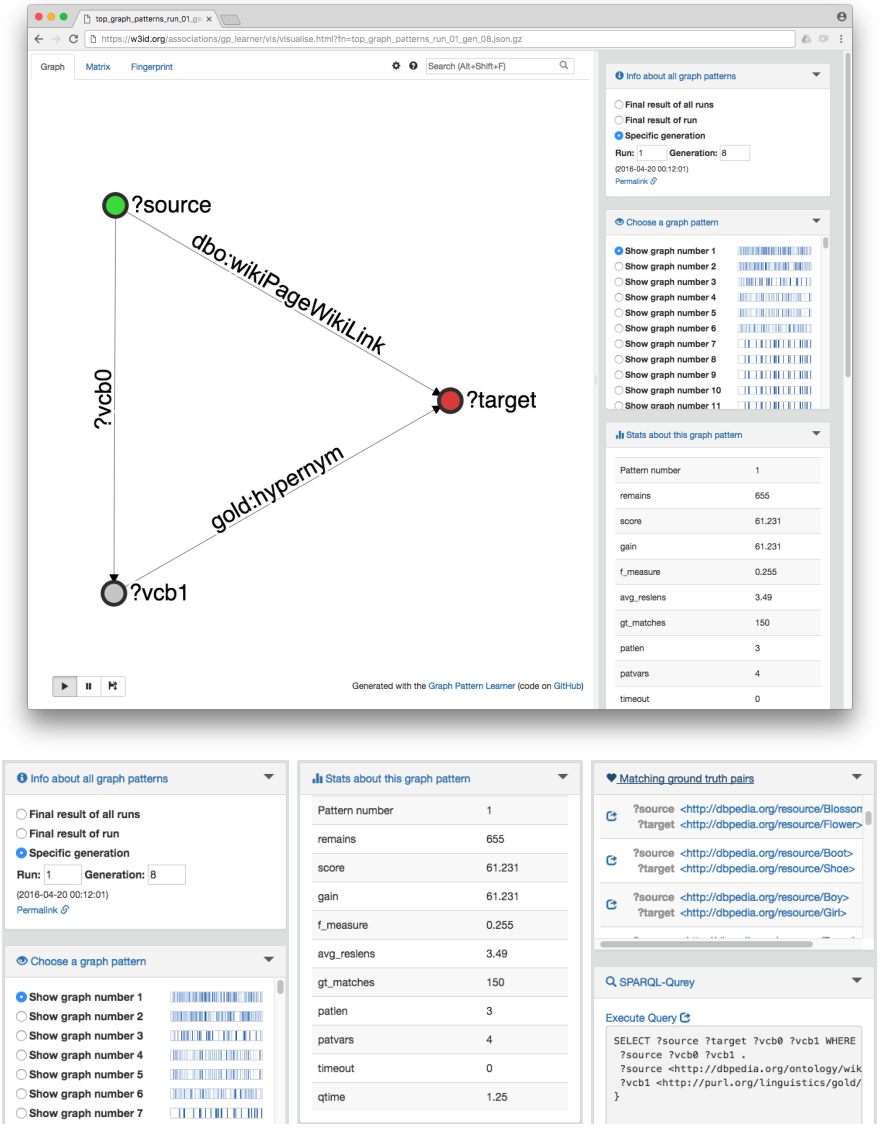


Figure 8.3: Visualisation of graph pattern 1 from run 1, generation 8 (top) and the side-panel (bottom) with selection of the stage (left), the pattern's fitness (middle), and the matching ground truth pairs and SPARQL Query (right).

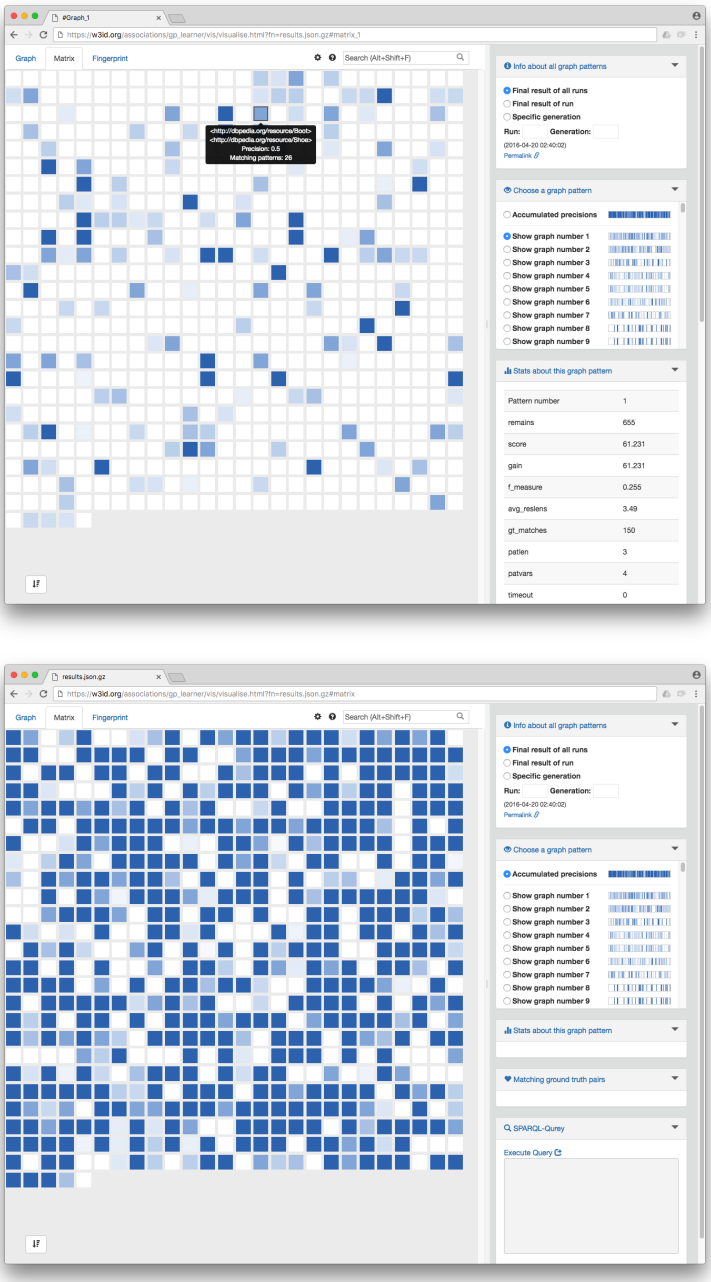


Figure 8.4: Visualisation of the coverage of graph pattern 1 over all runs and accumulated coverage over all patterns (bottom). Shown is the precision vector over all training ground truth pairs: Each block represents a source-target pair from the ground truth training set. The darker its colour the higher the precision for the ground truth pair. On the bottom the max-fused precision vectors over all graph patterns are shown. Hovering the mouse over a block shows additional information, as seen on top: Graph pattern 1 has a precision of 0.5 for the training set's ground truth pair (`dbr:Boot`, `dbr:Shoe`). The pair is covered by 26 patterns.

9 PATTERN BASED PREDICTION

After describing the core of our graph pattern learning algorithm in the previous chapter, we will now explain how the resulting learned ensemble of patterns can be used for prediction.

Patterns as predictors

As we remember from Section 7.3, each of the patterns learned already is a predictor: $\text{prediction}_{gp}(s)$. We can execute a *SELECT query* in which we bind the `?source` variable to a given (new) s and select the returned solutions for the `?target` variable. This results in one (unordered) list of target candidates per graph pattern that need to be fused to return a single ordered list of target predictions.

As our graph pattern learner sometimes produces several hundreds of graph patterns, we will first revisit our notion of ground truth coverage in Section 9.1. This will allow us to minimise the loss of coverage when reducing the amount of queries via clustering in Section 9.2. We will then describe how target candidates are generated (Section 9.3), how the resulting (reduced) patterns can be used as a feature space (Section 9.4), and how this allows us to fuse the individually generated predictions into a single ordered list of target predictions in Section 9.5. We will conclude this chapter with the description of an online demo that uses our findings in Section 9.6.

9.1 GROUND TRUTH COVERAGE

As briefly introduced in Section 7.3, we say that a graph pattern gp *covers* a ground truth pair $(s, t) = gtp \in \mathcal{GT}$ if the execution of the corresponding *ASK query* returns true. Further, as already shown in Figure 8.4 (in Section 8.9), we can extend this boolean notion by relying on the definition of $\text{precision}_{gp}(gtp_i)$. This allows us to define the *precision vector* $\mathbf{pv}_{gp} = (pv_{gp,1}, pv_{gp,2}, \ldots, pv_{gp,|\mathcal{GT}|}) \in \mathbb{Q}_{\geqslant 0}^{|\mathcal{GT}|}$, with $gtp_i \in \mathcal{GT}$ the i-th ground truth source-target pair and its components:

precision vector $\mathbf{pv}_{gp}$

$$pv_{gp,i} = \text{precision}_{gp}(gtp_i)$$

We also call the precision vector $\mathbf{pv}_{gp}$ the *ground truth precision coverage* vector, or simply *precision coverage* of gp.

ground truth precision coverage

Further, as we are ultimately interested in the coverage of our ground truth by an ensemble of patterns $GP_e \subseteq \text{gpl}(\mathcal{GT}, GP) \subset GP$, we can extend the notion to the *max precision coverage vector* $\mathbf{mpv}_{GP_e}$ w. r. t. the set of graph patterns GP_e. Its components are the max-fusion over all precision vectors:

max precision coverage vector $\mathbf{mpv}_{GP_e}$

$$mpv_{GP_e,i} = \max_{gp \in GP_e} \text{precision}_{gp}(gtp_i)$$

Intuitively, the max precision coverage vector tells us, which of the ground truth pairs was covered how well by all graph patterns in GP_e. The sum of its components (the *max precision vector sum* $\text{mpvs}(GP_e)$)

max precision vector sum $\text{mpvs}(GP_e)$

$$\text{mpvs}(GP_e) = \sum_{i=1}^{|\mathcal{GT}|} \text{mpv}_{GP_e,i}$$

gives us an overall measure for how well the ground truth pairs are covered by the graph patterns in GP_e. The max precision vector sum is the counter part of remains_{GP_e} (cf. Section 7.3.1):

$$\text{mpvs}(GP_e) = |\mathcal{GT}| - \text{remains}_{GP_e}$$

normalised max precision vector sum $\text{nmpvs}(GP_e)$

If desired, we can further normalise $\text{mpvs}(GP_e)$ to range $[0, 1]$:

$$\text{nmpvs}(GP_e) = \frac{\text{mpvs}(GP_e)}{|\mathcal{GT}|}$$

which can be interpreted as the average achieved precision over the ground truth pairs by all graph patterns in GP_e.

During the development of our algorithm, with default parameters, we typically achieved a normalised max precision vector sum of about 62% on the semantic association training ground truth pairs. This means that we can expect about 62% of our training ground truth pairs to be covered with high precision. At the same time, about 76.5% of them are covered by the graph patterns in $\text{gpl}(\mathcal{GT}, GP)$ at all.

Typically achieved coverage

While being a first indicator for the expectable performance of the patterns for prediction, we refer to Section 10.4 for a proper evaluation on the test set, which also includes the following query reduction and fusion approaches.

9.2 QUERY REDUCTION BY CLUSTERING

A reduction of queries for prediction is necessary, as the amount of learned "good" graph patterns $\text{gpl}(\mathcal{GT}, G)$ can easily become too large to still be efficient, even despite our canonicalisation efforts (cf. Section 7.3.3). One of the main reasons for this are similar or redundant patterns that are discovered within a single run of our algorithm. For example, the following two patterns are returned as very good patterns for our primary use-case.

gp_1 :
```
?source gold:hypernym ?target.
```

gp_2 :
```
?source gold:hypernym ?target. ?vcb0 owl:sameAs ?target.
```

Independently, each of the patterns gp_1 and gp_2 is a very good pattern and gets a high fitness, as each covers 28 of our training pairs and has an optimal average result length of 1. However, looking at the precision vectors, we find that $\mathbf{pv}_{gp_1} = \mathbf{pv}_{gp_2}$, meaning that both

patterns behave very similarly (in this case even equal) w. r. t. all training ground truth pairs.

Hence, to reduce the amount of graph patterns, we would like to find a minimal (reduced) sub-set $GP_r \subseteq gpl(\mathcal{GT}, G)$ that minimises the amount of redundancies, and at the same time minimises the *max precision loss*:

max precision loss

$$\text{loss}(GP_r) = \text{mpvs}(\text{gpl}(\mathcal{GT}, G)) - \text{mpvs}(GP_r)$$

and thereby also the normalised loss ratio:

$$\text{loss ratio}(GP_r) = \frac{\text{loss}(GP_r)}{\text{mpvs}(\text{gpl}(\mathcal{GT}, G))} = 1 - \frac{\text{mpvs}(GP_r)}{\text{mpvs}(\text{gpl}(\mathcal{GT}, G))}$$

We decided to solve this variant of the set-coverage problem in a heuristic way by clustering the graph patterns by their precision vectors and only keeping the fittest pattern in each cluster as representative of its cluster.

Currently, we employ hierarchical *clustering approaches*, as the number of meaningful clusters depends on the results of the graph pattern learner for the given use-case and hence is unknown in advance. Further, as we neither know which linkage method, nor which metric yields the best results in advance, we compute a clustering variant for each combination of common linkage methods (single, complete, weighted, average, centroid, median, and ward) with common metrics (euclidean, city-block and cosine), each in a raw and standard scaled ($\mu = 0, \sigma = 1$) form.

clustering approaches

Afterwards, to select a desirable clustering variant and amount of graph patterns to keep, we can plot the loss ratios of each variant $\text{loss ratio}(GP_{\text{cluster}(\text{variant},k)})$ over the number of clusters k. An example of such a plot can be seen in Figure 9.1. During development, in our primary use-case we could observe that the best clustering algorithms (frequently "scaled euclidean ward") achieved loss ratios of $\approx 10\%$ with 50 and $< 1\%$ with 100 requests, typically allowing us to save more than 300 queries per prediction with $< 1\%$ loss.

To ease decision making about the amount of graph patterns to keep for new use-cases, the precision loss plots can be generated via a simple command-line option. Further, if not specified manually, our algorithm automatically computes all variants, logs their losses and then selects the one which minimises the loss for the desired maximum number of queries to be performed during prediction (default: 100).

We will in the following denote the *resulting graph patterns* of the best clustering variant w. r. t. max precision loss as $GP_r \subseteq \text{gpl}(\mathcal{GT}, G)$.

resulting graph patterns GP_r

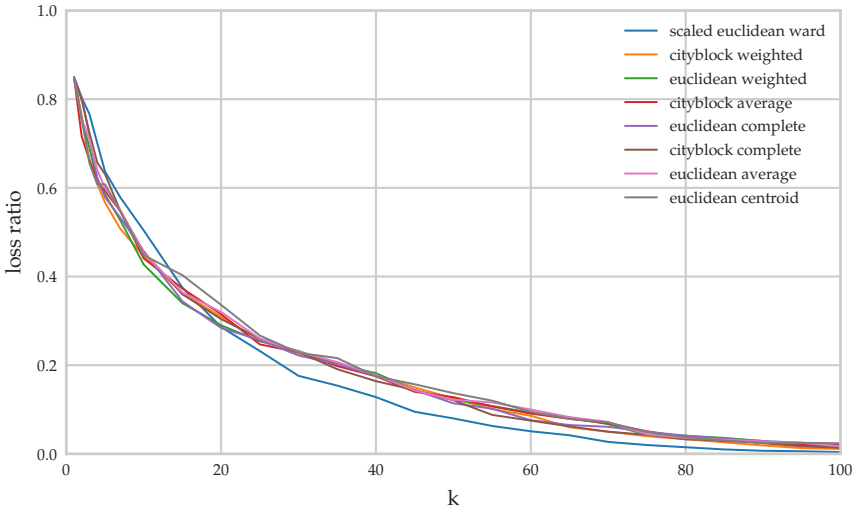


Figure 9.1: Query Reduction Precision Loss of the 8 best variants with the smallest Area Under the Curve (AUC) (lower is better).

9.3 TARGET CANDIDATE GENERATION

As already mentioned in Section 7.3, each individual $gp \in gpl(\mathcal{GT}, GP)$ can be used to predict target candidates for a (new) source s by executing the corresponding *SELECT query* after binding `?source` to s:

$$\text{prediction}_{gp}(s) = \underset{\texttt{?target}}{\text{SELECT}}(\phi_{\texttt{?source}:=s}(gp))$$

target candidates $TC_{GP_r}(s)$

Given an ensemble of result patterns $GP_r \subseteq gpl(\mathcal{GT}, GP)$, we can define the set of *target candidates* $TC_{GP_r}(s)$ generated for a given source s as the simple union of all target candidates generated by each of the patterns $gp \in GP_r$:

$$TC_{GP_r}(s) = \bigcup_{gp \in GP_r} \text{prediction}_{gp}(s)$$

9.4 PATTERNS AS FEATURE SPACE

Using an idea similar to the one in Section 9.1 and Section 9.2, which allowed us to cluster graph patterns by their precision vectors, we can represent each of the target candidates $t_i \in TC_{GP_r}(s)$ as boolean *coverage vector* $\mathbf{c_{t_i}} = (c_{t_i,1}, c_{t_i,2}, \ldots, c_{t_i,|GP_r|}) \in \mathbb{B}^{|GP_r|}$ of the graph patterns with its components:

coverage vector $\mathbf{c_{t_i}}$

$$c_{t_i,j} = \begin{cases} 1, \text{ if } t_i \in \text{prediction}_{gp_j}(s) \\ 0, \text{ otherwise} \end{cases}$$

We can further combine all of the target candidates and their coverage row vectors back into one large *coverage matrix* C:

coverage matrix C

$$C = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} gp_1 & gp_2 & \cdots & gp_m \end{array} \\ \begin{pmatrix} c_{t_1,1} & c_{t_1,2} & \cdots & c_{t_1,m} \\ c_{t_2,1} & c_{t_2,2} & \cdots & c_{t_2,m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{t_n,1} & c_{t_n,2} & \cdots & c_{t_n,m} \end{pmatrix} \end{array}$$

with $n = |TC_{GP_r}(s)|$ the number of target candidates and $m = |GP_r|$ the number of graph patterns.

Intuitively speaking, the row vectors of C tell us for one target candidate $t_i \in TC_{GP_r}(s)$ which of the graph patterns predicted it for a given source s. As the number of graph patterns can be considered as static from a perspective of C, each of the graph patterns gp_j now has a dual role: First, it is a SPARQL BGP, which allows us to generate target candidates for a given source s and further to explain why each of those target candidates was generated. Second, it now is a dimension of the m-dimensional vector space spanned by all graph patterns, into which all of their target candidates are embedded.

Dual role of graph patterns

9.5 FUSION METHODS

Looking at the feature space spanned by our graph patterns in the previous section, it becomes apparent that C contains a lot of information. In this section, we will describe ways to use this information to fuse the individual (unordered) lists of target candidates $\text{prediction}_{gp_j}(s)$ generated by each of the patterns in $gp_j \in GP_r \subseteq gpl(\mathcal{GT}, G)$ into one single ranked list of target predictions.

Formally, we can represent the *fusion of target candidates* as the problem of assigning a score $v_i \in \mathbb{R}$ to each of the $t_i \in TC_{GP_r}(s)$:

fusion of target candidates

$$\text{fusion}_{\text{score},GP_r}(s) = \{(t_i, v_i) \mid t_i \in TC_{GP_r}(s), v_i = \text{score}(t_i, \cdot)\}$$

with $\text{score}(t_i, \cdot)$ representing a scoring function, such as the ones described in the remainder of this section. The scoring functions will always depend on t_i, but often use further information from the context, such as the graph patterns GP_r, all generated target candidates $TC_{GP_r}(s)$, or the vector space representation C thereof, as we will see in Section 9.5.1. More advanced scoring functions will rely on full machine learning models trained on the above and their individual behaviours w. r. t. the training ground truth pairs $\mathcal{GT}$, as will be described in Section 9.5.2.

If contextually clear, we will for better readability also simply refer to the fused predictions as fp with $fp = \text{fusion}_{\text{score},GP_r}(s)$. Further,

we can order fp descending by the assigned score to retrieve the final ranked list of target predictions:

$$\text{ranking}(fp) = (t_1, t_2, \ldots, t_{|fp|}),\ \forall (t_i, v_i) \in fp : \forall i' < i : v_{i'} \geqslant v_i$$

$$\text{rank}_{fp}(t) = \begin{cases} i, \text{if } \exists t_i \in \text{ranking}(fp) : t_i = t \\ 0, \text{otherwise} \end{cases}$$

If not manually specified otherwise, our algorithm will (train and) compute all of the following fusion variants and their corresponding rankings. This will allow us to select the best variant for evaluation, as we will see in Section 10.4.5.

9.5.1 *Basic Fusion*

We will start with basic fusion methods that do not require any additional training and simply use information from the context, such as all the target candidates generated $TC_{GP_r}(s)$, which pattern each of them was generated by from our coverage vector space C, or information from the patterns GP_r themselves.

For simplicity, we will denote fitness attributes of a $gp \in GP_r$ with a $gp[\text{attrib}]$ notation in the following. For example, $gp[\text{score}]$ is the score of the fitness tuple (cf. Section 8.3). We will also define $gp[\text{precision}]$ as the inverse of the average result list length:

$$gp[\text{precision}] = \begin{cases} \frac{1}{gp[\text{avg result len}]}, \text{ if } gp[\text{avg result len}] > 0 \\ 0, \text{otherwise} \end{cases}$$

Further, we extend this notation to vector form over all graph patterns in GP_r:

$$\mathbf{GP_r}[\text{attrib}] = \begin{pmatrix} gp_1[\text{attrib}] \\ gp_2[\text{attrib}] \\ \vdots \\ gp_{|GP_r|}[\text{attrib}] \end{pmatrix}$$

target occs

- *target occurrences*: The simplest of our fusion methods. Its score is simply the count of graph patterns $gp_j \in GP_r$ that each target candidate $t_i \in TC_{GP_r}(s)$ was generated by. We can express this as the sum over the row vector corresponding to t_i in C:

$$\text{target occurrences}_C(t_i) = \sum_j C_{i,j}$$

scores

- *scores*: The sum of all graph pattern scores (from their fitness) of gp_j that returned t_i. This can be seen as a simple extension of the above scaled by the graph pattern scores:

$$\text{scores}_{C,GP_r}(t_i) = \sum_j C_{i,j}\, gp_j[\text{score}]$$

- F_1 *measures*: The sum of all graph pattern F_1-measures (from their fitness) of gp_j that returned t_i: *f measures*

$$\text{f measures}_{C,GP_r}(t_i) = \sum_j C_{i,j}\ gp_j[\text{f measure}]$$

- *gp precisions*: The sum of all graph pattern precisions (as the inverse of gp_j[avg result len]) of gp_j that returned t_i: *gp precisions*

$$\text{gp precisions}_{C,GP_r}(t_i) = \sum_j C_{i,j}\ gp_j[\text{precision}]$$

While the aforementioned fusion scoring functions only use information from the graph patterns' fitness and one of the target candidates, we can further take into account the actual precision during prediction time of each graph pattern. For this, we assign a total vote of 1 to each graph pattern and distribute its vote over all target candidates it predicted in each of the variants above. We denote these variants with postfix "precisions":

- *precisions* (short for target occurrences precisions): *precisions*

$$\text{precisions}_C(t_i) = \sum_j \frac{C_{i,j}}{\sum_k C_{k,j}}$$

- *scores precisions*: *scores precisions*

$$\text{scores precisions}_{C,GP_r}(t_i) = \sum_j \frac{C_{i,j}\ gp_j[\text{score}]}{\sum_k C_{k,j}}$$

- F_1 *measures precisions*: *f measures precisions*

$$\text{f measures precisions}_{C,GP_r}(t_i) = \sum_j \frac{C_{i,j}\ gp_j[\text{f measure}]}{\sum_k C_{k,j}}$$

- *gp precisions precisions*: *gp precisions precisions*

$$\text{gp precisions precisions}_{C,GP_r}(t_i) = \sum_j \frac{C_{i,j}\ gp_j[\text{precision}]}{\sum_k C_{k,j}}$$

9.5.2 *Advanced Fusion*

While the basic fusion methods have the advantage that they are very easy to compute, they might fail to use information that arises through the interplay of several graph patterns. One way to observe this interplay is by re-using the training ground truth dataset $\mathcal{GT}$ that was used by our graph pattern learner $\text{gpl}(\mathcal{GT}, G)$ to form the result

patterns GP_r. For each source-target pair $(s,t) \in \mathcal{GT}$ we can generate all target candidates $t_i \in TC_{GP_r}(s)$ and form our vector space representation C. Additionally, we can construct a label vector l:

$$l = \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{pmatrix}, \text{with } l_i = \begin{cases} 1, \text{ if } t_i = t \\ 0, \text{ otherwise} \end{cases} \quad \text{and } \sum_i l_i \leqslant 1$$

Intuitively l contains a boolean label for each row in C. Looking at the row vectors, we are interested in those vectors, that often have a 1 as label and less interested in those that frequently have a 0 label.

fusion model

It is easy to see that we can actually use the row vectors of C and corresponding labels l as training data for supervised machine learning algorithms. This allows us to train a model (in the following called a *fusion model*), that given a row vector shall predict how likely it is that the target candidate t_i belonging to the row vector C_i is the true target ($t_i = t$, label: true) or not ($t_i \neq t$, label: false). As we are ultimately searching for a scoring function for fusion, the prediction here can either be in form of a simple classification likelihood, a regression or (if supported by the model) an immediate (interrelated) scoring of all target candidates.

Stratified sampling

However, one of the challenges we face is the strong imbalance between positive (1) and negative (0) training samples. By design for each source-target pair only 1 true target exists, while the number of target candidates predicted by our patterns can easily reach 100 or more. We counter this by a configurable amount of samples per class (default: 500) that are randomly drawn from all row vectors.

Merging colliding vectors

Furthermore, there is no guarantee that vectors are label collision free: it can happen that $C_i = C_j$, but $l_i \neq l_j$. For regressors, we hence merge all equal row vectors in C into a row disjoint $\tilde{C}$ and average over their corresponding labels, turning them into a ratio $\tilde{l}_i$:

$$\forall k \exists i : \tilde{C}_i = C_k, \forall i \exists k : \tilde{C}_i = C_k, \forall j \neq i : \tilde{C}_i \neq \tilde{C}_j, \text{ and } \tilde{l}_i = \operatorname*{avg}_{\forall k : \tilde{C}_i = C_k} (l_k)$$

Optionally, such row vector merging can also be enabled for classifiers, using an additional ratio threshold to turn the ratios back into class labels true or false. Our experiments during development however showed that while having a positive effect on regressors, the effect in case of classifiers is negligible.

In the following, we will briefly list all supervised machine learning algorithms that we applied to our problem[1]:

- **Classifiers**: Given the row vectors C_i and a label l_i (true or false) for each, it is straight forward to model our fusion scoring task

1 For implementation we used the excellent scikit-learn python library [138].

as a 2 class classification problem. However, simply predicting 1 or 0 for each of potentially many row vectors has a rather limited value for ranking. Hence, we will instead use the prediction probability of the true class as returned score whenever using a classification algorithm.

The following classification algorithms are used: k-nearest neighbours, SVM (linear & RBF), decision tree, *random forest*, gradient boosting (gtb), *AdaBoost*, neural net, gaussian naive bayes, quadratic discriminant analysis (qda), stochastic gradient descent (sgd), and *logistic regression*.

random forest
adaboost
logistic regression

- **Regressors**: Alternatively, we can model our fusion scoring task as regression problem: Trained on the merged row vectors to predict the ratio of true labels $[0, 1]$, we can directly use the predictions as a scoring function for our fusion.

  The used regression algorithms (also including regression variants of classifiers) are: k-nearest neighbours, SVR (linear & RBF), decision tree, random forest, gradient boosting, AdaBoost, kernel ridge, *bayesian ARD*, bayesian ridge, elastic net, least angle regression (lars), lasso, lasso lars, linear, ridge, stochastic gradient descent (sgd), and multi layer perceptron (mlp).

  ARD regression

- **Learning to rank**: Last but not least, we can also model our fusion scoring task as a full fledged learning to rank problem. Learning to rank is a subclass of machine learning algorithms that are designed to calculate rankings/scores for a whole set of inputs at once. They can thereby in theory exploit dependencies between the different candidates for ranking.[2]

  As representative of this class of machine learning algorithms, we currently employ the *RankSVM* [99], a variant of the linear SVM.

  RankSVM

Default behaviour

If not selected manually, our algorithm will by default train each of the aforementioned machine learning models for later evaluation on the test set (cf. Section 10.4.5). The training of a model by default includes a parameter optimisation via a grid search if common for the algorithm over typical parameter ranges. Further for each model, we also perform a meta-optimisation with common pre-processing pipelines, such as standard scaling ($\mu = 0$, $\sigma = 1$) and/or length normalising of all vectors. The parameter and pre-processing optimisation is evaluated via a k-fold (default: $k = 3$) cross validation over the training ground truth dataset, by selecting the combination with the highest Mean Reciprocal Rank (MRR). The final model is re-trained on the full training dataset with the best pre-processing and parameters.

Parameter optimisation

2 They are however typically not optimised for our rather special ranking case, in which the training data does only consist of one true positive candidate and no known ranking difference between the remainder.

9.6 DEMO

After the aforementioned query reduction (Section 9.2) and fusion techniques (Section 9.5), we conclude this chapter with the description of an online demo. Parts of this section have already been published in [82].

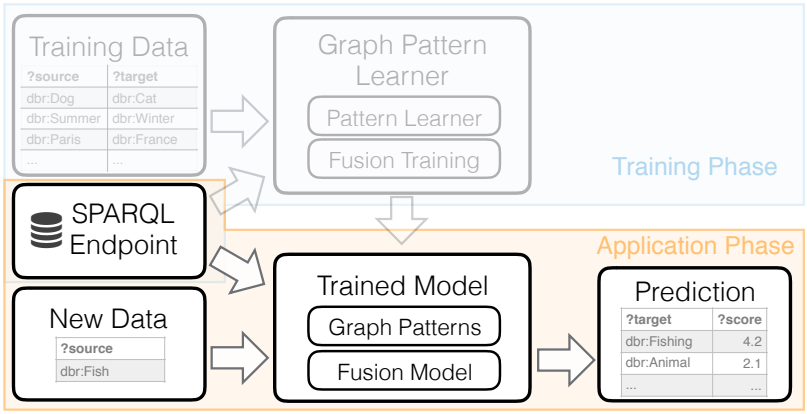


Figure 9.2: Graph Pattern Learner Application Phase

Summarising, the demo shows the application phase of our full system, as outlined in Figure 9.2. Given a set of learned and clustered graph patterns $GP_r \subseteq gpl(\mathcal{GT}, G)$ and trained fusion models, together forming the "trained model", the demo allows its users to enter an arbitrary new source s (e. g., `dbr:Fish`). It will then execute the prediction queries corresponding to all graph patterns GP_r against the given SPARQL endpoint G and fuse the individual unordered result lists into a single ranked list of target predictions.

After a short introduction, the main screen of our online demo[3] asks the user to enter a stimulus DBpedia entity for which to predict target candidates. For usability reasons, we realised the input-box as an auto-complete box (Figure 9.3) via the Wikipedia OpenSearch API[4], allowing a fuzzy search for matching Wikipedia Articles, including the resolving of redirects. After selecting one of the Wikipedia articles from the auto-complete box, the Wikipedia URI is transformed to the corresponding DBpedia resource, forming our input source s and starting the prediction.

Auto-complete

On server side, our demo will now perform all $predict_{gp}(s)$ queries for the clustered $gp \in GP_r \subseteq gpl(\mathcal{GT}, G)$ against our local SPARQL endpoint. In case of our online demo, the queries (currently 100) are performed in parallel, typically returning their (100) result lists within 2 seconds. The returned target candidate lists are then combined into one fused target prediction result list by the selected of our fusion models.

Prediction

Fusion

3 https://w3id.org/associations/gp_learner/demo/predict.html
4 https://www.mediawiki.org/wiki/API:Opensearch

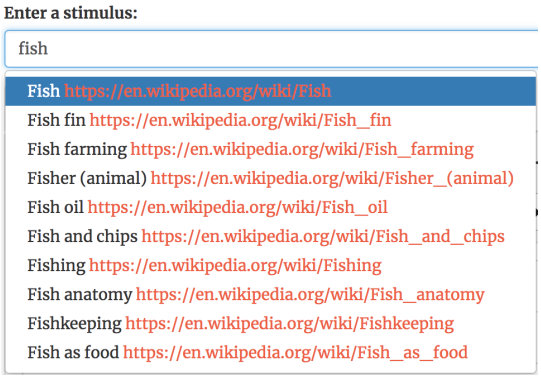


Figure 9.3: Demo: Stimulus Auto-Complete Input-Box

Selection of fusion method

The final prediction results are presented in the "Fused Prediction Results" tab (Figure 9.4 (left)) as predicted responses. The user can switch between several of the fusion variants (cf. Section 9.5) via a drop-down (Figure 9.4 (right)) and provide feedback about the generated targets. The feedback currently is logged to collect data for potential future improvements.

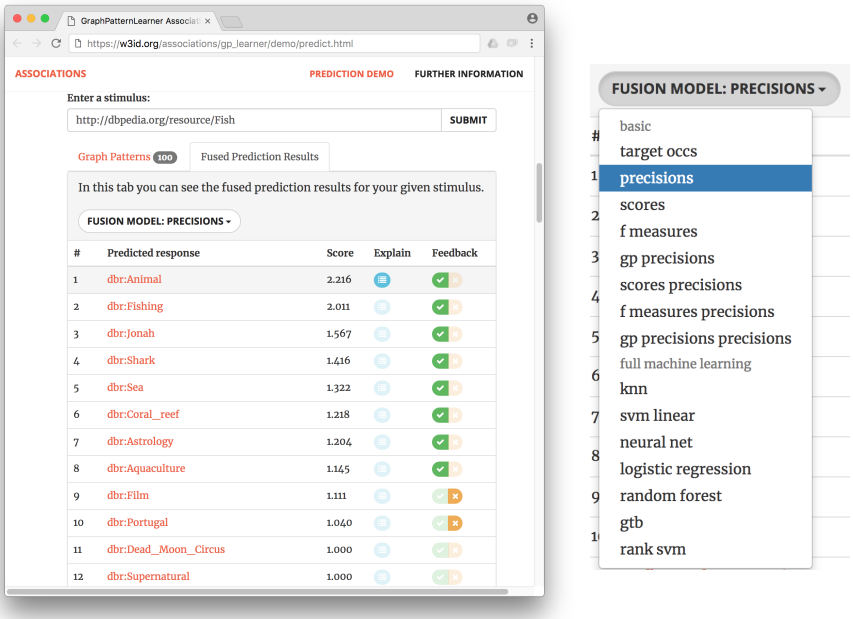


Figure 9.4: Demo: Prediction results for `dbr:Fish` (left) and fusion method drop-down (right)

Explainability

The user can also click the explain button next to each target candidate to gain insights into why it was predicted. A click on the button leads back to the initial "Graph Patterns" tab. While initially (without any entered stimulus) the tab only showed all of the graph patterns GP_r used for prediction (Figure 9.5 (left)), we now see that patterns that generated the target candidate to be explained are highlighted

and expanded (Figure 9.5 (right)). Further, for each of the graph patterns we can explore its fitness (Figure 9.6) from our evolutionary graph pattern learning algorithm, as well as the other target candidates it generated for the currently entered source node.

The demo and its code are available online[5] and base on a small API that we built into the graph pattern learner to allow us to simply turn a fully trained model into an online web-service.

5 https://w3id.org/associations

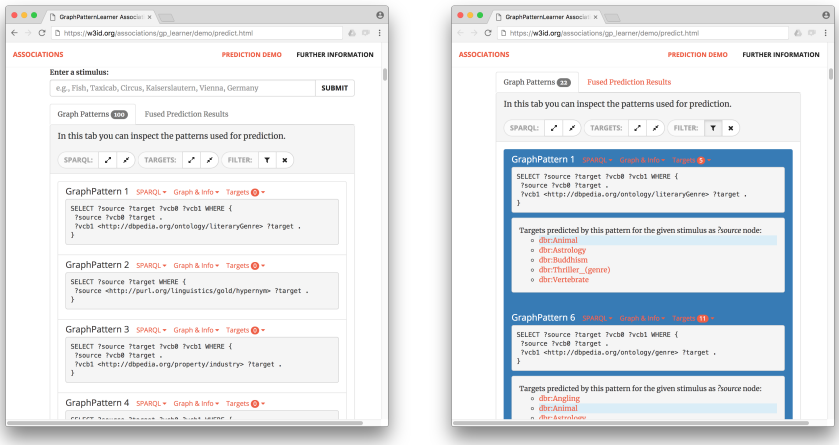


Figure 9.5: Demo: Patterns for prediction (left) and filtering on patterns that predicted `dbr:Animal` for stimulus `dbr:Fish` (right).

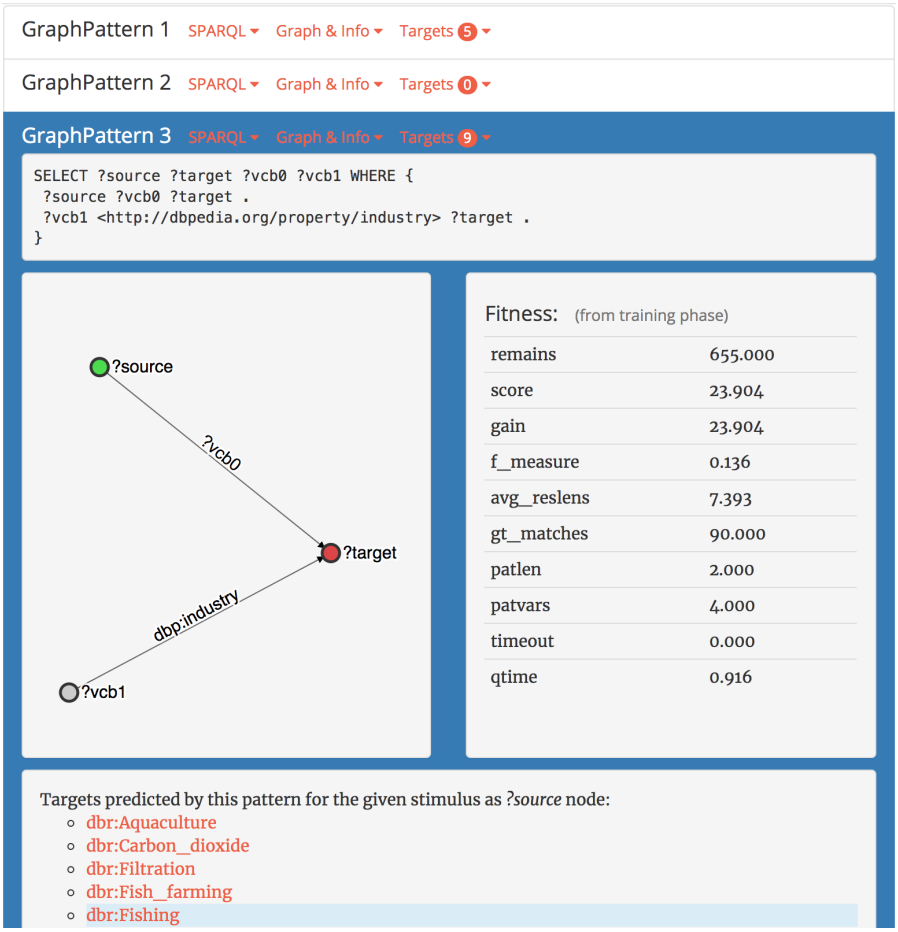


Figure 9.6: Demo: Fully expanded pattern, highlighted to explain why `dbr:Fishing` was predicted as target for stimulus `dbr:Fish`.

10 EVALUATION

10.1 OVERVIEW

After presenting our graph pattern learning approach in the previous chapters, we will extensively evaluate our approach in this chapter. Parts of this chapter have already been published in [80].

We will first describe the local and cluster setup used for our evaluations in Section 10.2, and briefly explain the used quality metrics in Section 10.3.

The main evaluation for simulating human associations can then be found in Section 10.4, including a brief recap of the datasets used (Section 10.4.1) and several baselines (Section 10.4.2), before reporting the results of our approach: We report basic statistics and the achieved training set coverage of our graph pattern learner (Section 10.4.3), mention notable learned graph patterns (Section 10.4.4), evaluate the prediction quality of our full system (Section 10.4.5), analyse the rank-degree correlations (Section 10.4.6), and the spread (Section 10.4.7).

Afterwards, we further evaluate the practical limits of our approach on artificially injected patterns of increasing complexity (Section 10.5). We conclude this chapter by comparing our fully trained human association model with the KORE relatedness fact rankings in Section 10.6.

10.2 DESCRIPTION OF LOCAL SETUP

As briefly mentioned in Section 1.3 and Section 2.1.1.2, we use a local *Linked Data mirror* in form of a *SPARQL endpoint* for our experiments and evaluation, in order to not disrupt the service of public endpoints.

In this section, we will briefly describe the setup of the local mirror, the loaded dataset and how the setup is scaled in cluster use-cases.

10.2.1 *Local Linked Data Mirror & Loaded Datasets*

Virtuoso as SPARQL endpoint

We used Virtuoso OpenSource[1] version 7.2.4.2 as local SPARQL endpoint for our experiments in this work. Even though the computational requirements for our evolutionary algorithm alone are minimal (we recommend more than 4GB of free RAM), network latency becomes a very noticeable factor with hundreds of thousand of SPARQL

1 https://github.com/openlink/virtuoso-opensource

HTTP requests. Hence, to reduce connection overhead, we decided to perform all computations directly on the server(s) which run(s) the SPARQL endpoint. As our algorithm already parallelises the fitness evaluation, we are practically mostly limited by the performance of the SPARQL endpoint. This also means that a single (parallelised) graph pattern learner can fully utilise a whole SPARQL endpoint and that it does not make sense to run multiple concurrent graph pattern learners against the same SPARQL endpoint.

Used servers

When running our local experiments and evaluations, we mainly used the following servers:

- Supermicro H8QG7, 2x AMD Opteron Processor 6328, 16 threads total, 256GB RAM, 1TB SSD storage, OS: Ubuntu 14.04 LTS

  This server was used for the main development and configured to run Virtuoso with 64GB of RAM. When running our evolutionary algorithm, we used 8 processes, leaving the rest to the SPARQL endpoint. In the current (relatively aggressive) standard configuration, our full algorithm terminates after around 8 hours.

- NVIDIA DGX-1, 2x Intel Xeon CPU E5-2698 v4, 80 threads total, 512GB RAM, 4x2TB SSD Raido storage, OS: Ubuntu 14.04 LTS

  This server was used mainly for its many CPUs, typically completing our algorithm in 2 hours with 40 parallel processes. The server is configured to run Virtuoso with 128GB of RAM.

The SPARQL endpoint was initially populated on our main development server. We imported several large dataset dumps from the centre of the LOD Cloud, as briefly mentioned in Section 2.1 and Section 6.3:

1. DBpedia 2015-04 EN Core[2] (~ 412M triples)
2. DBpedia 2015-04 EN Extended[3] (~ 160M triples, including ~ 159M Wikilinks (`dbo:wikiPageWikiLink`))
3. DBpedia 2015-04 DE[4] (~ 180M triples, including ~ 60M Wikilinks)
4. Freebase RDF[5] as of 2015-08-09 (~ 3.1G triples)
5. BabelNet 3.6 RDF[6] (~ 1.9G triples, crawled)
6. LinkedGeoData[7] as of 2014-09-09 (~ 1G triples)
7. Wikidata 2015-10-26 RDF[8] (~ 841M triples)

2 http://downloads.dbpedia.org/2015-04/core/
3 http://downloads.dbpedia.org/2015-04/core-i18n/en/
4 http://downloads.dbpedia.org/2015-04/core-i18n/de/
5 https://developers.google.com/freebase/
6 http://babelnet.org/rdf/
7 http://linkedgeodata.org
8 https://tools.wmflabs.org/wikidata-exports/rdf/exports/20151026/

8. DBLP[9] as of 2015-11-04 (~ 82M triples)
9. YAGO 3 Labels[10] (~ 45M triples)
10. Wordnet 3.1 RDF[11] (~ 5.6M triples)
11. OpenCyc RDF[12] as of 2012-05-10 (~ 2.4M triples)
12. Wordnet 2.0 RDF[13] (~ 1.9M triples)
13. UMBEL[14] as of 2015-11-15 (~ 480k triples)
14. Schema.org[15] as of 2015-11-04 (~ 8.7k triples)

During the import process we created three images (full copies) of the Virtuoso database[16], to which we will refer as follows:

- *core dataset* (~ 412M triples): Only contains the first dataset (DBpedia 2015-04 EN Core). *core dataset*
- *extended dataset* (~ 572M triples): Contains datasets 1-2 (DBpedia 2015-04 EN Core and Extended dataset). *extended dataset*
- *all datasets* (~ 7.9G triples): Contains datasets 1-14. *all datasets*

The different (static) Virtuoso database images allow us to quickly run a desired database image on any server, simply by copying the database directory and starting Virtuoso locally. The majority of our experiments during development was run against all datasets. The main benefit of the smaller datasets is that they can efficiently be run on servers with less than 64GB of free RAM. As our graph pattern learner currently is graph agnostic, they also allow us to easily run our algorithm on smaller sub-sets. The core and extended datasets were also used for our analysis in Section 6.3.

10.2.2 *Cluster Setup*

Parts of our experiments were run on the Elwetritsch cluster[17], especially to evaluate the qualitative spread of our algorithm (cf. Section 10.4.7) and to perform the thousands of runs necessary for our pattern injection evaluation (cf. Section 10.5).

At the time of our experiments, the Elwetritsch cluster consisted of ~ 500 compute nodes (servers) with different configurations totalling

9 http://dblp.l3s.de
10 http://www.yago-knowledge.org
11 http://wordnet-rdf.princeton.edu/
12 http://opencyc.org
13 https://www.w3.org/2006/03/wn/wn20/
14 http://umbel.org/
15 http://schema.org
16 Available online via https://w3id.org/associations.
17 https://elwe.rhrk.uni-kl.de

in 7600 cores, 33.9TB RAM, 244TB node local SSD temp storage and 330TB persistent network storage.

Spread experiments

For our spread experiments we mostly relied on about 250 compute nodes with $\geqslant$ 16 cores and $\geqslant$ 64GB RAM. The pattern injection evaluation was executed on the extended dataset, allowing us to rely on an additional 150 compute nodes with 32GB RAM.

Pattern injection evaluation

As, especially for the injection evaluation, the database needs to be modified independently, and as a single graph pattern learner can already fully utilise a SPARQL endpoint, we decided to parallelise our experiments similar to the local setup: For execution of an experiment on a compute node, we ship the database image to the compute node and run a node local Virtuoso database that is only used by the node local graph pattern learner. To counter the initial setup cost (database transfer), we allow several consecutive experiments on one node to re-use the already local database image (after resetting any changes).

Scaling by localising databases

Relying on this setup, we were able to efficiently parallelise about 200k CPU hours worth of experiments and evaluations.

10.3 USED QUALITY METRICS

To evaluate the result quality of our graph pattern learner, we evaluate how well the overall approach (including query reduction by clustering and fusion) can simulate our ground truth (test) source-target pairs $\mathcal{GT}_{\text{test}}$. We calculate common metrics, such as Recall@k, MAP, MRR and NDCG: for each test set source-target pair $(s,t) \in \mathcal{GT}_{\text{test}}$ we generate the ranked predictions ($\text{rp}_{\text{variant}}(s)$) and determine the rank $r_{s,t}$ of the true target t in our prediction result (cf. Section 9.5):

$$\text{rp}_{\text{variant}}(s) = \text{ranking}(\text{fusion}_{\text{variant},GP_r}(s))$$
$$r_{s,t} = \text{rank}_{\text{fusion}_{\text{variant},GP_r}(s)}(t)$$

10.3.1 *Recall@k*

After computing the ranked predictions for each of our test set source-target pairs, an obvious question is how frequently the true target is returned within the first k results of each ranked prediction. Given that we only have one true (relevant) target t for each prediction $\text{rp}_{\text{variant}}(s)$, this question is answered by the Recall@k. In our case *Recall@k* can be defined as follows:

Recall@k

$$\text{Recall@k} = \frac{|\{(s,t) \in \mathcal{GT}_{\text{test}} \mid r_{s,t} \leqslant k\}|}{|\mathcal{GT}_{\text{test}}|}$$

10.3.2 *Mean Average Precision (MAP) & Mean Reciprocal Rank (MRR)*

While Recall@k is easy to interpret and plot, we are sometimes interested in a single number describing the quality of an approach.

Two such metrics are the Mean Average Precision (MAP) and the Mean Reciprocal Rank (MRR) which, as we will see, are equivalent in our case due to the fact that we only have one true target per ranked prediction.

MAP is typically defined over a set of queries Q on a document corpus with help of the average precision AvgPrecision over a single ranked result list: *MAP*

$$\text{AvgPrecision} = \frac{1}{|\text{relevant docs}|} \sum_{k=1}^{n} \text{Prec}(k)\,\text{rel}(k)$$

where k is the rank of the current ranked document, n is the number of retrieved documents, $\text{Prec}(k)$ the precision at k and $\text{rel}(k) = 1$ if the doc at k is relevant, else 0.

In our case, given that only one "document" (the true target) is relevant and we know its rank $r = r_{s,t}$, the $\text{Prec}(r) = \frac{1}{r}$. This lets us simplify the above to:

$$\text{AvgPrecision} = \text{Prec}(r)\,\text{rel}(r) = \frac{1}{r_{s,t}}$$

which is also known as the reciprocal rank.

With

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AvgPrecision}(q)$$

it follows that in our case

$$\text{MAP} = \frac{1}{|\mathcal{GT}_{\text{test}}|} \sum_{(s,t) \in \mathcal{GT}_{\text{test}}} \frac{1}{r_{s,t}} = \text{MRR}$$

We see that in our case MAP = MRR. Even though MAP is the more commonly reported metric, we will use MRR in the following to highlight the fact that we only have one relevant target per ranked prediction list. *MRR*

10.3.3 *NDCG*

Another common quality metric is the Normalised Discounted Cumulative Gain (NDCG), which Manning, Raghavan, and Schütze [117] define as follows:

> For a set of queries Q, let $R(j, d)$ be the relevance score assessors gave to document d for query j. Then,
>
> $$\text{NDCG}(Q, k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R(j,m)} - 1}{\log_2(1+m)}$$
>
> where Z_{kj} is a normalisation factor calculated to make it so that a perfect ranking's NDCG at k for query j is 1. For queries for which $k' < k$ documents are retrieved, the last summation is done up to k'.

Specialised for given ranks, $|\mathcal{GT}_{\text{test}}|$ queries, $Z_{kj} = 1$ and only one relevant document (makes the inner sum disappear as only one term in the sum is $\neq 0$):

$$\text{NDCG}(k) = \frac{1}{|\mathcal{GT}_{\text{test}}|} \sum_{(s,t) \in \mathcal{GT}_{\text{test}} : r_{s,t} \leqslant k} \frac{1}{\log_2(1 + r_{s,t})}$$

For simplicity, we also write NDCG instead of $\text{NDCG}(\infty)$ in the following.

10.4 SIMULATION OF HUMAN ASSOCIATIONS

After these descriptions of our local setup and the used quality metrics, in this section we will evaluate how well our algorithm performs on our primary use-case: the simulation of human associations.

We start this section with a brief mention of the used ground truth dataset (Section 10.4.1) and introduce several baselines (Section 10.4.2). Afterwards, we will provide basic statistics and the achieved training set coverage of our graph pattern learner (Section 10.4.3), and mention notable learned graph patterns (Section 10.4.4), before evaluating the prediction quality of our full system (Section 10.4.5). We will conclude this section with an analysis of rank-degree correlations (Section 10.4.6) and (as our approach is non-deterministic) the prediction quality spread of our full system (Section 10.4.7).

10.4.1 *Dataset*

For all evaluations in this chapter, we used the resulting semantic association ground truth dataset $\mathcal{GT}$ as mentioned in Section 6.2. As already described, the dataset consists of 727 source-target pairs ($|\mathcal{GT}| = 727$).

Before starting the development of our approach, we have randomly performed a static 9:1 training-test set split, forming a training set $\mathcal{GT}_{\text{train}}$ of 655 training source-target pairs and a test set $\mathcal{GT}_{\text{test}}$ of 72 pairs. A full list of all training and test set pairs can be found in Table B.1.

All development and training of our models have been performed on the training set in order to reduce the chance of over-fitting our algorithm to our ground truth. If not explicitly stated otherwise, all following results are reported over the ground truth test set $\mathcal{GT}_{\text{test}}$.

10.4.2 *Baselines*

In order to evaluate our approach, we compare it with several own and already existing methods as baselines. A comparison of all baselines and their computed quality metrics on $\mathcal{GT}_{\text{test}}$ can be found in Table 10.1 in Section 10.4.2.7.

10.4.2.1 *Neighbourhood Based*

When trying to predict target candidates for a given source node s in a semantic network, one obvious idea is to focus on the nodes from the neighbourhood of s.

We can define the in-neighbourhood (denoted as *in nb*) of s as nodes that link to s (`?t ?p s`) and the out-neighbourhood (denoted as *out nb*) of s as nodes that s links to (`s ?p ?t`). Further, we can define the any-neighbourhood (denoted as *any nb*) as the union of the above and the bidi-neighbourhood (denoted as *bidi nb*) as the intersection.

in nb

out nb

any nb

bidi nb

Further, we can restrict the type of links (`?p`), for example to Wikilinks (`dbo:wikiPageWikiLink`), which we will denote with a *wl* postfix. For example, with "out nb wl" we describe all nodes that s links to with a `dbo:wikiPageWikiLink`.

wl

From a prediction standpoint for a given s we now have several variants to form a set of target candidates, but until now they are unordered.

One simplistic way of ranking them is by random shuffling (random guessing baseline), but as known from Section 6.3 the in- and out-neighbourhoods of all of our nodes are so big that this will only achieve negligible expectable Recall@10 of $< 1\%$. For this reason, the random guessing baseline is not included in Table 10.1.

A better approach to order the target candidates consists of ordering them descending by their in-degree (denoted as *indeg*) or the out-degree (*outdeg*).

indeg

outdeg

A more advanced ordering is by descending PageRank or HITS score for each target candidate. As mentioned in Section 3.2.1.1 we use the pre-computed scores from [171] for this. We denote these orderings with a *pagerank* and *hits* postfix.

pagerank

hits

10.4.2.2 *Neighbourhood Overlap Based*

While the above baselines only focus on the 1-neighbourhood of a given node s, a common relatedness measure, the *Milne-Witten Relatedness* [126], calculates the similarity of two nodes s and t based on their in-neighbourhood overlap in comparison to the individual in-neighbourhoods. Let I_s be the *in nb wl* set of s and I_t the in nb wl set of t, then

Milne-Witten Relatedness

$$mw_{s,t} = \frac{\log(\max\{|I_s|, |I_t|\}) - \log(|I_s \cap I_t|)}{\log(n) - \log(\min\{|I_s|, |I_t|\})}$$

with n the number of DBpedia resources.

The Milne-Witten Relatedness can thereby easily be computed for any given candidate t, but it does not provide an efficient way to find such target candidates. Especially with node in-degrees $\gg 1000$, as mentioned in Section 6.3, simply selecting all `?t` from

```
?i dbo:wikiPageWikiLink s . ?i dbo:wikiPageWikiLink ?t
```

is computationally infeasible.

As we are searching for target candidates with high overlaps however, we decided to compute the Milne-Witten Relatedness only for the top-k target candidates retrieved with the query:

```
SELECT ?t (COUNT(DISTINCT *) as ?c) WHERE {
    ?i dbo:wikiPageWikiLink s .
    ?i dbo:wikiPageWikiLink ?t .
} ORDER BY DESC(?c) LIMIT k
```

mw wl

We will denote the top-300 target candidates generated in this fashion and ranked by their Milne-Witten Relatedness as *mw wl*. When varying k between 1 and 300, we found that $k = 5$ works especially well, leading us to also report the top-5 overlap candidates ranked by descending Milne-Witten Relatedness as *mw wl top5overlap*. The attentive reader will observe that the Recall@k for the latter does not change for $k \geqslant 5$, as the method only predicts up to 5 ranked targets.

mw wl top5overlap

10.4.2.3 *Vector Space Based*

Apart from these simple neighbourhood based baselines, we also compare against a series of common vector space models, as explained in Section 3.2.2.

To use the following models as prediction baselines, we devise 3 variants of how to retrieve potential target candidate vectors $\mathbf{t}$ for a given source vector $\mathbf{s}$:

sim

- Simple similarity (postfix *sim*):

  A simple similarity search around s: up to 100 target candidates are returned ordered descending by their distance $\|\mathbf{s} - \mathbf{t}\|$.

pred

- Prediction vector (postfix *pred*):

  Using the training source-target pairs $(s_i, t_i) \in \mathcal{GT}_{\text{train}}$ and their corresponding individual vector space representations $\mathbf{s_i}$ and $\mathbf{t_i}$, we can form an average prediction vector $\mathbf{p}$ of all associations.

  $$\mathbf{p} = \frac{1}{|\mathcal{GT}_{\text{train}}|} \sum_{(s_i,t_i) \in \mathcal{GT}_{\text{train}}} (\mathbf{t_i} - \mathbf{s_i})$$

  Using this vector, we can search up to 100 target candidates t for a given s around $\mathbf{s} + \mathbf{p}$ and order them descending by their distance $\|(\mathbf{s} + \mathbf{p}) - \mathbf{t}\|$.

simpred

- Similarity weighted prediction vectors (postfix *simpred*):

  Expanding on the previous idea, there might not exist a best single association prediction vector. Hence, for a given $\mathbf{s}$ we can instead focus on the top-10 most similar $\mathbf{s_i}$ from the training set (the 10 nearest neighbours from training for the given source). Let $d_i = \|\mathbf{s} - \mathbf{s_i}\|$ be their distances to $\mathbf{s}$. We can then use each

of their corresponding $\mathbf{t_i}$ to get the analogy vector $\mathbf{p_i} = \mathbf{t_i} - \mathbf{s_i}$ for the association $(s_i, t_i) \in \mathcal{GT}_{\text{train}}$. For each of these, analogy vectors we generate up to 100 target candidates t around $\mathbf{s} + \mathbf{p_i}$ and score them with $d_i \cdot \|(\mathbf{s} + \mathbf{p_i}) - \mathbf{t}\|$. Finally, we return the top-100 over all of the generated target candidates ranked by their ascending score sum.

With these prediction variants, we compare against the following vector space models (cf. Section 3.2.2) after the necessary vocabulary transformations[18] if needed.[19]

- Word2Vec [124] (denoted as *word2vec*) *word2vec*

  We use the GoogleNews-vectors-negative300 available at: `https://code.google.com/archive/p/word2vec/`.

- RDF2Vec [150, 152] & GloRDFVe [39] (denoted as *rdf2vec*) *rdf2vec*

  We tried all RDF2Vec and Global RDF Vector Space Embeddings models available at:
  `http://data.dws.informatik.uni-mannheim.de/rdf2vec/`
  In Table 10.1 we only report the performance of the best model:
  `DBpedia/2015-10/4depth/cbow/DB2Vec_cbow_500_5_5_2_500`
  The performances of all available models is listed in Table C.1 in Appendix C.

- NASARI [33, 34] (denoted as *nasari*) *nasari*

  Due to vocabulary incompatibilities, we were only able to use the EN binary embed vector model available at:
  `http://lcl.uniroma1.it/nasari`

10.4.2.4 *Content Based*

Another group of approaches to predict target candidates for a given source s uses the contents of the corresponding Wikipedia articles. The idea here is to return such articles that are similar to the given source node's article.

We represent these approaches with a simple Lucene[20] full-text index of the English Wikipedia. For a given source, we retrieve the full-text of its corresponding Wikipedia article and perform a similar documents search with it. We refer to this document similarity approach as *wiki doc sim*. *wiki doc sim*

18 Not all models use DBpedia URIs as labels (their vocabulary) for their vectors, but many of them use some variant of Wikipedia related identifiers. Where possible, we use this to rewrite the in- and outputs from and to DBpedia URIs.

19 To load the models, we relied on the excellent gensim [147] python library.

20 `https://lucene.apache.org/`

10.4.2.5 *Text Corpus Based*

Additionally, as briefly mentioned in Section 2.2.2, we also compare against three text corpus based baselines to simulate human associations: Rapp [146] (denoted as *(text rapp)*), Washtell and Markert [180] (denoted as *(text washtell)*), and Galea and Bruza [63] (denoted as *(text galeabruza)*).

(text rapp)
(text washtell)
(text galeabruza)

Unlike the other methods, these baselines already evaluate themselves against EAT, but don't provide ready to use models online. Hence, where possible, we report their quality metrics (missing values are due to missing information in the original publications). To emphasise that these baselines are not computed on the same ground truth dataset, we enclose them in brackets.

10.4.2.6 *Human Performance*

Last but not least, we also computed the average human top response prediction performance for our $\mathcal{GT}$. We do this by using the reported response counts $c_{s,r}$ (out of 100) of our verified mappings listed in Table A.1 in Appendix A. We first group over all resulting stimulus and response URIs and average the counts leading to the same pair of URIs. Afterwards, for each stimulus URI we select only the response URI that achieved the highest averaged count. Finally, we average over all these top average counts, leaving us with the average human top response prediction performance for $\mathcal{GT}$, which is 32.7% over all ground truth pairs with a standard deviation of 12.1%.

Using the information about training and test set split from Table B.1 in Appendix B, we can further compute the average human top response prediction performance for our ground truth training set $\mathcal{GT}_{\text{train}}$ of 32.5% ($\sigma = 11.8\%$) and our test set $\mathcal{GT}_{\text{test}}$ of 34.1% ($\sigma = 14.2\%$). We also report the latter as Recall@1 of *human performance* in Table 10.1.

human performance

10.4.2.7 *Comparison of Baselines*

A comparison of all aforementioned baseline approaches can be found in Table 10.1. Summarising, we can say that our top-5 overlap adaptation of the Milne-Witten Relatedness (*mw wl top5overlap*) outperforms all other baselines w. r. t. MRR and NDCG. It is however outperformed in Recall@1 by the simpler neighbourhood variant *bidi nb wl indeg*.

While the text baselines did not report enough statistics to compute MAP, MRR or NDCG, *(text rapp)* clearly outperforms all other baselines with a Recall@1 of 27% (also higher than any other Recall@2). For Recall@10 *(text galeabruza)* is better than all other baselines with a Recall@10 of 44%. In between, *mw wl top5overlap* is better or at par with the best of the text baselines.

Focusing on the simple neighbourhood baselines, we can see that for most a ranking by descending page rank is best. However, the

Table 10.1: Comparison of Baselines. (All values in percent.)

				Recall@k							
	method	MRR	NDCG	1	2	3	5	10	25	50	100
baselines	any nb hits	5.0	11.0	1.4	4.2	4.2	6.9	12.5	16.7	27.8	38.9
	any nb indeg	10.4	20.5	4.2	6.9	11.1	13.9	19.4	**41.7**	55.6	63.9
	any nb outdeg	4.5	13.1	1.4	1.4	1.4	5.6	12.5	22.2	33.3	54.2
	any nb pagerank	**11.3**	**21.9**	**5.6**	**6.9**	**11.1**	**15.3**	**23.6**	**41.7**	**58.3**	**68.1**
	any nb wl hits	5.0	11.0	1.4	4.2	4.2	6.9	12.5	16.7	29.2	38.9
	any nb wl indeg	10.5	20.6	4.2	**6.9**	12.5	13.9	19.4	**41.7**	56.9	63.9
	any nb wl outdeg	4.5	13.2	1.4	1.4	1.4	5.6	12.5	22.2	34.7	54.2
	any nb wl pagerank	**11.5**	**22.1**	**5.6**	**6.9**	**12.5**	**15.3**	**23.6**	**41.7**	**59.7**	**68.1**
	in nb hits	4.6	9.5	1.4	2.8	4.2	8.3	11.1	15.3	22.2	31.9
	in nb indeg	12.9	20.7	5.6	11.1	**15.3**	18.1	**30.6**	41.7	45.8	51.4
	in nb outdeg	2.9	10.5	0.0	0.0	0.0	4.2	9.7	20.8	33.3	45.8
	in nb pagerank	**14.0**	**21.9**	**6.9**	**12.5**	**15.3**	**19.4**	29.2	**43.1**	**47.2**	**52.8**
	in nb wl hits	4.7	9.8	1.4	2.8	4.2	8.3	11.1	15.3	22.2	33.3
	in nb wl indeg	12.9	20.9	5.6	11.1	15.3	18.1	**30.6**	41.7	45.8	**52.8**
	in nb wl outdeg	2.9	10.5	0.0	0.0	0.0	4.2	9.7	20.8	33.3	45.8
	in nb wl pagerank	**14.2**	**22.0**	**6.9**	**12.5**	**16.7**	**19.4**	**30.6**	**43.1**	**47.2**	**52.8**
	out nb hits	10.2	18.1	5.6	5.6	11.1	15.3	18.1	36.1	45.8	51.4
	out nb indeg	12.1	20.0	5.6	**12.5**	**15.3**	15.3	19.4	37.5	48.6	52.8
	out nb outdeg	**12.6**	**20.9**	**6.9**	**12.5**	**15.3**	**16.7**	18.1	**38.9**	**50.0**	**56.9**
	out nb pagerank	11.7	19.5	5.6	9.7	**15.3**	**16.7**	**20.8**	36.1	48.6	51.4
	out nb wl hits	10.5	18.3	5.6	6.9	11.1	15.3	18.1	36.1	45.8	51.4
	out nb wl indeg	13.0	20.7	6.9	**13.9**	**15.3**	15.3	19.4	37.5	48.6	52.8
	out nb wl outdeg	**13.6**	**21.6**	**8.3**	**13.9**	**15.3**	**16.7**	18.1	**38.9**	**50.0**	**56.9**
	out nb wl pagerank	12.7	20.3	6.9	11.1	**15.3**	**16.7**	**20.8**	36.1	48.6	51.4
	bidi nb hits	14.6	19.5	9.7	12.5	13.9	20.8	27.8	33.3	**36.1**	**37.5**
	bidi nb indeg	**21.4**	**24.8**	**15.3**	**20.8**	**26.4**	**31.9**	31.9	34.7	**36.1**	36.1
	bidi nb outdeg	18.4	22.8	11.1	19.4	20.8	27.8	**34.7**	**36.1**	**36.1**	**37.5**
	bidi nb pagerank	18.9	22.9	12.5	19.4	20.8	29.2	31.9	34.7	**36.1**	36.1
	bidi nb wl hits	15.0	19.8	9.7	12.5	16.7	20.8	27.8	33.3	**36.1**	**37.5**
	bidi nb wl indeg	**21.4**	**24.8**	**15.3**	**20.8**	**26.4**	**31.9**	33.3	34.7	**36.1**	36.1
	bidi nb wl outdeg	18.4	22.8	11.1	19.4	20.8	27.8	**34.7**	**36.1**	**36.1**	**37.5**
	bidi nb wl pagerank	20.2	23.8	13.9	20.8	22.2	29.2	31.9	34.7	**36.1**	36.1
	mw wl	11.0	17.9	6.9	9.7	11.1	13.9	18.1	23.6	**37.5**	**50.0**
	mw wl top5overlap	**22.0**	**25.6**	**13.9**	**20.8**	**31.9**	**36.1**	**36.1**	**36.1**	36.1	36.1
	word2vec sim	5.4	8.1	2.8	5.6	6.9	8.3	9.7	13.9	15.3	19.4
	word2vec pred	7.3	11.0	**4.2**	**6.9**	6.9	9.7	12.5	18.1	20.8	26.4
	word2vec simpred	**7.8**	**12.5**	**4.2**	5.6	**8.3**	**12.5**	**15.3**	**23.6**	**27.8**	**31.9**
	rdf2vec sim	8.6	11.5	4.2	9.7	12.5	13.9	15.3	18.1	19.4	22.2
	rdf2vec pred	**10.3**	**14.4**	**5.6**	**12.5**	**13.9**	**15.3**	**16.7**	19.4	23.6	31.9
	rdf2vec simpred	8.4	13.7	**5.6**	5.6	6.9	11.1	**16.7**	**20**.8	**31.9**	**36.1**
	nasari sim	**4.0**	**7.8**	**1.4**	**2.8**	**2.8**	5.6	**12.5**	**16.7**	**19.4**	**23.6**
	nasari pred	3.8	7.7	**1.4**	1.4	**2.8**	**6.9**	**12.5**	**16.7**	**19.4**	**23.6**
	nasari simpred	2.1	4.9	**1.4**	1.4	1.4	1.4	2.8	9.7	13.9	18.1
	wiki doc sim	6.6	12.5	4.2	5.6	5.6	6.9	12.5	19.4	29.2	40.3
	(text rapp)	-	-	**27.0**	**-**	**-**	-	-	-	-	-
	(text washtell)	-	-	11.0	13.0	16.0	20.0	22.0	33.0	41.0	53.0
	(text galeabruza)	-	-	16.0	21.0	26.0	**36.0**	**44.0**	**55.0**	**63.0**	**70.0**
	human performance	-	-	**34.1**	-	-	-	-	-	-	-

overall best of these variants *bidi nb wl indeg* achieves its optimum when ranking by in-degree. Once more (cf. Section 6.3), we can observe that a bidirectional Wikilink seems to contain a lot of information that can be used for the simulation of human associations.

To our surprise, the Wikipedia article similarity (*wiki doc sim*) and even the best of the vector space models (*rdf2vec pred*) do not perform too well compared to the other (much simpler) baselines. One potential reason for this is that the document similarity and vector spaces mostly model similarity, not associations. Another reason could be that the analogy conserving capabilities of the vector space models, which we tried to exploit by our three variants, are not sufficient to simulate human associations on their own. With enough training data, it would however be interesting to investigate how well machine learning approaches on top of these vector space models perform.

10.4.3 *Basic Statistics & Achieved Training Coverage*

During our development and various experiments, we typically ran our graph pattern learner $\text{gpl}(\mathcal{GT}_{\text{train}}, G)$ with the (rather aggressive) default configuration of a population size of 200, up to 32 instantiations per fix var mutation, a maximum of 20 generations each in a maximum of 64 runs (cf. Section 8.2).

The first 5 runs of our algorithm are typically completed within 6, 12, 18, 26 and 30 minutes. In the first couple of minutes, all of the very simple patterns that model a considerable fraction of the training set's pairs are found. Within the first hour we typically reach a max precision vector sum on the training set of $> 50\%$ (cf. Section 9.1). The remaining time is spent on minor improvements, by finding patterns for the remaining niches that were not covered by the other patterns yet. Typically, the built-in early termination (no improvement over the last 5 generations / runs) will stop the training of our algorithm within 6 hours on our main development server (cf. Section 10.2).

Training time

In this time we find about 500 graph patterns with a score > 2 (cf. Section 8.3), which are subsequently clustered into 100 result patterns to reduce the amount of queries necessary for prediction (cf. Section 9.2).

The 100 result patterns GP_r typically achieve a normalised max precision vector sum ($\text{nmpvs}(GP_r)$) of $\approx 62\%$ and an overall coverage of $\approx 76.5\%$ of the training set $\mathcal{GT}_{\text{train}}$ (cf. Section 7.3, Section 9.1).

Coverage

Afterwards, depending on the selected fusion variants, the fusion training (cf. Section 9.5) will typically take another 2 hours forming the final prediction model of our approach.

10.4.4 *Notable Learned Graph Patterns*

Before evaluating the resulting full system in Section 10.4.5, we will for brevity reasons mention the three most notable patterns from the resulting learned patterns. We invite the reader to explore the full results online[21] with the interactive visualisation presented in Section 8.9. The three patterns that we especially want to highlight are::

```
?source gold:hypernym ?target
```

```
?source dbo:wikiPageWikiLink ?target.
?target dbo:wikiPageWikiLink ?source.
```

```
?source dbo:wikiPageWikiLink ?target.
?v0 skos:exactMatch ?v1.
?v1 dbprop:industry ?target.
```

The first two are intuitively understandable patterns which typically are amongst the top patterns. The first one shows that human associations often seem to be represented via `gold:hypernym` in DBpedia (the response is often a hypernym (broader term) for the stimulus). The second one shows that bidirectional Wikilinks contain a lot of information for the simulation of associations. While we have observed this several times before in this work (e. g., in Section 6.3 and Section 10.4.2.7), our graph pattern learner is able to reliably identify this pattern without any prior information within the first of its runs.

Hypernyms

Wikilinks

The third pattern is mentioned, as it represents a whole class of *Intra-dataset learning*. The pattern makes use of a connection of the `?target` to BabelNet's `skos:exactMatch`. It is thereby able to use information from *all datasets* (cf. Section 10.2) and shows that our algorithm is able to use more knowledge from other datasets as it is loaded into our local Linked Data mirror.

Intra-dataset learning

10.4.5 *Full System Evaluation (Prediction & Fusion)*

As human associations are not readily modelled in DBpedia, it is difficult to assess the quality of the learned patterns gp directly. Hence, we evaluate the quality indirectly via their prediction quality on the test set $\mathcal{GT}_{\text{test}}$.

As explained in Section 10.3, for each of the $(s,t) \in \mathcal{GT}_{\text{test}}$ we can use its stimulus s as given source for our algorithm, generate target candidates $TC_{GP_r}(s)$ (cf. Section 9.3) with it, and fuse the so generated target candidates into a single ranked prediction list via a fusion variant $\text{ranking}(\text{fusion}_{\text{variant},GP_r}(s))$ (cf. Section 9.5). We can then determine the rank of the true target t in this ranked prediction list with $\text{rank}_{\text{fusion}_{\text{variant},GP_r}(s)}(t)$.

21 https://w3id.org/associations

Doing this for all $(s,t) \in \mathcal{GT}_{\text{test}}$ we can compute the aforementioned quality metrics (cf. Section 10.3) for each of the basic and advanced fusion variants described in Section 9.5.

An example of a ranked target prediction list (for the fusion method *precisions*) for source $s = \texttt{dbr:Sled}$ is the ranked list:

$$\text{ranking}(\text{fusion}_{\text{precisions},GP_r}(\texttt{dbr:Sled})) = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{matrix} \begin{pmatrix} \texttt{dbr:Snow} \\ \texttt{dbr:Christmas} \\ \texttt{dbr:Deer} \\ \texttt{dbr:Kite} \\ \texttt{dbr:Transport} \\ \texttt{dbr:Donkey} \\ \texttt{dbr:Ice} \\ \texttt{dbr:Ox} \\ \texttt{dbr:Obelisk} \\ \texttt{dbr:Santa\_Claus} \end{pmatrix}$$

In this case, the ground truth target $t = t_1 = \texttt{dbr:Snow}$ is at rank $\text{rank}_{\text{fusion}_{\text{precisions},GP_r}(\texttt{dbr:Sled})}(\texttt{dbr:Snow}) = 1$. As we can see most of the results are relevant as associations to humans. Nevertheless, for the purpose of our evaluation, we will only consider the single target t corresponding to a source s from our ground truth $(s,t) \in \mathcal{GT}_{\text{test}}$ as relevant and all other t_i as irrelevant.

Based on the ranked result lists, we can calculate the Recall@k[22], MAP (in our case equal to the MRR) and NDCG as explained in Section 10.3.

As we have performed many full trainings of our whole algorithm, for each fusion variant, we selected the model with the highest reported MRR on the training set $\mathcal{GT}_{\text{train}}$ and report its quality metrics over the whole test set $\mathcal{GT}_{\text{test}}$ in Table 10.2 and Figure 10.1.

Outperformed all baselines!

As we can see, our basic fusion variants already outperform all baselines by a factor of nearly 2: While the best baseline, our adaptation of the Milne-Witten Relatedness (*mw wl top5overlap*), achieves an MRR of 22% and an NDCG of 25.6%, our best basic fusion variant *gp precisions* achieves an MRR of 40.7% and NDCG of 48.9%. Looking at a Recall@1 of 29.2% it further outperforms even the reported performance of the best text corpus based baselines *(text rapp)* with 27% and comes close to average human top response prediction performance of 34.1%. Furthermore, *gp precisions* achieves a Recall@10 of 63.9%, which even outperforms all of our advanced fusion variants.

Some of the advanced fusion methods however manage to increase the MRR up to 43.3% and the NDCG to 50%. Amongst the best working fusion variants are the *logistic regression* and *random forest* classifiers, *ARD regression*, and the *RankSVM*. Looking at the Recall@1, we can actually notice that *ARD regression* and the *RankSVM* with 34.7%

22 We do not report Precision@k, as it degenerates to Recall@k/k due to the fact that we only have 1 relevant target per result of any $(s,t) \in \mathcal{GT}$.

Table 10.2: Comparison of Fusion Variants. (All values in percent.)

				Recall@k							
	method	MRR	NDCG	1	2	3	5	10	25	50	100
gpl basic fusion	target occs	34.6	44.2	22.2	36.1	41.7	47.2	56.9	68.1	72.2	**80.6**
	scores	33.6	41.4	22.2	33.3	38.9	47.2	58.3	61.1	65.3	69.4
	f measures	37.7	46.4	23.6	40.3	**48.6**	**54.2**	61.1	68.1	**75.0**	77.8
	gp precisions	**40.7**	**48.9**	**29.2**	**41.7**	45.8	**54.2**	**63.9**	**70.8**	**75.0**	79.2
	precisions	37.2	45.5	26.4	36.1	41.7	52.8	56.9	68.1	**75.0**	76.4
	scores precisions	31.7	41.4	22.2	26.4	33.3	41.7	56.9	**70.8**	72.2	77.8
	f measures precisions	34.1	43.3	22.2	33.3	40.3	48.6	59.7	66.7	72.2	77.8
	gp precisions precisions	35.2	44.0	26.4	34.7	37.5	40.3	54.2	**70.8**	**75.0**	77.8
gpl advanced fusion	adaboost	32.3	42.3	19.4	34.7	38.9	45.8	55.6	68.1	73.6	**80.6**
	decision tree	26.4	36.3	15.3	26.4	27.8	36.1	54.2	62.5	68.1	73.6
	gtb	40.4	48.6	30.6	38.9	44.4	54.2	61.1	70.8	75.0	79.2
	knn	24.3	34.4	13.9	20.8	25.0	36.1	52.8	61.1	68.1	72.2
	logistic regression	42.1	**50.0**	30.6	44.4	47.2	54.2	**62.5**	70.8	**76.4**	79.2
	naive bayes	30.1	39.3	16.7	33.3	36.1	45.8	55.6	63.9	69.4	72.2
	neural net	36.4	45.5	25.0	33.3	43.1	52.8	59.7	72.2	75.0	79.2
	qda	30.6	40.5	20.8	29.2	33.3	38.9	51.4	63.9	72.2	79.2
	random forest	**43.3**	**50.0**	33.3	**45.8**	**50.0**	52.8	59.7	65.3	70.8	75.0
	sgd	37.2	45.9	26.4	38.9	43.1	48.6	56.9	68.1	73.6	79.2
	svm linear	39.6	48.1	26.4	41.7	**50.0**	**55.6**	**62.5**	72.2	73.6	79.2
	svm rbf	26.9	37.5	15.3	25.0	27.8	38.9	56.9	68.1	73.6	76.4
	ada boost r	39.4	47.5	29.2	38.9	43.1	52.8	61.1	69.4	73.6	77.8
	ard r	**43.2**	**49.3**	**34.7**	44.4	**50.0**	51.4	59.7	61.1	68.1	72.2
	bayesian ridge	35.2	44.4	23.6	34.7	38.9	50.0	61.1	66.7	**76.4**	79.2
	decision tree r	23.2	34.2	11.1	22.2	30.6	37.5	43.1	56.9	72.2	77.8
	elastic net	27.0	36.2	15.3	27.8	33.3	40.3	45.8	62.5	65.3	70.8
	gradient boosting r	39.3	46.6	30.6	37.5	41.7	47.2	56.9	69.4	73.6	73.6
	kernel ridge	35.9	45.0	23.6	36.1	41.7	51.4	61.1	66.7	**76.4**	79.2
	kneighbors r	38.9	46.0	30.6	36.1	43.1	51.4	56.9	65.3	70.8	72.2
	lars	34.3	42.2	23.6	33.3	41.7	47.2	55.6	56.9	66.7	72.2
	lasso	29.7	36.7	22.2	31.9	34.7	36.1	41.7	45.8	54.2	66.7
	lasso lars	5.8	17.9	0.0	0.0	4.2	8.3	23.6	36.1	58.3	70.8
	linear r	31.1	39.8	20.8	31.9	36.1	41.7	48.6	61.1	68.1	73.6
	mlp r	35.1	44.0	25.0	31.9	38.9	50.0	56.9	66.7	73.6	77.8
	random forest r	35.8	45.1	20.8	38.9	45.8	51.4	**62.5**	**75.0**	75.0	77.8
	ridge	39.6	47.8	27.8	41.7	48.6	51.4	58.3	70.8	73.6	77.8
	sgd r	36.0	44.8	26.4	36.1	37.5	48.6	55.6	63.9	73.6	79.2
	svr linear	35.4	43.9	23.6	34.7	43.1	51.4	58.3	66.7	69.4	75.0
	svr rbf	35.4	43.9	23.6	34.7	43.1	51.4	58.3	66.7	69.4	75.0
	rank svm	42.4	**49.9**	**34.7**	41.7	44.4	51.4	58.3	66.7	73.6	79.2

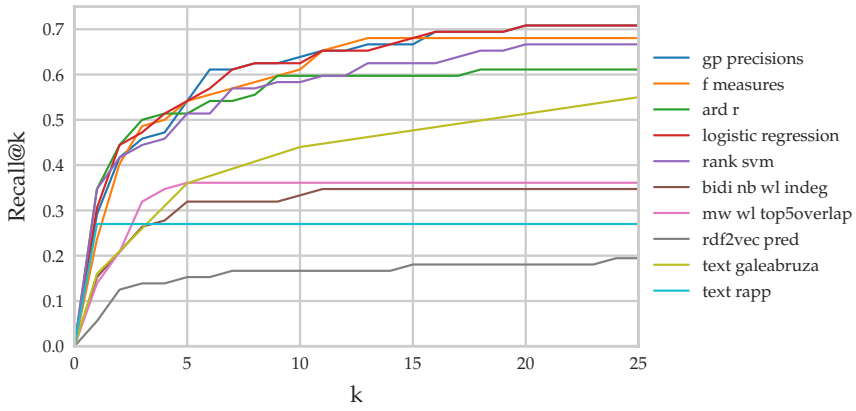


Figure 10.1: Recall@k of Selected Fusion Variants and Baselines

Human-like performance!

slightly outperform even *average human top response prediction performance* (34.1%), closely followed by random forests with 33.3%.

We can further observe that our fusion strategies reach a maximal Recall@100 of 80.6%, which is slightly higher than expected from the observed coverage based on our training set (cf. Section 10.4.3), further indicating that our approach did not over-fit to the training ground truth pairs.

10.4.6 *Analysis of Rank-Degree Correlations*

After this main result, we were further interested in which source-target pairs our algorithm can cover well and which not.

We were especially interested in the performance w. r. t. the node degrees of our source-target nodes. Hence, we plotted the degrees of our source and target nodes $(s,t) \in \mathcal{GT}_{\text{test}}$ over the achieved prediction ranks for t, as can be seen in Figure 10.2 and calculated their correlations, as listed in Table 10.3.

Largely degree independent

Visually, it seems as if the prediction ranks of our graph pattern learner and its coverage are largely independent of the source-target node degrees. We however notice a slight preference of the true target ever being found for sources with high in-degrees and targets with high in-degrees.

When computing correlations (including degrees of such source-target pairs (s,t) where t is not included in our algorithms predictions and has rank 0), we cannot identify a strong reliable correlation between ranks of the source-target pairs and our prediction result ranks. If at all, then there is a weak correlation between the source node out-degrees and the achieved prediction ranks of the true target.

When ignoring source-target pairs (s,t) where the target is not included in the predictions of our algorithm, we find a significant (but

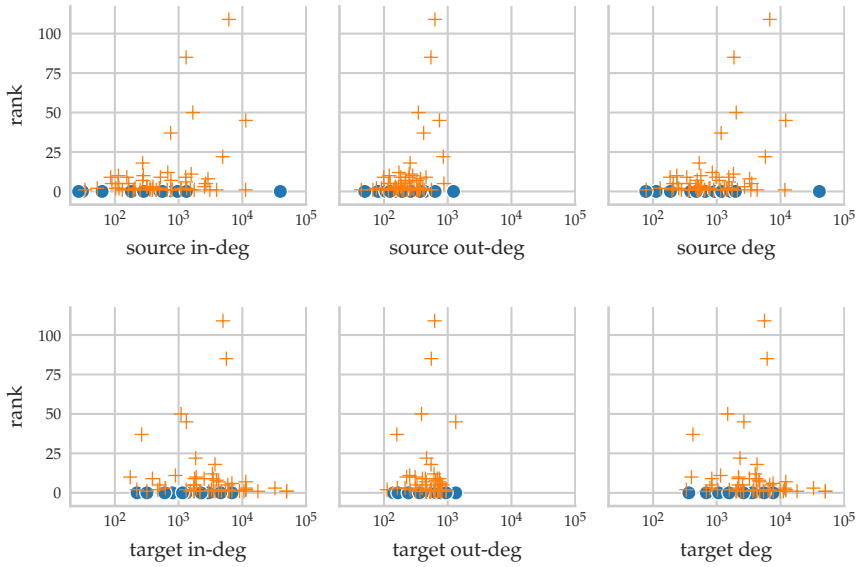


Figure 10.2: Node Degrees vs Ranks: Plotted are the node degrees of source and target nodes of the ground truth test set pairs against the true target ranks in the fused prediction lists. In case the true target was not within the predictions, they are marked as blue dots (rank 0). As can be seen, there is no obvious correlation.

Detected targets: weak degree-rank correlation

weak) positive correlation between source node degrees (especially out) and our algorithm's predicted ranks. At the same time, we find a significant (but weak) negative correlation between target node degrees (especially in) and our algorithm's predicted ranks. Such correlations intuitively make sense, as they mostly describe the growth of the search space and thereby difficulty of the problem for prediction: Source nodes with a high degree are connected to a lot of potential targets. At the same time, targets with a high degree can be seen as easier to find and more important.

10.4.7 *Prediction Quality Spread Evaluation*

After the main results of our algorithm and the analysis of degree-rank correlations, we were further interested in the reliability of our algorithm. After all, our algorithm is non-deterministic, as it identifies good graph patterns with an evolutionary approach that uses chance levels in all of its major components.

To analyse the reliability of our overall approach, we decided to execute the full algorithm many times[23] and aggregate the achieved

23 More than 1100 times for the basic fusion variants and more than 370 times including full training of all advanced fusion methods.

Table 10.3: Node Degree vs Rank Correlations. Correlations on top are calculated with missing true targets as rank 0. On the bottom we ignore such source-target pairs. Values show the correlation coefficient between node degrees (any-, in-, or out-degree) and predicted rank of the true targets over the ground truth test set. The p-values in brackets denote the probability of a random, uncorrelated system to produce at least the correlation coefficient shown.

Missing target as rank 0:

node	degree	Pearson	Spearman	Kendall's Tau
source	any	-0.003 (0.978)	0.191 (0.107)	0.139 (0.099)
source	in	-0.012 (0.921)	0.155 (0.192)	0.111 (0.188)
source	out	0.202 (0.088)	**0.237 (0.045)**	0.174 (0.040)
target	any	-0.059 (0.625)	-0.120 (0.315)	-0.077 (0.363)
target	in	-0.058 (0.629)	-0.119 (0.321)	-0.078 (0.358)
target	out	-0.145 (0.225)	-0.075 (0.532)	-0.050 (0.559)

Ignoring source-target pairs in case of not predicted target:

node	degree	Pearson	Spearman	Kendall's Tau
source	any	0.390 (0.003)	0.190 (0.165)	0.134 (0.173)
source	in	0.371 (0.005)	0.149 (0.277)	0.105 (0.287)
source	out	**0.477 (0.000)**	0.257 (0.058)	0.190 (0.053)
target	any	-0.081 (0.559)	-0.293 (0.030)	-0.208 (0.035)
target	in	-0.081 (0.558)	**-0.307 (0.023)**	-0.224 (0.023)
target	out	0.050 (0.716)	-0.174 (0.204)	-0.130 (0.187)

quality metrics MAP (in our case equal to MRR), NDCG and Recall@10 on the test set over all of the executions[24].

The results can be found in the box-and-whisker plots in Figure 10.3, Figure 10.4, and Figure 10.5. The box shows the Inter-Quartile Range (IQR), so the middle 50%. The whiskers extend to the smallest/largest point up to $1.5 \cdot \text{IQR}$ away from the box. All other points are regarded as outliers and shown as dots.

As we can observe, our prediction results typically have MRRs within an IQR of $\approx$ 8% in between 20% and 40%. The NDCGs typically have IQRs of $\approx$ 5% in between 30% and 45%. The Recalls@10 typically have IQRs of $\approx$ 7% in between 45% and 60%.

Winner: logistic regression

The best of our fusion methods (with respect to quartile 3 and mean) is the *logistic regression* classifier, which is also among the most stable (small IQR) of all advanced fusion methods. It typically is able to achieve a Recall@10 in between 54% and 59%.

The most successful basic fusion method is *gp precisions* with a Recall@10 between 52% and 58%.

Despite spread, it is very rare, that our better fusion methods drop below the best baseline's (Milne-Witten *mw wl top5overlap*) achieved MRR of 22%.

24 Note, that while we report the spread on test set here, we did select the best model reported in Section 10.4.5 based on its training set performance.

Hence, for practical purposes, and depending on the desired investment of computation time, we suggest to either rely on the basic fusion method *gp precisions*, or on one of the more stable advanced fusion approaches such as *logistic regression*.

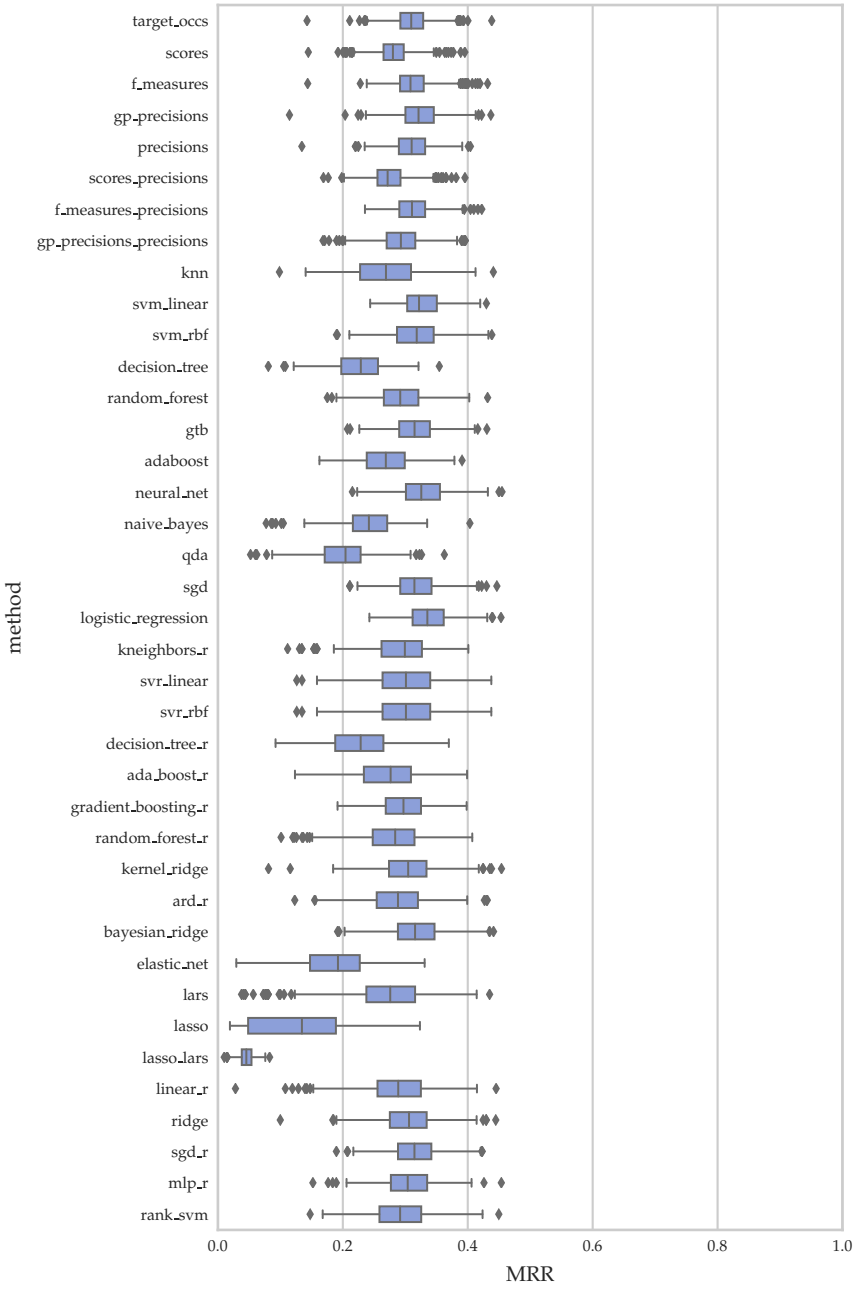


Figure 10.3: Fusion Method Comparison: MRR. Plotted are the distributions over the ground truth test set's targets. The basic fusion methods are aggregated over more than 1100, the remaining over more than 370 full training and subsequent test cycles of the algorithm.

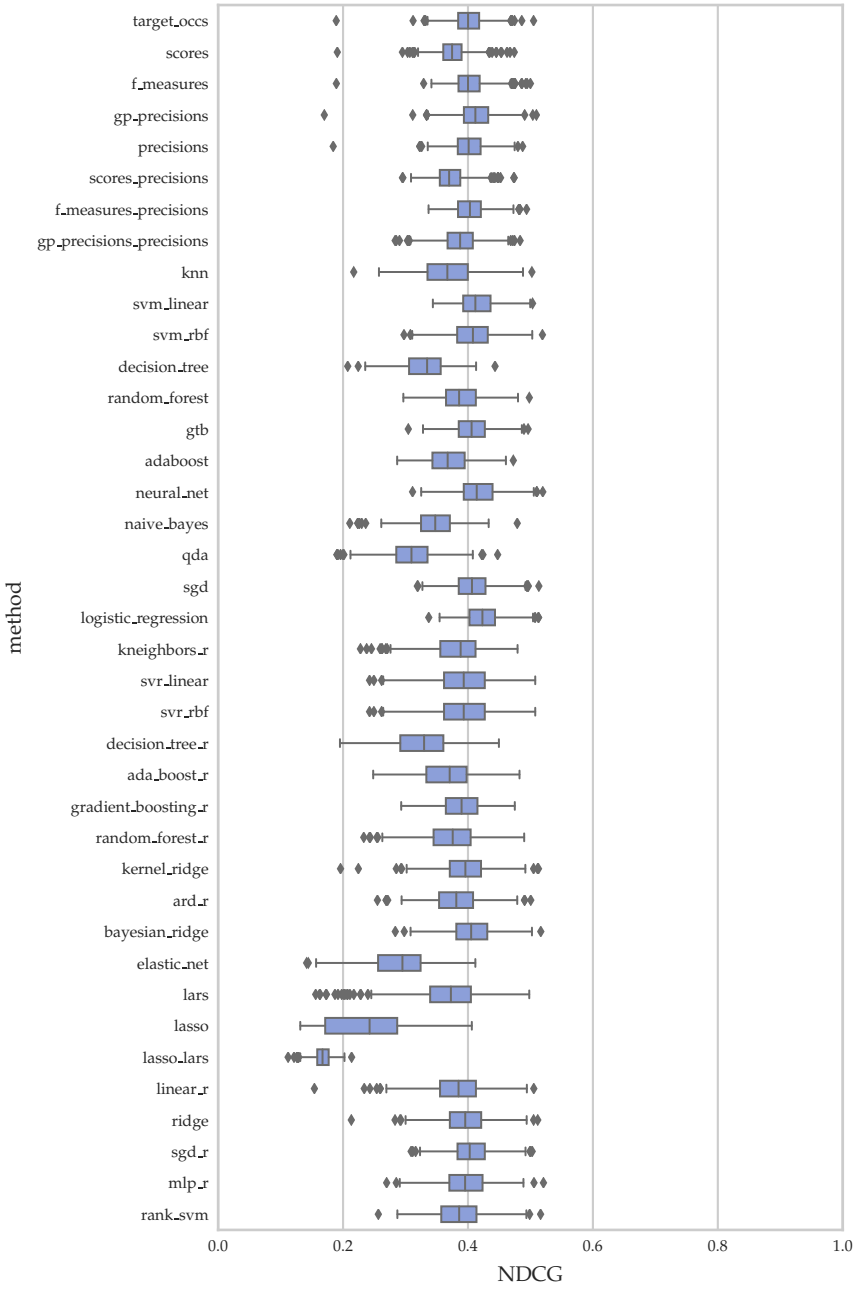


Figure 10.4: Fusion Method Comparison: NDCG. Plotted are the distributions over the ground truth test set's targets. The basic fusion methods are aggregated over more than 1100, the remaining over more than 370 full training and subsequent test cycles of the algorithm.

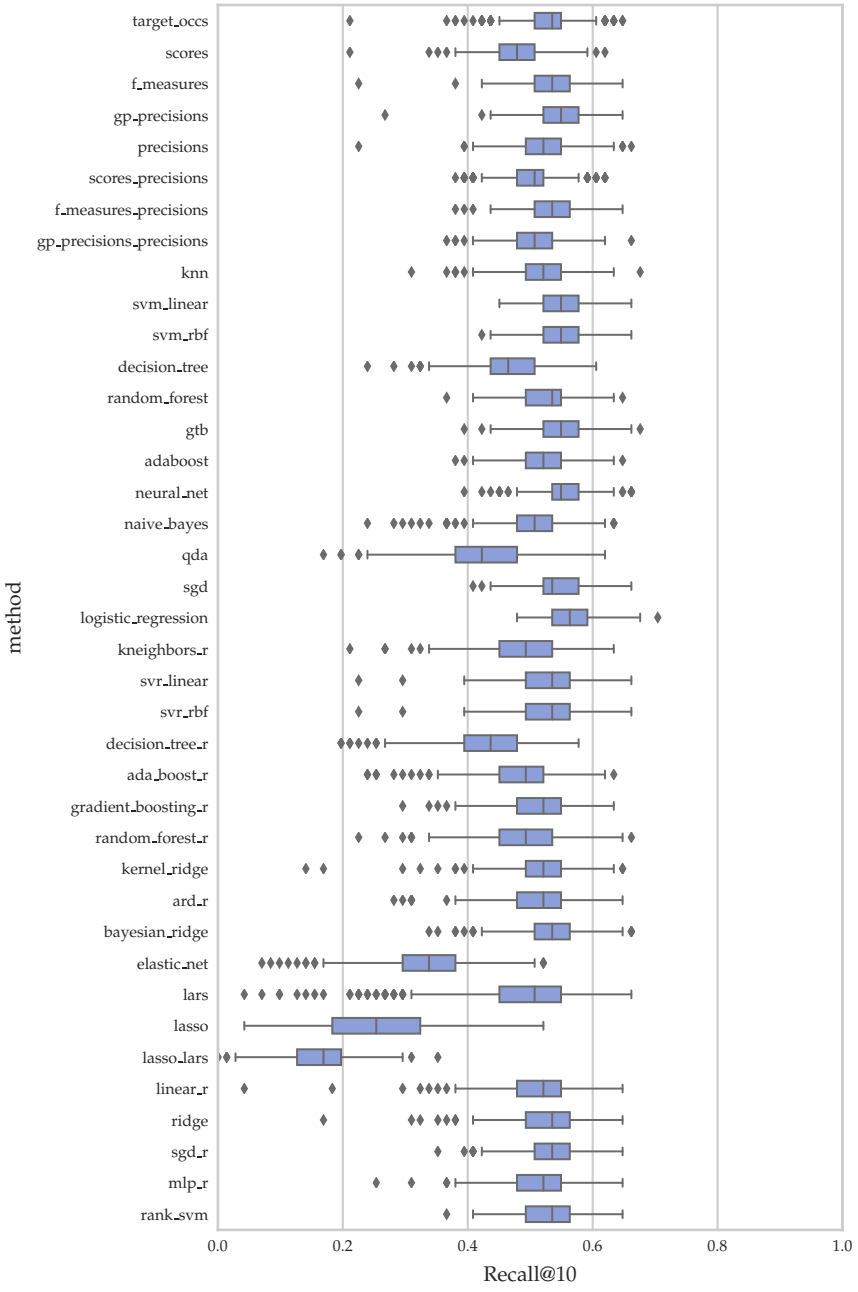


Figure 10.5: Fusion Method Comparison: Recall@10. Plotted are the distributions over the ground truth test set's targets. The basic fusion methods are aggregated over more than 1100, the remaining over more than 370 full training and subsequent test cycles of the algorithm.

10.5 PATTERN INJECTION

After the main evaluation of our approach in the previous section, we now know that our approach is well suited for the simulation of human associations (our primary goal), and even achieves human-like Recall@1. However, in this section, we want to expand our scope and evaluate the more general overall capability of our graph pattern learner to find patterns of a certain complexity/difficulty. In other words, we want to investigate the practical limits of what our pattern learner can learn.

Main idea

The main idea behind our approach to analyse this is simple: We will generate synthetic patterns of certain complexities (as described below). We will then inject instantiations of these patterns into an existing (local) knowledge base. Finally, we will check if our graph pattern learner is able to detect them.

Challenges

While this idea is simple, its implementation and execution is not. As we have already seen in Section 7.3.2, the amount of patterns (even of structurally different ones) is staggering. Even for a moderate size of complete (including a `?source` and `?target`) variable-only patterns with up to 4 triples we already have nearly 10.9k structurally different isomorphism classes of patterns. For each of them, we have to perform a (potentially full) training cycle of our graph pattern learner with an isolated and modified local SPARQL endpoint. Additionally, as our algorithm is non-deterministic, we should (if computationally feasible) run several of such training cycles per injected pattern. Hence, in order to make this feasible at all, we rely on the cluster setup described in Section 10.2.2.

Injecting a graph pattern

Further, there are many possible ways to "inject" the patterns into a knowledge base. To have practical relevance, we decided against generating a purely synthetic knowledge base, but instead use the DBpedia core dataset G_{core} (cf. Section 10.2.1) image[25]. Before injecting an instantiation of the graph pattern in question, we initially drew 2000 fully random DBpedia entities, and combined them into 1000 random source-target pairs $\mathcal{GT}_{random}$. Using random node pairs in this way, has the advantage of providing similar distractions in form of out- and in-links to our algorithm as real-world use-cases. We then checked that (as expected) our graph pattern learner is not able to detect a good pattern modelling these 1000 random pairs. After this, for the graph pattern gp_i in question, we instantiate all of its variables with generated URIs for each of the 1000 random source-target pairs[26] and inject the so formed triples into our local

Random node pairs

Instantiation

25 We selected this dataset instead of the all datasets due to its reduced RAM requirement, allowing us to scale our experiment to nearly all of the cluster's compute nodes.

26 To resemble real world patterns, we decided to generate one static URI for each of the predicate variables in gp_i, that is shared over all source-target pairs. The remaining variables (for nodes) are dynamically instantiated per source-target pair.

endpoint via SPARQL UPDATE queries, forming the modified local knowledge base G'_{core}. Finally, we execute our graph pattern learner on the random source-target pairs and the modified local endpoint $gpl(\mathcal{GT}_{random}, G'_{core})$. To save computation time, we terminate the algorithm as soon as the normalised max precision coverage vector sum reaches $nmpvs(gpl(\mathcal{GT}_{random}, G'_{core})) = 1$ and deem the experiment for gp_i successful. If the algorithm terminates without reaching full coverage, we deem the experiment for gp_i unsuccessful.

In the following, we will focus on two different ways to define the complexity of a pattern.

10.5.1 *Path Length Evaluation*

A simple approach to define the complexity of a graph pattern is by focusing on the distance of `?source` and `?target`. Intuitively, the further apart the target from the source, the harder it should be to identify a pattern.

path patterns

To evaluate how well our graph pattern learner deals with certain distances, we generated random direction *path patterns* of length l. For this, we first generate patterns of the following form:

$$\texttt{?source } ?p_1\ ?n_1\texttt{ . ... } ?n_i\ ?p_{i+1}\ ?n_{i+1}\texttt{ . ... } ?n_{l-1}\ ?p_l\texttt{ ?target.}$$

and subsequently flip each of its triples with a 50% chance. It is easy to see that for a given length l there are 2^l structurally different patterns.

For our evaluation, per length $l \in \{1, \ldots, 10\}$ we ran n experiments. Each experiment consists of resetting the database, forming a new randomly directed path pattern of length l, injecting its instantiation (as described above) and checking whether the injected pattern could be learned. The average success rates over these n experiments can be found in Table 10.4. We further ran our algorithm in the default configuration, as well as an extended configuration, in which it early terminates after no improvements over the last 10 generations and runs (default: 5).

Table 10.4: Injection Path Length Evaluation

configuration										
l	1	2	3	4	5	6	7	8	9	10
default										
success rate	100%	100%	68.2%	22.2%	0.3%	0%	0%	0%	0%	0%
n	2000	2000	2000	2000	1000	1000	1000	500	500	100
extended										
success rate	100%	100%	77.1%	44.8%	17.3%	0%	0%	0%	0%	0%
n	2000	2000	1500	500	350	300	300	200	150	150

As we can see, our algorithm is able to learn all path patterns of lengths up to 2 and significant amounts of patterns up to length 4

and even 5. While at first sight this might not seem impressive, we remind the reader that we are evaluating on the DBpedia core dataset, on random source-target pairs that are DBpedia nodes. Each of these nodes have many in- and outgoing triples. Especially, each of these node pairs is connected with a path of length 2:

```
?source a owl:Thing. ?target a owl:Thing.
```

While not suitable for prediction, the pattern (and similar more DBpedia specific ones) introduces significant amounts of distraction in form of candidates for exploration to our algorithm. Hence, we are actually surprised, how well our algorithm in its current form already deals with paths of up to length 5.

However, as parts of knowledge bases are sometimes modelled in a linear fashion, we already plan to introduce another special mutation kind for longer paths with a limited degrees, as will be mentioned in Chapter 13.

10.5.2 *Enumeration Based Evaluation*

Another way to define complexity is by the amount of triples in a pattern $l = |gp|$. However, as we saw in Section 7.3.2, the number of such patterns grows strongly exponential. Hence, we will focus on the structurally different isomorphism classes of patterns, which we were able to enumerate up to length 4.

For our evaluation, for a given $l \in \{1, \ldots, 4\}$ we form one experiment[27] out of each of its n structurally different enumerated *canonical graph patterns*. Each experiment consists of resetting the database, injecting an instantiation of its graph pattern (as described above) and testing whether our graph pattern learner is able to find the pattern. The average success rates over all n experiments can be found in Table 10.5.

canonical graph patterns

Table 10.5: Injection Pattern Enumeration Evaluation

l	1	2	3	4
success rate	100%	100%	93.2%	91.5%
n	2	28	486	10374

We can see that our algorithm is able to detect all 30 patterns of length 1 and 2, and more than 90% of all nearly 11k patterns of lengths 3 and 4. While we are already satisfied with the achieved success rates, especially given that each pattern was only tested once, we plan to further investigate the remaining isomorphism classes of patterns in the future.

27 We decided against sampling or several full cycles of our approach, as the amount of patterns of length 4 is already quite large.

10.6 COMPARISON TO KORE ENTITY RELATEDNESS RANKINGS

After the main evaluation of our approach w. r. t. human associations and the pattern injection evaluation in the previous sections, we want to conclude this chapter with a short comparison of our fully trained human association model with the KORE entity relatedness ranking dataset.

As already mentioned in Section 3.1, the KORE entity relatedness ranking dataset [93] consists of 21 source entities from 5 different categories (IT companies, Hollywood celebrities, video games, television series, and the singleton category Chuck Norris). For each of the 21 source entities, a ranked list of 20 selected linked target entities was created via a crowd-sourcing experiment in which all pairwise relatedness comparisons of each of the linked entities were performed by 5 participants each.

To compare against the KORE entity relatedness approaches, for each of the source entities, we use our EAT based model to predict a ranked list of target candidates. We then compare our ranking to the KORE ground truth ranks with the Spearman correlation coefficients, to be comparable to the results reported in the original KORE paper [93]. In the process, we ignore targets predicted by our model that are not in the KORE ground truth lists and treat missing KORE target entities as last item of our predictions (with rank 101). We report the resulting correlations of our best two fusion variants in Table 10.6 as model "eat".

As we can see, our association simulating model is clearly outperformed on the entity relatedness ranking task by the KORE baselines, which reach a Spearman correlation with the rankings of their ground truth dataset of 67.3%. In comparison to this, we only reach 33.6%. We however noticed that our association ground truth contains nearly no examples of any of the given categories, which means that our EAT based model has rarely (if ever) seen a Hollywood celebrity (and Chuck Norris), IT company, TV series, or video game source node and corresponding target during its training.

Problematic domain change

Hence, in order to provide a fairer comparison, we decided to re-train a model on the KORE ground truth dataset. As the KORE ground truth only consists of 21 different sources, we decided to perform a 21-fold leave-one-out training-test set split: For each of the 21 source entities, we train a model consisting of the other 20 source entities and their corresponding top-1, top-3 or top-5 assigned ranked targets as training dataset. We then predict the target candidates of the missing 1 source and compare the results to the KORE ground truth. We report the achieved correlations and denote these models as top-1, top-3 and top-5 in Table 10.6.

As we can see, the performance of our graph pattern learner improves significantly, even out-performing one of the KORE baselines

Table 10.6: Comparison with KORE. (All values in percent.)

Spearman correlations:

model	fusion method	Chuck Norris	Hollywood celebrities	IT companies	TV series	Video Games	All
KORE		58.5	64.6	76.4	51.9	78.0	67.3
$KORE_{LSH\text{-}G}$		58.5	64.7	58.6	53.8	72.2	62.1
$KORE_{LSH\text{-}F}$		65.3	52.2	20.8	42.6	49.9	42.5
eat	mlp r	53.7	51.8	26.3	35.8	16.5	33.6
	kernel ridge	63.9	40.0	26.1	32.0	12.5	29.4
top-1	target occs	55.3	47.7	50.4	60.4	33.1	48.2
	scores	48.6	48.3	47.7	60.6	31.2	47.0
top-3	gtb	51.0	44.1	59.8	58.9	50.6	53.2
	sgd	24.2	46.4	56.7	56.8	48.2	50.7
top-5	rank svm	50.5	36.7	56.1	49.7	48.9	48.0
	mlp r	58.2	44.9	41.1	52.9	43.8	46.3

NDCGs:

model	fusion method	Chuck Norris	Hollywood celebrities	IT companies	TV series	Video Games	All
eat	ada boost r	75.1	72.3	49.0	53.8	55.8	58.6
	target occs	76.7	65.4	47.9	45.2	55.8	54.7
top-1	naive bayes	74.0	61.1	71.0	69.0	69.1	67.9
	logistic regression	70.8	71.6	68.1	66.5	62.2	67.3
top-3	naive bayes	76.4	66.3	63.5	72.8	65.1	67.4
	target occs	68.0	64.8	69.2	70.7	64.6	67.3
top-5	sgd	80.7	61.7	63.2	69.4	64.5	65.5
	svm linear	69.9	65.4	62.0	68.2	65.1	65.4

with achieved Spearman correlations of up to 53.2% for the top-3 model. This indicates that given more of the right training data, our algorithm will likely also be able to be an interesting alternative approach for relatedness rankings.

Strong improvement after re-training

In contrast to the KORE ranking algorithm, we however want to stress that our approach is designed to solve a different problem. Unlike KORE it is not given a list of 20 target candidates to rank for each of the 21 given source entities. Instead, it is only given the 21 source nodes and has to solve an open target prediction problem by generating the target candidates itself and ranking them. For this reason, we also report the achieved NDCGs in comparison to the KORE ground truth rankings, as they are better suited for evaluations of only partially overlapping lists of target candidates.

KORE: fact ranking

GPL: node prediction

11 OTHER APPLICATIONS

While our evolutionary graph pattern learner is primarily designed to solve the problem of predicting human associations, it is applicable to other scenarios in which one would like to train a machine learning model on a knowledge base given a simple list of source-target pairs.

To conclude Part III, in this chapter, we briefly show the application of our algorithm to two such other use-cases: In Section 11.1 we will apply the graph pattern learner to the DBpediaNYD entity relatedness dataset. In Section 11.2 we will train a model based on data of the recommender engine TasteDive.

The achieved MRRs, NDCGs and Recalls@k for the two applications are summarised in Table 11.1. We only report the selected best variants for each of the use-cases.

Table 11.1: Other Applications. (All values in percent.)

Use-Case			Recall@k							
method	MRR	NDCG	1	2	3	5	10	25	50	100
DBpediaNYD:										
gp precisions	**31.1**	**42.6**	**21.0**	25.0	29.0	**43.0**	61.0	**75.0**	**84.0**	**86.0**
precisions	23.4	36.5	10.0	19.0	23.0	40.0	62.0	74.0	81.0	85.0
target occs	29.1	41.2	16.0	**26.0**	**31.0**	42.0	**63.0**	**75.0**	**84.0**	**86.0**
TasteDive Books:										
f measures	29.3	36.2	20.1	**29.9**	33.6	37.3	48.5	56.7	59.0	61.2
precisions	25.3	33.7	11.9	23.9	**34.3**	41.8	**56.0**	**59.0**	**61.9**	**61.9**
scores precisions	**30.0**	**36.7**	**21.6**	29.1	33.6	39.6	49.3	56.0	59.0	61.2
target occs	26.3	34.3	14.2	24.6	32.8	**42.5**	53.7	58.2	59.7	61.2

11.1 ENTITY RELATEDNESS: DBPEDIANYD

As mentioned in the Section 3.1, the DBpediaNYD dataset [137] consists of the calculated symmetric and asymmetric Normalised Yahoo Distances (adaptations of the Normalised Google Distance [38]) between subject and object labels of about 7000 randomly drawn DBpedia facts.

From these 7000, we selected the top 1000 source-target pairs w. r. t. their asymmetric relatedness scores and provided them to our graph pattern learner as ground truth dataset. Given this different ground truth data, we hope that the graph pattern learner is able to identify patterns for semantic relatedness (as computed via the external Yahoo service) directly from the knowledge base (*all datasets*).

Analogue to the previous experiments, we performed a 9:1 training-test set split, trained our algorithm on the new training set and report the results on the test set in Table 11.1. To save computation time, we skipped the computations of advanced fusion methods and solely focused on the basic fusion variants, out of which we report the top-3.

As in our association use-case, the *gp precisions* achieves the best results w. r. t. MRR and NDCG with an MRR of 31.1% and an NDCG of 42.6%. We further reach a Recall@1 of 21% and with *target occs* a Recall@10 of 63%.

With the achieved prediction performance, our approach might be an interesting alternative to predict related nodes without relying on external services like search engines. Unlike many other relatedness approaches, which (like the NYD) depend on a list of already generated target candidates that can then individually be scored w. r. t. the given source node, our approach will handle both: target candidate generation and scoring.

11.2 RECOMMENDER ENGINE: TASTEDIVE

As a second use-case, we decided to train a model based on the online recommendation website TasteDive[1], which is an "if you like this, you might also like that" recommender engine: It allows users to enter a thing they like and then recommends a list of things they might also like. The recommendations seem to be learned and constantly improved from the user-profiles of its registered users.

We chose TasteDive as a use-case, as it provides an easy to use API[2] and as nearly all of its "things" provide a short description from and a link to a Wikipedia article. The latter allows us to easily represent all TasteDive "things" via their corresponding DBpedia URIs.

TasteDive further provides top-level categories for its "things": music, movies, tv shows, books, authors and games.

For our use-case, we focused on the books category. To form a ground truth recommendation source-target pair dataset, we crawled about 1350 book recommendation pairs as follows: We formed an initial seed of sources from trending and top books. For each source, we then retrieved the list of its recommendations. We note the first recommendation as target for the source, forming a source-target pair. All of the returned recommendations become new potential sources to crawl, while making sure not to repeat ourselves.

The resulting ground truth list for example contains the following source-target pairs: (`dbr:Beauty_and_the_Beast`, `dbr:The_Sleeping_Beauty_Quartet`), (`dbr:The_Hunt_for_Red_October`, `dbr:The_Cardinal_of_the_Kremlin`), and (`dbr:Gone_Girl_(novel)`, `dbr:Dark_Places_(novel)`).

1 https://tastedive.com
2 https://tastedive.com/read/api

Analogue to the previous use-case, we trained our algorithm on this new dataset, skipped the computation of advanced fusion methods, and report results in Table 11.1.

The best prediction performance on the TasteDive books use-case could be achieved by the *scores precisions* fusion method, resulting in an MRR of 30%, an NDCG of 36.7%, and a Recall@1 of 21.6%. With the fusion method *precisions* we further reach a Recall@10 of 56%.

This means that about 1 in 5 top-1 predictions and 1 in 2 top-10 predictions of our graph pattern learner actually matches/contains the top recommendation of TasteDive. Unlike TasteDive, our graph pattern learner does currently not make use of any information from user profiles, but solely uses the knowledge from our local SPARQL endpoint.

Hence, especially also for new users and new items, our approach might be very interesting as a complementary way to generate recommendations.

Summarising, in this part we have described our graph pattern learning approach and how the learned patterns can be clustered, used for prediction and their results fused into a single ranked list of predicted targets. We further evaluated our approach against many baselines, saw that it is able to achieve human-like performance for the simulation of human associations, and in this chapter applied it to other approaches. We will conclude this work with a overall summary and future work in Part IV.

Part IV

CONCLUSION

12

SUMMARY

In recent years, research in the field of machine learning has made a lot of progress. However, most approaches still do not utilise open knowledge in the form of Linked Data, as incorporating such knowledge into machine learning approaches is still a challenging and often manual task. Effectively using existing knowledge is however one of the requirements for Artificial General Intelligence (AGI) and general human-like thinking. Hence, in this work, we focused on a small part of such human-like thinking processes, namely human associations, and investigated how Linked Data can be used to simulate them.

For this, the two main goals of this work were, first, to generate a high-quality dataset of semantic associations, and second, to use the generated dataset to train a machine learning model that can simulate human associations on Linked Data.

DATASET GENERATION In order to reach the first goal, Part II of this work investigated three different approaches to generate a large, high-quality dataset of semantic associations (associations between semantic entities).

The first two of these approaches (Chapter 4) are Games With A Purpose (GWAPs), allowing us to make data-entry as much fun as possible. The first one, called *BetterRelations*, is a browser game that pairs its players into teams. In each round, the players are presented with the textual representation of two facts about an entity and asked which one of the facts will come to their partner's mind first. On agreement, the players earn points. Behind the scenes, the game uses a noise resistant sorting algorithm that we developed to allow us to rank existing facts about pre-determined entities by human association strengths, based on the pair-wise comparisons by the players. However, one of the short-comings of BetterRelations is its bias to facts that already existed in the knowledge base.

BetterRelations

Hence, we created a second game called *Knowledge Test Game*. This game resembles a Family Feud like setting: The players are shown a screen telling them that we asked 100 people to name something associated with a semantic entity. The players are asked to try and guess what they said via a text input field, which on entry live disambiguates the entered string to semantic entities from DBpedia. If the same entity was selected by others, they are rewarded with points. By this, the game quickly collects high-quality semantic associations between stimulus and response entities.

Knowledge Test Game

Semi-automatic mapping approach

While games allowed us to make data-entry as much fun as possible, they still involved a lot of human work. Hence, we also focused on a third approach in Chapter 5, that maps an existing psychological human association dataset, the Edinburgh Associative Thesaurus [101] (EAT), to DBpedia entities. EAT is a large collection of free-text associations (e.g., "cat - dog") that were collected directly form students. With a semi-automatic mapping approach we were able to map about 1000 distinct most frequently occurring stimulus-response pairs to their corresponding DBpedia entities. After each mapping was verified by 3 reviewers, we were able to collect a ground truth of 727 semantic associations corresponding to about 25.5k raw associations collected directly from humans.

Generated semantic association dataset

After a comparison of the three approaches in Chapter 6, we saw that the semi-automatic mapping approach was by far the most successful in collecting a large variety of semantic associations and allowed us to complete the first of our two main goals. Hence, we decided to focus the following efforts on its generated 727 distinct semantic associations as a ground truth dataset.

PATTERN LEARNING FROM LINKED DATA To fulfil the second main goal, in Part III we described an evolutionary algorithm, the so called *Graph Pattern Learner*. We developed it to be able to simulate human associations with Linked Data after training on the previously generated ground truth dataset.

Graph Pattern Learner

Pattern learner

Our algorithm (Chapter 8) is able to learn graph patterns in an end-to-end fashion. Given a ground truth list consisting of source-target pairs of semantic entities, it learns graph patterns from a given SPARQL endpoint without any further manual human intervention or feature engineering. Additionally, the algorithm is designed to scale to knowledge bases consisting of billions of triples.

Our approach, being an evolutionary algorithm, follows the general structure of such: populations composed of individuals that mate and mutate to form the next generation, with the fittest individuals having a higher chance to survive.

The individuals of our evolutionary algorithm are graph patterns (SPARQL BGPs) with at least a `?source` and `?target` variable. They can mate by exchanging triples, and mutate by introducing, deleting, flipping triples or substituting variables with entities from the SPARQL endpoint. Their fitness is evaluated against the given endpoint by performing a series of queries. Simply put, patterns are the fitter, the more ground truth source-target pairs they cover with high precision and low query evaluation times.

As the given ground truth list of source-target pairs is possibly modelled not only with a single kind of pattern in the knowledge base, our algorithm performs several runs. In each run, it re-focuses on the parts of the ground truth that are not already covered. In this

way, it reaches good overall coverage over the whole ground truth, independent of modelling in the knowledge base.

In order to reduce the computation times, our algorithm uses many sophisticated techniques such as batching, timeouts, fit to live filters, parallelisation, caching, error recursion and pattern simplification. The resulting learned graph patterns can be explored via an interactive visualisation (Section 8.9).

Prediction

Each of the learned graph patterns can be used for prediction of target candidates for a given (new) source node (Chapter 9). We explained how the learned patterns have a dual function as explainable generators and dimensions of a vector space. The vector space can be used to cluster similar patterns and to fuse the predictions of each individual pattern into a single ranked list.

Query clustering

Fusion

The resulting full machine learning model (consisting of the learned graph patterns and trained fusion variants) that was trained with help of our semantic association ground truth dataset can be tested in an interactive online demo and allows live predictions for human associations.

Evaluation

In Chapter 10, we extensively evaluated our algorithm's ability to simulate human associations. Comparing against a large variety of baselines, our approach outperforms all other baselines by a factor of nearly 2 by achieving an MRR (& MAP) of 43%, an NDCG of 50% and a Recall@10 of 63.9%. With a Recall@1 of 34.7% it outperforms not only the best text corpus based baseline, but also reaches human individual top response prediction performance. Given these evaluation results, we can safely say that our graph pattern learner fulfils the second main goal of our work and is able to simulate human associations with Linked Data.

Human-like performance!

We further investigated our algorithm's performance spread, its ability to learn graph patterns of varying complexities in general, and applied the human association based model (as well as a retrained one) to the KORE entity relatedness dataset.

Summarising, all goals of this work were reached:

- We created a large ground truth dataset of 727 human semantic associations backed by 25.5k raw associations collected directly from humans.
- We developed a novel machine learning approach that uses the generated dataset. It detects patterns for human associations in Linked Data, and uses them to simulate human associations. In the process, it reaches human-like top response prediction performance.

All generated datasets, the full source code of our approach, the interactive visualisations and demo, and further supplementary material are available online:

https://w3id.org/associations

Noteworthy byproducts of this work were the transformation of the full EAT dataset into easily accessible RDF form (Section 5.2) and a SPARQL BGP canonicalisation approach (Section 7.3.3). Further, we demonstrated that our graph pattern learner can be applied to and learn models for other scenarios, simply by exchanging its training data (Chapter 11), a fact that we plan to exploit further in the future, as mentioned in the following chapter.

13 FUTURE WORK

After the summary in the previous chapter, we will now conclude this work with a list of promising directions for future work, fitting into four main areas: extension of the semantic association dataset, improvements of our graph pattern learning algorithm, combination with other (complementary) approaches, and other application areas of our graph pattern learning algorithm.

EXTENDING THE SEMANTIC ASSOCIATION DATASET While the resulting dataset of 727 semantic associations is quite large already, it certainly does not cover every domain. This especially became apparent when applying our fully trained human association model to the KORE ranking use-case, which focuses on the domains IT companies, TV series, Hollywood celebrities and video games, in Section 10.6. While for many entities of these domains our model was still able to generate reasonable response predictions, none of the domains is represented in our training data. Hence, an obvious approach to improve the simulation of human associations is the collection of more semantic associations from such domains.

Cover more domains

As mentioned in Section 3.1, in this work we focused on the EAT dataset and mapped its strongest associations to DBpedia entities in Chapter 5. It would be beneficial to try and map other free-text association corpora, such as the University of South Florida free association, rhyme, and word fragment norms [131, 132] (USFA), to DBpedia entities with a similar approach. Additionally, to extend the amount of semantic associations, other mapping targets, such as Wikidata should be investigated.

Mapping of USFA

IMPROVING THE GRAPH PATTERN LEARNER Another way to improve (not only) the simulation of human associations is by improving our graph pattern learner.

In future, we want to investigate if the algorithm's results can be further improved by a smarter generation of the initial population. Promising candidates for this are approaches that try to find meaningful connections between nodes [42], extend SPARQL for extended path queries SPARQLeR [106], or find top-k shortest paths [92].

Smarter initial population

We also plan to enhance our algorithm to support RDF *Literals* in the input source-target pairs, which would allow us to learn patterns directly from input lists of textual pairs. If successful, such an approach would remove the necessity of prior entity disambiguation.

Support for Literals

LDF / HDT

We further will attempt to modify the GPL to learn from Linked Data Fragments [173] or directly from HDT [54], instead of relying on full SPARQL implementations. This would allow us to run our graph pattern learner on much larger excerpts of the LOD Cloud, such as LOD-a-lot [53]. Especially in combination with the aforementioned extension of our algorithm to directly support learning from pairs of Literals, this could make our algorithm applicable to a whole variety of other use-cases.

More SPARQL features

An alternative improvement idea is to use more advanced SPARQL query features such as `FILTER`, `UNION`, or even aggregations such as `COUNT()` that could be introduced during the mating or mutation steps. Such features could, amongst other things, allow our algorithm to exploit numerical values or node degrees.

Coverage

It would also be interesting to investigate alternatives to the current coverage approach, for example by exploring semi-isolated subpopulations (demes), or to investigate the effects of including negative samples (currently we only use positive samples and treat everything else as negative).

Clustering

Apart from such modifications of core algorithm, we also want to investigate further non-hierarchical alternatives for the clustering of the graph patterns for query reduction and employ more learning to rank approaches as advanced fusion methods.

Auto-tuning

Last but not least, we plan to investigate two auto-tuning mechanisms: in later runs, our algorithm could slowly increase the initial population's path lengths and decrease the necessary minimal fitness of a pattern to increase coverage.

COMBINATION WITH OTHER APPROACHES Apart from improvements to our graph pattern learner, it will be interesting to investigate the combination of our approach with other complementary ones.

Text based

Good candidates of such other approaches are our text corpus based baselines mentioned in Section 10.4.2.

Vector space based

Based on the vector space formed by our approach, we also plan to investigate combinations with other vector space based approaches. Good candidates for this are Word2Vec, RDF2Vec, NASARI, but also multi-modal fusion with deep learning approaches from the visual and auditory domains.

OTHER APPLICATION AREAS OF THE GRAPH PATTERN LEARNER Last but not least, we plan to apply our graph pattern learner to other domains, such as the medical one. For example, we would like to investigate if our graph pattern learner is able to learn patterns for the prediction of drug-drug interactions. Such other application domains will be especially interesting in combination with the aforementioned planned extension of our algorithm to work with Linked Data Fragments and directly on Literals.

Drug-drug interactions

Part V

APPENDIX

A

VERIFIED MAPPINGS OF EAT TO WIKIPEDIA

Table A.1: Verified Mappings of EAT to Wikipedia. Stimulus and response are strong associations from EAT that occurred $c_{s,r}$ (out of 100) times. The stimulus and response URIs are the Wikipedia article URIs as determined by our semi-automatic mapping approach described in Section 5.3.2.

stimulus	response	$c_{s,r}$	stimulus URI	response URI
fifteen	sixteen	30	wiki:15_(number)	wiki:16_(number)
eighteen	nineteen	31	wiki:18_(number)	wiki:19_(number)
two	three	33	wiki:2_(number)	wiki:3_(number)
thirty	forty	32	wiki:30_(number)	wiki:40_(number)
seven	eight	36	wiki:7_(number)	wiki:8_(number)
tummy	stomach	22	wiki:Abdomen	wiki:Stomach
acceleration	speed	32	wiki:Acceleration	wiki:Speed
accident	car	21	wiki:Accident	wiki:Automobile
accountant	money	28	wiki:Accountant	wiki:Money
acid	alkali	20	wiki:Acid	wiki:Alkali
acorn	tree	36	wiki:Acorn	wiki:Tree
adolescence	youth	30	wiki:Adolescence	wiki:Youth
adults	children	27	wiki:Adult	wiki:Child
in-law	mother	40	wiki:Affinity_(law)	wiki:Mother
afternoon	evening	21	wiki:Afternoon	wiki:Evening
aggression	fight	22	wiki:Aggression	wiki:Combat
alchemy	gold	24	wiki:Alchemy	wiki:Gold
alderman	mayor	22	wiki:Alderman	wiki:Mayor
algebra	maths	34	wiki:Algebra	wiki:Mathematics
aluminium	metal	26	wiki:Aluminium	wiki:Metal
amen	prayer	21	wiki:Amen	wiki:Prayer
amethyst	stone	31	wiki:Amethyst	wiki:Rock_(geology)
amnesia	memory	21	wiki:Amnesia	wiki:Memory
anchor	ship	27	wiki:Anchor	wiki:Ship
angels	heaven	23	wiki:Angel	wiki:Heaven
anguish	pain	30	wiki:Anguish	wiki:Pain
ankle	foot	28	wiki:Ankle	wiki:Foot
annoyance	anger	21	wiki:Annoyance	wiki:Anger
answer	question	48	wiki:Answer	wiki:Question
antimony	metal	22	wiki:Antimony	wiki:Metal
anvil	blacksmith	23	wiki:Anvil	wiki:Blacksmith
anxiety	worry	40	wiki:Anxiety	wiki:Worry
apes	monkeys	32	wiki:Ape	wiki:Monkey
appetite	food	43	wiki:Appetite	wiki:Food
arithmetic	maths	21	wiki:Arithmetic	wiki:Mathematics
arithmetic	sums	21	wiki:Arithmetic	wiki:Summation
arm	leg	40	wiki:Arm	wiki:Leg
armour	knight	21	wiki:Armour	wiki:Knight
armies	war	24	wiki:Army	wiki:War
arrest	police	33	wiki:Arrest	wiki:Police
arson	fire	63	wiki:Arson	wiki:Fire
ashtray	cigarette	28	wiki:Ashtray	wiki:Cigarette
aspidistra	plant	26	wiki:Aspidistra	wiki:Plant
astronaut	moon	22	wiki:Astronaut	wiki:Moon
aunt	uncle	61	wiki:Aunt	wiki:Uncle
author	book	49	wiki:Author	wiki:Book
average	mean	21	wiki:Average	wiki:Mean
bacon	egg	24	wiki:Bacon	wiki:Egg
bacon	eggs	21	wiki:Bacon	wiki:Egg
bagpipes	scotland	23	wiki:Bagpipes	wiki:Scotland
baker	bread	39	wiki:Baker	wiki:Bread
bakery	bread	41	wiki:Bakery	wiki:Bread
baking	bread	21	wiki:Baking	wiki:Bread
barges	canal	23	wiki:Barge	wiki:Canal
bark	tree	26	wiki:Bark	wiki:Tree
barn	hay	20	wiki:Barn	wiki:Hay
barns	hay	26	wiki:Barn	wiki:Hay
barracks	army	28	wiki:Barracks	wiki:Army
barracks	soldiers	34	wiki:Barracks	wiki:Soldier
barrel	beer	51	wiki:Barrel	wiki:Beer
barrels	beer	58	wiki:Barrel	wiki:Beer
casks	beer	23	wiki:Barrel	wiki:Beer
casks	wine	23	wiki:Barrel	wiki:Wine
barrister	law	34	wiki:Barrister	wiki:Law

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
barrister	lawyer	28	wiki:Barrister	wiki:Lawyer
beach	sand	28	wiki:Beach	wiki:Sand
beak	bird	42	wiki:Beak	wiki:Bird
beard	hair	20	wiki:Beard	wiki:Hair
beards	hair	23	wiki:Beard	wiki:Hair
bed	sleep	32	wiki:Bed	wiki:Sleep
bedsit	room	28	wiki:Bedsit	wiki:Room
bees	honey	31	wiki:Bee	wiki:Honey
beetle	insect	22	wiki:Beetle	wiki:Insect
beetroot	red	35	wiki:Beetroot	wiki:Red
belief	religion	20	wiki:Belief	wiki:Religion
blacksmith	horse	20	wiki:Blacksmith	wiki:Horse
blade	razor	20	wiki:Blade	wiki:Razor
blanket	bed	27	wiki:Blanket	wiki:Bed
bleeding	blood	20	wiki:Bleeding	wiki:Blood
haemorrhage	blood	54	wiki:Bleeding	wiki:Blood
blight	potato	24	wiki:Blight	wiki:Potato
blister	foot	22	wiki:Blister	wiki:Foot
blossom	flower	21	wiki:Blossom	wiki:Flower
blouse	shirt	24	wiki:Blouse	wiki:Shirt
boots	shoes	36	wiki:Boot	wiki:Shoe
bottles	beer	35	wiki:Bottle	wiki:Beer
boulder	stone	23	wiki:Boulder	wiki:Rock_(geology)
boulders	rocks	42	wiki:Boulder	wiki:Rock_(geology)
boy	girl	78	wiki:Boy	wiki:Girl
brake	car	21	wiki:Brake	wiki:Automobile
brakes	car	35	wiki:Brake	wiki:Automobile
branch	tree	50	wiki:Branch	wiki:Tree
branches	tree	39	wiki:Branch	wiki:Tree
branches	trees	32	wiki:Branch	wiki:Tree
breeds	dogs	20	wiki:Breed	wiki:Dog
brewing	beer	63	wiki:Brewing	wiki:Beer
bribe	money	37	wiki:Bribery	wiki:Money
brightness	light	36	wiki:Brightness	wiki:Light
brine	salt	42	wiki:Brine	wiki:Salt
brine	sea	26	wiki:Brine	wiki:Sea
broth	soup	55	wiki:Broth	wiki:Soup
brown	black	27	wiki:Brown	wiki:Black
bud	flower	26	wiki:Bud	wiki:Flower
budgerigar	bird	34	wiki:Budgerigar	wiki:Bird
bullet	gun	45	wiki:Bullet	wiki:Gun
dodgems	cars	37	wiki:Bumper_cars	wiki:Automobile
bungalow	house	58	wiki:Bungalow	wiki:House
burrow	rabbit	44	wiki:Burrow	wiki:Rabbit
buses	red	20	wiki:Bus	wiki:Red
busby	hat	20	wiki:Busby	wiki:Hat
butcher	meat	48	wiki:Butcher	wiki:Meat
cable	wire	25	wiki:Cable	wiki:Wire
cactus	plant	42	wiki:Cactus	wiki:Plant
corpse	dead	38	wiki:Cadaver	wiki:Death
calculation	sum	21	wiki:Calculation	wiki:Summation
camping	tent	34	wiki:Camping	wiki:Tent
candles	light	26	wiki:Candle	wiki:Light
canoe	boat	31	wiki:Canoe	wiki:Boat
canvas	tent	41	wiki:Canvas	wiki:Tent
cap	hat	28	wiki:Cap	wiki:Hat
fizz	lemonade	22	wiki:Carbonation	wiki:Lemonade
career	job	41	wiki:Career	wiki:Job
carpenter	wood	46	wiki:Carpentry	wiki:Wood
carpentry	wood	48	wiki:Carpentry	wiki:Wood
cart	horse	66	wiki:Cart	wiki:Horse
cartilage	knee	25	wiki:Cartilage	wiki:Knee
carton	box	24	wiki:Carton	wiki:Box
carton	milk	40	wiki:Carton	wiki:Milk
cassock	priest	21	wiki:Cassock	wiki:Priest
cows	milk	30	wiki:Cattle	wiki:Milk
cavalry	horse	22	wiki:Cavalry	wiki:Horse
cavalry	horses	28	wiki:Cavalry	wiki:Horse
centimetre	inch	20	wiki:Centimetre	wiki:Inch
cerebrum	brain	52	wiki:Cerebrum	wiki:Brain
chalet	switzerland	22	wiki:Chalet	wiki:Switzerland
chart	map	28	wiki:Chart	wiki:Map
chef	food	22	wiki:Chef	wiki:Food
chemistry	physics	23	wiki:Chemistry	wiki:Physics
cheque	money	32	wiki:Cheque	wiki:Money
cherries	red	23	wiki:Cherry	wiki:Red
cherubs	angels	40	wiki:Cherub	wiki:Angel
chimpanzee	monkey	42	wiki:Chimpanzee	wiki:Monkey
chivalry	knight	27	wiki:Chivalry	wiki:Knight
chocolates	sweets	20	wiki:Chocolate	wiki:Candy
christ	jesus	29	wiki:Christ	wiki:Jesus
cigar	smoke	27	wiki:Cigar	wiki:Smoke
cigars	smoke	38	wiki:Cigar	wiki:Smoke
cigarettes	smoke	26	wiki:Cigarette	wiki:Smoke

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
circles	squares	30	wiki:Circle	wiki:Square
city	town	28	wiki:City	wiki:Town
clergyman	vicar	27	wiki:Clergy	wiki:Vicar
climbing	mountain	20	wiki:Climbing	wiki:Mountain
clocks	time	43	wiki:Clock	wiki:Time
clouds	rain	24	wiki:Cloud	wiki:Rain
clouds	sky	24	wiki:Cloud	wiki:Sky
clown	circus	26	wiki:Clown	wiki:Circus
clutch	car	25	wiki:Clutch	wiki:Automobile
coast	sea	27	wiki:Coast	wiki:Sea
cod	fish	56	wiki:Cod	wiki:Fish
cafe	coffee	32	wiki:Coffeehouse	wiki:Coffee
coffins	death	27	wiki:Coffin	wiki:Death
coif	hair	38	wiki:Coif	wiki:Hair
coin	money	20	wiki:Coin	wiki:Money
coins	money	44	wiki:Coin	wiki:Money
college	university	22	wiki:College	wiki:University
colleague	friend	60	wiki:Collegiality	wiki:Friendship
colonel	army	32	wiki:Colonel	wiki:Army
colour	red	31	wiki:Color	wiki:Red
colours	red	21	wiki:Color	wiki:Red
coma	sleep	32	wiki:Coma	wiki:Sleep
comb	hair	59	wiki:Comb	wiki:Hair
comrade	friend	50	wiki:Comrade	wiki:Friendship
concept	idea	46	wiki:Concept	wiki:Idea
confectionery	sweets	50	wiki:Confectionery	wiki:Candy
conjunctivitis	eyes	31	wiki:Conjunctivitis	wiki:Eye
constable	police	24	wiki:Constable	wiki:Police
constellation	star	33	wiki:Constellation	wiki:Star
construction	building	36	wiki:Construction	wiki:Building
convent	nun	48	wiki:Convent	wiki:Nun
convict	prison	23	wiki:Convict	wiki:Prison
cookies	biscuits	22	wiki:Cookie	wiki:Biscuit
corduroy	trousers	53	wiki:Corduroy	wiki:Trousers
coroner	death	36	wiki:Coroner	wiki:Death
corporal	army	20	wiki:Corporal	wiki:Army
corps	army	34	wiki:Corps	wiki:Army
corrosion	rust	47	wiki:Corrosion	wiki:Rust
cosmology	stars	26	wiki:Cosmology	wiki:Star
courts	law	27	wiki:Court	wiki:Law
crate	beer	26	wiki:Crate	wiki:Beer
crayon	pencil	31	wiki:Crayon	wiki:Pencil
crew	ship	38	wiki:Crew	wiki:Ship
criminology	police	23	wiki:Criminology	wiki:Police
crocus	flower	36	wiki:Crocus	wiki:Flower
crop	wheat	26	wiki:Crop	wiki:Wheat
crow	bird	34	wiki:Crow	wiki:Bird
crowd	people	33	wiki:Crowd	wiki:People
crucifix	christ	24	wiki:Crucifix	wiki:Christ
crucifix	cross	32	wiki:Crucifix	wiki:Cross
cuisine	food	30	wiki:Cuisine	wiki:Food
cuisine	kitchen	21	wiki:Cuisine	wiki:Kitchen
dagger	knife	26	wiki:Dagger	wiki:Knife
darts	pub	20	wiki:Darts	wiki:Pub
data	computer	24	wiki:Data	wiki:Computer
dative	ablative	20	wiki:Dative_case	wiki:Ablative_case
david	goliath	28	wiki:David	wiki:Goliath
days	weeks	21	wiki:Day	wiki:Week
debts	money	30	wiki:Debt	wiki:Money
deceit	lie	20	wiki:Deception	wiki:Lie
deity	god	67	wiki:Deity	wiki:God
dentist	teeth	30	wiki:Dentist	wiki:Tooth
dentists	teeth	45	wiki:Dentist	wiki:Tooth
dentistry	teeth	41	wiki:Dentistry	wiki:Tooth
odontology	teeth	23	wiki:Dentistry	wiki:Tooth
dermis	skin	45	wiki:Dermis	wiki:Skin
desire	want	21	wiki:Desire	wiki:Want
detergent	soap	32	wiki:Detergent	wiki:Soap
diabetic	sugar	41	wiki:Diabetes_mellitus	wiki:Sugar
carnations	flowers	37	wiki:Dianthus_caryophyllus	wiki:Flower
nappies	babies	33	wiki:Diaper	wiki:Infant
nappies	baby	21	wiki:Diaper	wiki:Infant
nappy	baby	38	wiki:Diaper	wiki:Infant
digestion	food	23	wiki:Digestion	wiki:Food
document	paper	33	wiki:Document	wiki:Paper
dozen	eggs	34	wiki:Dozen	wiki:Egg
drawbridge	castle	50	wiki:Drawbridge	wiki:Castle
dreams	sleep	35	wiki:Dream	wiki:Sleep
drought	water	21	wiki:Drought	wiki:Water
indigestion	pain	20	wiki:Dyspepsia	wiki:Pain
ear	nose	21	wiki:Ear	wiki:Nose
ears	nose	21	wiki:Ear	wiki:Nose
economy	money	24	wiki:Economy	wiki:Money
edinburgh	scotland	26	wiki:Edinburgh	wiki:Scotland

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
education	school	30	wiki:Education	wiki:School
elect	vote	26	wiki:Election	wiki:Voting
elms	trees	47	wiki:Elm	wiki:Tree
ember	fire	42	wiki:Ember	wiki:Fire
employment	job	27	wiki:Employment	wiki:Job
enemy	friend	34	wiki:Enemy	wiki:Friendship
engagement	marriage	25	wiki:Engagement	wiki:Marriage
engine	car	20	wiki:Engine	wiki:Automobile
motors	cars	37	wiki:Engine	wiki:Automobile
envy	green	28	wiki:Envy	wiki:Green
erosion	soil	23	wiki:Erosion	wiki:Soil
oesophagus	throat	24	wiki:Esophagus	wiki:Throat
estuary	river	57	wiki:Estuary	wiki:River
ethics	morals	32	wiki:Ethics	wiki:Morality
evidence	court	21	wiki:Evidence	wiki:Court
expense	money	37	wiki:Expense	wiki:Money
expenses	money	39	wiki:Expense	wiki:Money
export	import	20	wiki:Export	wiki:Import
eyes	blue	20	wiki:Eye	wiki:Blue
fete	garden	25	wiki:F%C3%AAte	wiki:Garden
falcon	bird	44	wiki:Falcon	wiki:Bird
famine	hunger	35	wiki:Famine	wiki:Hunger
fangs	teeth	41	wiki:Fang	wiki:Tooth
fare	bus	39	wiki:Fare	wiki:Bus
fascist	hitler	20	wiki:Fascism	wiki:Adolf_Hitler
feathers	bird	26	wiki:Feather	wiki:Bird
faeces	shit	26	wiki:Feces	wiki:Shit
fee	money	31	wiki:Fee	wiki:Money
felony	crime	41	wiki:Felony	wiki:Crime
feminine	masculine	26	wiki:Femininity	wiki:Masculinity
filly	horse	55	wiki:Filly	wiki:Horse
finch	bird	42	wiki:Finch	wiki:Bird
firs	trees	46	wiki:Fir	wiki:Tree
firearm	gun	64	wiki:Firearm	wiki:Gun
fishes	sea	20	wiki:Fish	wiki:Sea
flames	fire	55	wiki:Flame	wiki:Fire
flavour	taste	31	wiki:Flavor	wiki:Taste
flirt	girl	21	wiki:Flirting	wiki:Girl
flood	water	46	wiki:Flood	wiki:Water
flounder	fish	25	wiki:Flounder	wiki:Fish
flue	chimney	41	wiki:Flue	wiki:Chimney
fluid	liquid	22	wiki:Fluid	wiki:Liquid
fluid	water	24	wiki:Fluid	wiki:Water
foal	horse	39	wiki:Foal	wiki:Horse
froth	beer	36	wiki:Foam	wiki:Beer
fog	mist	26	wiki:Fog	wiki:Mist
grain	wheat	21	wiki:Food_grain	wiki:Wheat
ford	car	35	wiki:Ford_Motor_Company	wiki:Automobile
forest	trees	28	wiki:Forest	wiki:Tree
forestry	trees	32	wiki:Forestry	wiki:Tree
fortnight	week	21	wiki:Fortnight	wiki:Week
foundry	iron	55	wiki:Foundry	wiki:Iron
fountain	water	37	wiki:Fountain	wiki:Water
francs	france	27	wiki:Franc	wiki:France
francs	money	33	wiki:Franc	wiki:Money
freezing	cold	69	wiki:Freezing	wiki:Cold
shipment	cargo	36	wiki:Freight_transport	wiki:Cargo
friend	enemy	22	wiki:Friendship	wiki:Enemy
frock	dress	35	wiki:Frock	wiki:Dress
frost	cold	21	wiki:Frost	wiki:Cold
fuel	petrol	20	wiki:Fuel	wiki:Gasoline
furlong	mile	47	wiki:Furlong	wiki:Mile
galaxy	stars	59	wiki:Galaxy	wiki:Star
gale	wind	41	wiki:Gale	wiki:Wind
gallon	petrol	20	wiki:Gallon	wiki:Gasoline
gallon	pint	22	wiki:Gallon	wiki:Pint
gallows	hanging	21	wiki:Gallows	wiki:Hanging
gambling	money	21	wiki:Gambling	wiki:Money
gardens	flowers	36	wiki:Garden	wiki:Flower
petrol	car	31	wiki:Gasoline	wiki:Automobile
gelding	horse	30	wiki:Gelding	wiki:Horse
gentleman	lady	38	wiki:Gentleman	wiki:Lady
geology	rocks	44	wiki:Geology	wiki:Rock_(geology)
geometry	maths	28	wiki:Geometry	wiki:Mathematics
geranium	flower	32	wiki:Geranium	wiki:Flower
geranium	plant	20	wiki:Geranium	wiki:Plant
gibbon	monkey	26	wiki:Gibbon	wiki:Monkey
presents	christmas	23	wiki:Gift	wiki:Christmas
gums	teeth	38	wiki:Gingiva	wiki:Tooth
giraffe	neck	33	wiki:Giraffe	wiki:Neck
girders	steel	32	wiki:Girder	wiki:Steel
glitters	gold	69	wiki:Glitter	wiki:Gold
gloom	dark	25	wiki:Gloom	wiki:Darkness
gloves	hands	27	wiki:Glove	wiki:Hand

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
glucose	sugar	56	wiki:Glucose	wiki:Sugar
gnat	fly	22	wiki:Gnat	wiki:Fly
goats	milk	22	wiki:Goat	wiki:Milk
gorilla	ape	44	wiki:Gorilla	wiki:Ape
gram	weight	40	wiki:Gram	wiki:Weight
gramme	weight	45	wiki:Gram	wiki:Weight
granite	stone	22	wiki:Granite	wiki:Rock_(geology)
grapes	wine	24	wiki:Grape	wiki:Wine
green	grass	30	wiki:Green	wiki:Grass
greenhouse	plants	22	wiki:Greenhouse	wiki:Plant
greyhounds	dogs	24	wiki:Greyhound	wiki:Dog
groin	leg	21	wiki:Groin	wiki:Leg
gull	bird	35	wiki:Gull	wiki:Bird
seagull	bird	26	wiki:Gull	wiki:Bird
gull	sea	34	wiki:Gull	wiki:Sea
haggis	scotland	31	wiki:Haggis	wiki:Scotland
curlers	hair	60	wiki:Hair_roller	wiki:Hair
hammock	bed	28	wiki:Hammock	wiki:Bed
handkerchief	nose	23	wiki:Handkerchief	wiki:Nose
harbours	ships	41	wiki:Harbor	wiki:Ship
hare	rabbit	37	wiki:Hare	wiki:Rabbit
hares	rabbits	37	wiki:Hare	wiki:Rabbit
harmony	music	21	wiki:Harmony	wiki:Music
harpoon	spear	21	wiki:Harpoon	wiki:Spear
harpoon	whale	24	wiki:Harpoon	wiki:Whale
harpsichord	music	28	wiki:Harpsichord	wiki:Music
hatred	love	22	wiki:Hatred	wiki:Love
haze	mist	30	wiki:Haze	wiki:Mist
headmaster	school	33	wiki:Head_teacher	wiki:School
headache	pain	47	wiki:Headache	wiki:Pain
headgear	hat	41	wiki:Headgear	wiki:Hat
hearse	death	29	wiki:Hearse	wiki:Death
warmth	cold	23	wiki:Heat	wiki:Cold
heels	shoes	30	wiki:Heel	wiki:Shoe
haemoglobin	blood	66	wiki:Hemoglobin	wiki:Blood
haemorrhoids	blood	27	wiki:Hemorrhoid	wiki:Blood
herd	cows	23	wiki:Herd	wiki:Cattle
herds	cows	34	wiki:Herd	wiki:Cattle
homicide	death	22	wiki:Homicide	wiki:Death
homicide	murder	34	wiki:Homicide	wiki:Murder
homosexual	queer	27	wiki:Homosexuality	wiki:Queer
hooves	horses	41	wiki:Hoof	wiki:Horse
hops	beer	43	wiki:Hops	wiki:Beer
hornpipe	dance	29	wiki:Hornpipe	wiki:Dance
hostel	youth	46	wiki:Hostel	wiki:Youth
hostile	enemy	24	wiki:Hostility	wiki:Enemy
hue	colour	27	wiki:Hue	wiki:Color
intestines	guts	21	wiki:Human_gastrointestinal_tract	wiki:Gut_(anatomy)
stature	height	37	wiki:Human_height	wiki:Height
humour	laugh	23	wiki:Humour	wiki:Laughter
husband	wife	85	wiki:Husband	wiki:Wife
hypothalamus	brain	26	wiki:Hypothalamus	wiki:Brain
icicle	cold	38	wiki:Icicle	wiki:Cold
impressionism	art	25	wiki:Impressionism	wiki:Art
imprisonment	jail	21	wiki:Imprisonment	wiki:Prison
inch	mile	26	wiki:Inch	wiki:Mile
inn	pub	26	wiki:Inn	wiki:Pub
ireland	green	25	wiki:Ireland	wiki:Green
irritation	itch	20	wiki:Irritation	wiki:Itch
jaundice	yellow	77	wiki:Jaundice	wiki:Yellow
jeep	car	29	wiki:Jeep	wiki:Automobile
jockey	horse	55	wiki:Jockey	wiki:Horse
brahms	music	37	wiki:Johannes_Brahms	wiki:Music
joke	laugh	43	wiki:Joke	wiki:Laughter
judge	jury	44	wiki:Judge	wiki:Jury
jug	milk	20	wiki:Jug	wiki:Milk
jug	water	21	wiki:Jug	wiki:Water
juice	fruit	25	wiki:Juice	wiki:Fruit
juices	fruit	39	wiki:Juice	wiki:Fruit
july	august	40	wiki:July	wiki:August
june	july	49	wiki:June	wiki:July
jupiter	planet	34	wiki:Jupiter	wiki:Planet
karate	judo	20	wiki:Karate	wiki:Judo
kennels	dogs	71	wiki:Kennel	wiki:Dog
khaki	army	25	wiki:Khaki	wiki:Army
khaki	shorts	22	wiki:Khaki	wiki:Shorts
kipper	fish	31	wiki:Kipper	wiki:Fish
kippers	fish	27	wiki:Kipper	wiki:Fish
kittens	cat	30	wiki:Kitten	wiki:Cat
kittens	cats	28	wiki:Kitten	wiki:Cat
knight	armour	24	wiki:Knight	wiki:Armour
knitting	wool	22	wiki:Knitting	wiki:Wool
labyrinth	maze	34	wiki:Labyrinth	wiki:Maze

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
lake	water	25	wiki:Lake	wiki:Water
lakes	water	23	wiki:Lake	wiki:Water
lampshade	light	44	wiki:Lampshade	wiki:Light
lantern	light	41	wiki:Lantern	wiki:Light
lard	butter	23	wiki:Lard	wiki:Butter
lard	fat	30	wiki:Lard	wiki:Fat
larder	food	58	wiki:Larder	wiki:Food
laundry	washing	24	wiki:Laundry	wiki:Washing
leaves	tree	26	wiki:Leaf	wiki:Tree
leaves	trees	22	wiki:Leaf	wiki:Tree
leak	water	24	wiki:Leak	wiki:Water
ledger	book	33	wiki:Ledger	wiki:Book
leek	wales	33	wiki:Leek	wiki:Wales
leeks	wales	29	wiki:Leek	wiki:Wales
leg	arm	22	wiki:Leg	wiki:Arm
lemonade	drink	20	wiki:Lemonade	wiki:Drink
lent	easter	21	wiki:Lent	wiki:Easter
lessons	school	21	wiki:Lesson	wiki:School
library	books	48	wiki:Library	wiki:Book
licence	car	20	wiki:License	wiki:Automobile
licence	driving	22	wiki:License	wiki:Driving
lichen	moss	38	wiki:Lichen	wiki:Moss
licking	dog	21	wiki:Licking	wiki:Dog
lie	truth	26	wiki:Lie	wiki:Truth
lieutenant	army	31	wiki:Lieutenant	wiki:Army
lightning	thunder	28	wiki:Lightning	wiki:Thunder
limp	leg	23	wiki:Limp	wiki:Leg
lions	tigers	26	wiki:Lion	wiki:Tiger
lip	mouth	22	wiki:Lip	wiki:Mouth
literature	books	28	wiki:Literature	wiki:Book
litre	pint	23	wiki:Litre	wiki:Pint
locomotive	train	50	wiki:Locomotive	wiki:Train
loft	attic	31	wiki:Loft	wiki:Attic
beethoven	music	31	wiki:Ludwig_van_Beethoven	wiki:Music
mackerel	fish	79	wiki:Mackerel	wiki:Fish
mackintosh	rain	31	wiki:Mackintosh	wiki:Rain
mallet	hammer	49	wiki:Mallet	wiki:Hammer
mare	horse	53	wiki:Mare	wiki:Horse
margarine	butter	67	wiki:Margarine	wiki:Butter
martyr	saint	22	wiki:Martyr	wiki:Saint
mat	door	20	wiki:Mat	wiki:Door
matchbox	matches	21	wiki:Matchbox	wiki:Match
matron	hospital	30	wiki:Matron	wiki:Hospital
mattress	bed	60	wiki:Mattress	wiki:Bed
mauve	colour	20	wiki:Mauve	wiki:Color
mauve	purple	34	wiki:Mauve	wiki:Purple
meal	food	42	wiki:Meal	wiki:Food
meals	food	38	wiki:Meal	wiki:Food
mercenary	money	23	wiki:Mercenary	wiki:Money
metre	yard	26	wiki:Metre	wiki:Yard
metres	yards	22	wiki:Metre	wiki:Yard
milkmaid	cow	25	wiki:Milkmaid	wiki:Cattle
miner	coal	38	wiki:Miner	wiki:Coal
miners	coal	35	wiki:Miner	wiki:Coal
mire	mud	28	wiki:Mire	wiki:Mud
miser	money	29	wiki:Miser	wiki:Money
mist	fog	43	wiki:Mist	wiki:Fog
monastery	monk	40	wiki:Monastery	wiki:Monk
monastery	monks	23	wiki:Monastery	wiki:Monk
monkeys	apes	22	wiki:Monkey	wiki:Ape
month	year	41	wiki:Month	wiki:Year
moorland	heath	23	wiki:Moorland	wiki:Heath
mortgage	house	59	wiki:Mortgage_loan	wiki:House
mosaic	pattern	24	wiki:Mosaic	wiki:Pattern
everest	mountain	39	wiki:Mount_Everest	wiki:Mountain
mountains	hills	20	wiki:Mountain	wiki:Hill
mourning	death	27	wiki:Mourning	wiki:Death
mousse	chocolate	36	wiki:Mousse	wiki:Chocolate
nap	sleep	51	wiki:Nap	wiki:Sleep
napalm	bomb	36	wiki:Napalm	wiki:Bomb
napalm	vietnam	20	wiki:Napalm	wiki:Vietnam
nape	neck	79	wiki:Nape	wiki:Neck
daffodils	yellow	27	wiki:Narcissus_(plant)	wiki:Yellow
navigation	ship	20	wiki:Navigation	wiki:Ship
need	want	21	wiki:Need	wiki:Want
needs	wants	27	wiki:Need	wiki:Want
netball	game	32	wiki:Netball	wiki:Game
neurology	brain	24	wiki:Neurology	wiki:Brain
newt	frog	24	wiki:Newt	wiki:Frog
nostril	nose	69	wiki:Nostril	wiki:Nose
nostrils	nose	70	wiki:Nostril	wiki:Nose
noun	verb	44	wiki:Noun	wiki:Verb
novels	books	45	wiki:Novel	wiki:Book
oar	boat	47	wiki:Oar	wiki:Boat

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
oars	boat	51	wiki:Oar	wiki:Boat
ocean	sea	36	wiki:Ocean	wiki:Sea
octaves	music	45	wiki:Octave	wiki:Music
octopus	eight	20	wiki:Octopus	wiki:8_(number)
offspring	child	26	wiki:Offspring	wiki:Child
offspring	children	25	wiki:Offspring	wiki:Child
omelette	egg	44	wiki:Omelette	wiki:Egg
onyx	stone	31	wiki:Onyx	wiki:Rock_(geology)
optimism	hope	22	wiki:Optimism	wiki:Hope
optimism	pessimism	25	wiki:Optimism	wiki:Pessimism
oranges	apples	26	wiki:Orange_(fruit)	wiki:Apple
orchard	apple	23	wiki:Orchard	wiki:Apple
ore	gold	20	wiki:Ore	wiki:Gold
orphan	child	30	wiki:Orphan	wiki:Child
osprey	bird	44	wiki:Osprey	wiki:Bird
ostrich	bird	20	wiki:Ostrich	wiki:Bird
ostrich	feather	22	wiki:Ostrich	wiki:Feather
overdraft	bank	50	wiki:Overdraft	wiki:Bank
oxygen	air	20	wiki:Oxygen	wiki:Atmosphere_of_Earth
pyjamas	bed	33	wiki:Pajamas	wiki:Bed
palate	mouth	26	wiki:Palate	wiki:Mouth
palate	taste	20	wiki:Palate	wiki:Taste
pansies	flowers	26	wiki:Pansy	wiki:Flower
pansy	flower	34	wiki:Pansy	wiki:Flower
pantry	food	48	wiki:Pantry	wiki:Food
parent	father	21	wiki:Parent	wiki:Father
paris	france	25	wiki:Paris	wiki:France
parking	car	23	wiki:Parking	wiki:Automobile
parrot	bird	21	wiki:Parrot	wiki:Bird
parson	vicar	28	wiki:Parson	wiki:Vicar
kneecap	leg	20	wiki:Patella	wiki:Leg
patients	hospital	44	wiki:Patient	wiki:Hospital
patriotism	country	21	wiki:Patriotism	wiki:Country
paws	cat	25	wiki:Paw	wiki:Cat
paw	dog	29	wiki:Paw	wiki:Dog
payment	money	36	wiki:Payment	wiki:Money
peace	war	42	wiki:Peace	wiki:War
pear	apple	28	wiki:Pear	wiki:Apple
peeler	potato	36	wiki:Peeler	wiki:Potato
pen	ink	30	wiki:Pen	wiki:Ink
peninsula	island	24	wiki:Peninsula	wiki:Island
pence	money	20	wiki:Penny	wiki:Money
pennies	money	21	wiki:Penny	wiki:Money
perfume	scent	22	wiki:Perfume	wiki:Odor
perjury	lie	23	wiki:Perjury	wiki:Lie
persons	people	53	wiki:Person	wiki:People
pets	dogs	23	wiki:Pet	wiki:Dog
petal	flower	57	wiki:Petal	wiki:Flower
petals	flower	33	wiki:Petal	wiki:Flower
petals	flowers	26	wiki:Petal	wiki:Flower
petrel	bird	35	wiki:Petrel	wiki:Bird
physics	chemistry	23	wiki:Physics	wiki:Chemistry
pillow	bed	23	wiki:Pillow	wiki:Bed
pillows	bed	34	wiki:Pillow	wiki:Bed
pillow	sleep	22	wiki:Pillow	wiki:Sleep
pineapple	fruit	23	wiki:Pineapple	wiki:Fruit
pint	beer	38	wiki:Pint	wiki:Beer
pints	beer	40	wiki:Pint	wiki:Beer
pints	milk	25	wiki:Pint	wiki:Milk
pistol	gun	42	wiki:Pistol	wiki:Gun
plaice	fish	67	wiki:Plaice	wiki:Fish
planet	mars	25	wiki:Planet	wiki:Mars
planets	stars	21	wiki:Planet	wiki:Star
plankton	sea	20	wiki:Plankton	wiki:Sea
sap	tree	52	wiki:Plant_sap	wiki:Tree
playground	children	27	wiki:Playground	wiki:Child
pleat	skirt	54	wiki:Pleat	wiki:Skirt
pleats	skirt	57	wiki:Pleat	wiki:Skirt
policeman	law	21	wiki:Police_officer	wiki:Law
pollen	flower	26	wiki:Pollen	wiki:Flower
perspex	glass	46	wiki:Poly(methyl_methacrylate)	wiki:Glass
polythene	plastic	20	wiki:Polyethylene	wiki:Plastic
pony	horse	30	wiki:Pony	wiki:Horse
porpoise	fish	22	wiki:Porpoise	wiki:Fish
porridge	oats	26	wiki:Porridge	wiki:Oat
ports	ships	32	wiki:Port	wiki:Ship
portrait	picture	29	wiki:Portrait	wiki:Image
pottage	soup	37	wiki:Pottage	wiki:Soup
prayer	god	20	wiki:Prayer	wiki:God
jailor	prison	25	wiki:Prison_officer	wiki:Prison
inmate	prison	28	wiki:Prisoner	wiki:Prison
prisoners	war	20	wiki:Prisoner	wiki:War
profession	job	22	wiki:Profession	wiki:Job

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
prosperity	wealth	29	wiki:Prosperity	wiki:Wealth
protestant	catholic	42	wiki:Protestantism	wiki:Catholicism
pub	drink	26	wiki:Pub	wiki:Drink
pump	water	26	wiki:Pump	wiki:Water
pupil	eye	33	wiki:Pupil	wiki:Eye
pupils	eyes	37	wiki:Pupil	wiki:Eye
puppies	dog	27	wiki:Puppy	wiki:Dog
puppies	dogs	30	wiki:Puppy	wiki:Dog
putty	window	22	wiki:Putty	wiki:Window
quadruped	four	27	wiki:Quadrupedalism	wiki:4_(number)
quarantine	dogs	21	wiki:Quarantine	wiki:Dog
quarto	paper	39	wiki:Quarto	wiki:Paper
quilt	bed	28	wiki:Quilt	wiki:Bed
quinine	drug	35	wiki:Quinine	wiki:Drug
racquet	tennis	59	wiki:Racket_(sports_equipment)	wiki:Tennis
rainbow	colours	30	wiki:Rainbow	wiki:Color
rayon	nylon	22	wiki:Rayon	wiki:Nylon
reagents	chemistry	22	wiki:Reagent	wiki:Chemistry
recipe	food	24	wiki:Recipe	wiki:Food
recital	music	23	wiki:Recital	wiki:Music
rectangle	square	56	wiki:Rectangle	wiki:Square
refectory	food	41	wiki:Refectory	wiki:Food
regiment	army	43	wiki:Regiment	wiki:Army
religion	god	23	wiki:Religion	wiki:God
remittance	money	27	wiki:Remittance	wiki:Money
reply	answer	47	wiki:Reply	wiki:Answer
reptile	snake	27	wiki:Reptile	wiki:Snake
reservoir	water	75	wiki:Reservoir	wiki:Water
resort	holiday	28	wiki:Resort	wiki:Holiday
resurrection	christ	34	wiki:Resurrection	wiki:Christ
rhythm	music	20	wiki:Rhythm	wiki:Music
ribbon	hair	25	wiki:Ribbon	wiki:Hair
ribbons	hair	28	wiki:Ribbon	wiki:Hair
rifle	gun	27	wiki:Rifle	wiki:Gun
rifles	guns	36	wiki:Rifle	wiki:Gun
rifleman	gun	23	wiki:Rifleman	wiki:Gun
rigging	ship	31	wiki:Rigging	wiki:Ship
rocks	sea	24	wiki:Rock_(geology)	wiki:Sea
rodent	rat	55	wiki:Rodent	wiki:Rat
rodents	rats	56	wiki:Rodent	wiki:Rat
roe	fish	35	wiki:Roe	wiki:Fish
rome	italy	29	wiki:Rome	wiki:Italy
romper	baby	22	wiki:Romper_suit	wiki:Infant
ruin	castle	21	wiki:Ruins	wiki:Castle
ruins	castle	29	wiki:Ruins	wiki:Castle
saddle	horse	63	wiki:Saddle	wiki:Horse
salad	lettuce	21	wiki:Salad	wiki:Lettuce
salary	money	33	wiki:Salary	wiki:Money
sandal	foot	20	wiki:Sandal	wiki:Foot
sandal	shoe	27	wiki:Sandal	wiki:Shoe
sandals	shoes	29	wiki:Sandal	wiki:Shoe
sarcasm	wit	41	wiki:Sarcasm	wiki:Wit
sari	india	22	wiki:Sari	wiki:India
satin	silk	29	wiki:Satin	wiki:Silk
saturn	planet	28	wiki:Saturn	wiki:Planet
sausage	meat	20	wiki:Sausage	wiki:Meat
saving	money	41	wiki:Saving	wiki:Money
sawdust	wood	32	wiki:Sawdust	wiki:Wood
scaffolding	building	28	wiki:Scaffolding	wiki:Building
scalpel	knife	25	wiki:Scalpel	wiki:Knife
scarf	neck	23	wiki:Scarf	wiki:Neck
schooner	ship	26	wiki:Schooner	wiki:Ship
scones	butter	20	wiki:Scone	wiki:Butter
sculpture	art	23	wiki:Sculpture	wiki:Art
scythe	grass	20	wiki:Scythe	wiki:Grass
seaman	sailor	27	wiki:Seaman	wiki:Sailor
seascape	landscape	30	wiki:Seascape	wiki:Landscape
seat	chair	20	wiki:Seat	wiki:Chair
seatbelt	car	47	wiki:Seat_belt	wiki:Automobile
seatbelt	safety	21	wiki:Seat_belt	wiki:Safety
seconds	time	25	wiki:Second	wiki:Time
sect	religion	23	wiki:Sect	wiki:Religion
serf	slave	30	wiki:Serfdom	wiki:Slavery
setter	dog	47	wiki:Setter	wiki:Dog
setter	red	28	wiki:Setter	wiki:Red
shilling	pence	20	wiki:Shilling	wiki:Penny
shillings	pence	52	wiki:Shilling	wiki:Penny
ship	sea	23	wiki:Ship	wiki:Sea
ships	sea	32	wiki:Ship	wiki:Sea
shiver	cold	58	wiki:Shivering	wiki:Cold
shoes	socks	20	wiki:Shoe	wiki:Sock
shooting	gun	21	wiki:Shooting	wiki:Gun
shore	sea	44	wiki:Shore	wiki:Sea

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
shoulder	arm	20	wiki:Shoulder	wiki:Arm
shovel	spade	43	wiki:Shovel	wiki:Spade
bushes	trees	22	wiki:Shrub	wiki:Tree
sickle	scythe	21	wiki:Sickle	wiki:Scythe
signalman	railway	39	wiki:Signalman	wiki:Rail_transport
ski	snow	26	wiki:Ski	wiki:Snow
skiing	snow	42	wiki:Skiing	wiki:Snow
cranium	head	37	wiki:Skull	wiki:Head
sledge	snow	37	wiki:Sled	wiki:Snow
sleeping	bed	21	wiki:Sleep	wiki:Bed
slipper	shoe	30	wiki:Slipper	wiki:Shoe
smog	fog	38	wiki:Smog	wiki:Fog
sneeze	cough	27	wiki:Sneeze	wiki:Cough
snipe	bird	22	wiki:Snipe	wiki:Bird
snore	sleep	47	wiki:Snoring	wiki:Sleep
snoring	sleep	29	wiki:Snoring	wiki:Sleep
solicitor	law	21	wiki:Solicitor	wiki:Law
solicitor	lawyer	26	wiki:Solicitor	wiki:Lawyer
somerset	cider	30	wiki:Somerset	wiki:Cider
sonata	music	23	wiki:Sonata	wiki:Music
soot	black	37	wiki:Soot	wiki:Black
soot	chimney	23	wiki:Soot	wiki:Chimney
sound	noise	32	wiki:Sound	wiki:Noise
spade	shovel	26	wiki:Spade	wiki:Shovel
spelling	words	20	wiki:Spelling	wiki:Word
splinter	wood	54	wiki:Splinter	wiki:Wood
spoon	fork	21	wiki:Spoon	wiki:Fork
spouse	husband	22	wiki:Spouse	wiki:Husband
spouse	wife	57	wiki:Spouse	wiki:Wife
squares	circles	26	wiki:Square	wiki:Circle
stable	horse	48	wiki:Stable	wiki:Horse
stallion	horse	59	wiki:Stallion	wiki:Horse
stallion	mare	20	wiki:Stallion	wiki:Mare
stamen	flower	42	wiki:Stamen	wiki:Flower
stanza	poem	23	wiki:Stanza	wiki:Poetry
starling	bird	58	wiki:Starling	wiki:Bird
starlings	birds	58	wiki:Starling	wiki:Bird
stationery	paper	39	wiki:Stationery	wiki:Paper
strategy	plan	33	wiki:Strategy	wiki:Plan
structure	building	29	wiki:Structure	wiki:Building
suburb	town	20	wiki:Suburb	wiki:Town
sufferer	pain	28	wiki:Suffering	wiki:Pain
sugars	sweet	20	wiki:Sugar	wiki:Sweetness
suggestion	idea	23	wiki:Suggestion	wiki:Idea
suicide	death	41	wiki:Suicide	wiki:Death
sums	maths	20	wiki:Summation	wiki:Mathematics
summer	winter	33	wiki:Summer	wiki:Winter
sun	moon	25	wiki:Sun	wiki:Moon
swallow	bird	27	wiki:Swallow	wiki:Bird
sweetness	sugar	31	wiki:Sweetness	wiki:Sugar
symphony	music	31	wiki:Symphony	wiki:Music
syrup	treacle	21	wiki:Syrup	wiki:Treacle
tadpole	frog	50	wiki:Tadpole	wiki:Frog
tanks	war	21	wiki:Tank	wiki:War
tankard	beer	52	wiki:Tankard	wiki:Beer
tar	road	25	wiki:Tar	wiki:Road
tarmac	road	59	wiki:Tarmac	wiki:Road
teacup	saucer	38	wiki:Teacup	wiki:Saucer
telegram	news	20	wiki:Telegraphy	wiki:News
terrain	land	26	wiki:Terrain	wiki:Land
thatch	roof	43	wiki:Thatching	wiki:Roof
reverend	vicar	24	wiki:The_Reverend	wiki:Vicar
thermodynamics	heat	29	wiki:Thermodynamics	wiki:Heat
thermometer	heat	24	wiki:Thermometer	wiki:Heat
thermometer	temperature	36	wiki:Thermometer	wiki:Temperature
thigh	leg	35	wiki:Thigh	wiki:Leg
tickle	laugh	24	wiki:Tickling	wiki:Laughter
tide	sea	28	wiki:Tide	wiki:Sea
tiger	lion	25	wiki:Tiger	wiki:Lion
tile	roof	36	wiki:Tile	wiki:Roof
tiles	roof	25	wiki:Tile	wiki:Roof
timer	clock	24	wiki:Timer	wiki:Clock
tinsel	christmas	34	wiki:Tinsel	wiki:Christmas
toe	foot	28	wiki:Toe	wiki:Foot
tomatoes	red	25	wiki:Tomato	wiki:Red
tomb	grave	23	wiki:Tomb	wiki:Grave
ton	weight	37	wiki:Ton	wiki:Weight
toothache	dentist	20	wiki:Toothache	wiki:Dentist
toothache	pain	38	wiki:Toothache	wiki:Pain
toothbrush	teeth	34	wiki:Toothbrush	wiki:Tooth
tornado	wind	36	wiki:Tornado	wiki:Wind
toys	children	20	wiki:Toy	wiki:Child
tractor	farm	30	wiki:Tractor	wiki:Farm
trams	buses	24	wiki:Tram	wiki:Bus

Table A.1: Verified Mappings of EAT to Wikipedia (continued)

stimulus	response	$c_{s,r}$	stimulus URI	response URI
tranquillity	peace	38	wiki:Tranquillity	wiki:Peace
transport	bus	24	wiki:Transport	wiki:Bus
trees	leaves	20	wiki:Tree	wiki:Leaf
troops	army	21	wiki:Troop	wiki:Army
troops	soldiers	22	wiki:Troop	wiki:Soldier
hurricane	wind	27	wiki:Tropical_cyclone	wiki:Wind
trot	horse	28	wiki:Trot	wiki:Horse
truth	lie	24	wiki:Truth	wiki:Lie
tulip	flower	26	wiki:Tulip	wiki:Flower
twig	branch	37	wiki:Twig	wiki:Branch
twig	tree	22	wiki:Twig	wiki:Tree
udder	cow	69	wiki:Udder	wiki:Cattle
umbrella	rain	58	wiki:Umbrella	wiki:Rain
uncle	aunt	56	wiki:Uncle	wiki:Aunt
underlay	carpet	55	wiki:Underlay	wiki:Carpet
womb	baby	20	wiki:Uterus	wiki:Infant
vacation	holiday	40	wiki:Vacation	wiki:Holiday
van	car	21	wiki:Van	wiki:Automobile
vase	flowers	43	wiki:Vase	wiki:Flower
veal	ham	24	wiki:Veal	wiki:Ham
veal	meat	31	wiki:Veal	wiki:Meat
vehicle	car	79	wiki:Vehicle	wiki:Automobile
velocity	speed	63	wiki:Velocity	wiki:Speed
village	town	24	wiki:Village	wiki:Town
vine	grape	35	wiki:Vine	wiki:Grape
vines	grapes	47	wiki:Vine	wiki:Grape
vinegar	salt	33	wiki:Vinegar	wiki:Salt
violins	music	27	wiki:Violin	wiki:Music
vocation	job	25	wiki:Vocation	wiki:Job
wage	money	26	wiki:Wage	wiki:Money
wages	money	32	wiki:Wage	wiki:Money
wallpaper	paint	20	wiki:Wallpaper	wiki:Paint
wand	fairy	26	wiki:Wand	wiki:Fairy
wardrobe	clothes	59	wiki:Wardrobe	wiki:Clothing
dustbin	rubbish	27	wiki:Waste_container	wiki:Municipal_solid_waste
wave	sea	40	wiki:Wave	wiki:Sea
waves	sea	57	wiki:Wave	wiki:Sea
wealth	money	26	wiki:Wealth	wiki:Money
weasel	stoat	57	wiki:Weasel	wiki:Stoat
weather	rain	24	wiki:Weather	wiki:Rain
weeks	days	38	wiki:Week	wiki:Day
whisk	egg	29	wiki:Whisk	wiki:Egg
semibreves	music	43	wiki:Whole_note	wiki:Music
wife	husband	46	wiki:Wife	wiki:Husband
wig	hair	51	wiki:Wig	wiki:Hair
gnu	animal	20	wiki:Wildebeest	wiki:Animal
wing	bird	27	wiki:Wing	wiki:Bird
wings	bird	24	wiki:Wing	wiki:Bird
winter	snow	25	wiki:Winter	wiki:Snow
wireless	radio	40	wiki:Wireless	wiki:Radio
mozart	music	46	wiki:Wolfgang_Amadeus_Mozart	wiki:Music
woman	man	59	wiki:Woman	wiki:Man
wool	sheep	30	wiki:Wool	wiki:Sheep
yachts	boats	28	wiki:Yacht	wiki:Boat
year	month	21	wiki:Year	wiki:Month
zoo	animals	34	wiki:Zoo	wiki:Animal
zoology	animals	30	wiki:Zoology	wiki:Animal

B

SEMANTIC ASSOCATION GROUND TRUTH DATASET

Table B.1: Semantic Assocation Ground Truth Dataset. Shown are the 727 semantic associations between unique DBpedia entity pairs, as described in Section 6.2. The train and test columns indicate which of the pairs were randomly sampled into the training and test set. In total, there are 655 training pairs and 72 test pairs.

stimulus entity	response entity	train	test
dbr:15_(number)	dbr:16_(number)	✓	
dbr:18_(number)	dbr:19_(number)	✓	
dbr:2_(number)	dbr:3_(number)		✓
dbr:30_(number)	dbr:40_(number)	✓	
dbr:7_(number)	dbr:8_(number)	✓	
dbr:Abdomen	dbr:Stomach		✓
dbr:Acceleration	dbr:Speed	✓	
dbr:Accident	dbr:Automobile	✓	
dbr:Accountant	dbr:Money		✓
dbr:Acid	dbr:Alkali	✓	
dbr:Acorn	dbr:Tree	✓	
dbr:Adolescence	dbr:Youth	✓	
dbr:Adult	dbr:Child	✓	
dbr:Affinity_(law)	dbr:Mother	✓	
dbr:Afternoon	dbr:Evening	✓	
dbr:Aggression	dbr:Combat		✓
dbr:Alchemy	dbr:Gold	✓	
dbr:Alderman	dbr:Mayor	✓	
dbr:Algebra	dbr:Mathematics	✓	
dbr:Aluminium	dbr:Metal	✓	
dbr:Amen	dbr:Prayer	✓	
dbr:Amethyst	dbr:Rock_(geology)	✓	
dbr:Amnesia	dbr:Memory	✓	
dbr:Anchor	dbr:Ship		✓
dbr:Angel	dbr:Heaven	✓	
dbr:Anguish	dbr:Pain		✓
dbr:Ankle	dbr:Foot	✓	
dbr:Annoyance	dbr:Anger	✓	
dbr:Answer	dbr:Question	✓	
dbr:Antimony	dbr:Metal	✓	
dbr:Anvil	dbr:Blacksmith	✓	
dbr:Anxiety	dbr:Worry	✓	
dbr:Ape	dbr:Monkey	✓	
dbr:Appetite	dbr:Food		✓
dbr:Arithmetic	dbr:Mathematics	✓	
dbr:Arithmetic	dbr:Summation	✓	
dbr:Arm	dbr:Leg	✓	
dbr:Armour	dbr:Knight	✓	
dbr:Army	dbr:War		✓
dbr:Arrest	dbr:Police	✓	
dbr:Arson	dbr:Fire	✓	
dbr:Ashtray	dbr:Cigarette	✓	
dbr:Aspidistra	dbr:Plant	✓	
dbr:Astronaut	dbr:Moon	✓	
dbr:Aunt	dbr:Uncle		✓
dbr:Author	dbr:Book	✓	
dbr:Average	dbr:Mean		✓
dbr:Bacon	dbr:Egg	✓	
dbr:Bagpipes	dbr:Scotland	✓	
dbr:Baker	dbr:Bread		✓
dbr:Bakery	dbr:Bread	✓	
dbr:Baking	dbr:Bread	✓	
dbr:Barge	dbr:Canal	✓	
dbr:Bark	dbr:Tree	✓	
dbr:Barn	dbr:Hay	✓	
dbr:Barracks	dbr:Army	✓	
dbr:Barracks	dbr:Soldier		✓
dbr:Barrel	dbr:Beer	✓	
dbr:Barrel	dbr:Wine	✓	
dbr:Barrister	dbr:Law	✓	
dbr:Barrister	dbr:Lawyer	✓	

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:Beach	dbr:Sand	✓	
dbr:Beak	dbr:Bird	✓	
dbr:Beard	dbr:Hair	✓	
dbr:Bed	dbr:Sleep	✓	
dbr:Bedsit	dbr:Room	✓	
dbr:Bee	dbr:Honey	✓	
dbr:Beetle	dbr:Insect	✓	
dbr:Beetroot	dbr:Red	✓	
dbr:Belief	dbr:Religion	✓	
dbr:Blacksmith	dbr:Horse	✓	
dbr:Blade	dbr:Razor	✓	
dbr:Blanket	dbr:Bed	✓	
dbr:Bleeding	dbr:Blood	✓	
dbr:Blight	dbr:Potato	✓	
dbr:Blister	dbr:Foot	✓	
dbr:Blossom	dbr:Flower	✓	
dbr:Blouse	dbr:Shirt	✓	
dbr:Boot	dbr:Shoe	✓	
dbr:Bottle	dbr:Beer		✓
dbr:Boulder	dbr:Rock_(geology)	✓	
dbr:Boy	dbr:Girl	✓	
dbr:Brake	dbr:Automobile	✓	
dbr:Branch	dbr:Tree	✓	
dbr:Breed	dbr:Dog	✓	
dbr:Brewing	dbr:Beer	✓	
dbr:Bribery	dbr:Money	✓	
dbr:Brightness	dbr:Light	✓	
dbr:Brine	dbr:Salt	✓	
dbr:Brine	dbr:Sea		✓
dbr:Broth	dbr:Soup	✓	
dbr:Brown	dbr:Black	✓	
dbr:Bud	dbr:Flower	✓	
dbr:Budgerigar	dbr:Bird	✓	
dbr:Bullet	dbr:Gun	✓	
dbr:Bumper_cars	dbr:Automobile	✓	
dbr:Bungalow	dbr:House		✓
dbr:Burrow	dbr:Rabbit		✓
dbr:Bus	dbr:Red	✓	
dbr:Busby	dbr:Hat		✓
dbr:Butcher	dbr:Meat	✓	
dbr:Cable	dbr:Wire	✓	
dbr:Cactus	dbr:Plant	✓	
dbr:Cadaver	dbr:Death	✓	
dbr:Calculation	dbr:Summation	✓	
dbr:Camping	dbr:Tent	✓	
dbr:Candle	dbr:Light	✓	
dbr:Canoe	dbr:Boat	✓	
dbr:Canvas	dbr:Tent	✓	
dbr:Cap	dbr:Hat	✓	
dbr:Carbonation	dbr:Lemonade	✓	
dbr:Career	dbr:Job	✓	
dbr:Carpentry	dbr:Wood	✓	
dbr:Cart	dbr:Horse	✓	
dbr:Cartilage	dbr:Knee	✓	
dbr:Carton	dbr:Box	✓	
dbr:Carton	dbr:Milk	✓	
dbr:Cassock	dbr:Priest	✓	
dbr:Cattle	dbr:Milk	✓	
dbr:Cavalry	dbr:Horse	✓	
dbr:Centimetre	dbr:Inch	✓	
dbr:Cerebrum	dbr:Brain	✓	
dbr:Chalet	dbr:Switzerland	✓	
dbr:Chart	dbr:Map	✓	
dbr:Chef	dbr:Food	✓	
dbr:Chemistry	dbr:Physics	✓	
dbr:Cheque	dbr:Money	✓	
dbr:Cherry	dbr:Red	✓	
dbr:Cherub	dbr:Angel	✓	
dbr:Chimpanzee	dbr:Monkey	✓	
dbr:Chivalry	dbr:Knight	✓	
dbr:Chocolate	dbr:Candy	✓	
dbr:Christ	dbr:Jesus	✓	
dbr:Cigar	dbr:Smoke	✓	
dbr:Cigarette	dbr:Smoke	✓	
dbr:Circle	dbr:Square	✓	
dbr:City	dbr:Town	✓	
dbr:Clergy	dbr:Vicar	✓	
dbr:Climbing	dbr:Mountain	✓	
dbr:Clock	dbr:Time	✓	
dbr:Cloud	dbr:Rain	✓	
dbr:Cloud	dbr:Sky	✓	
dbr:Clown	dbr:Circus	✓	
dbr:Clutch	dbr:Automobile	✓	
dbr:Coast	dbr:Sea	✓	

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:Cod	dbr:Fish		✓
dbr:Coffeehouse	dbr:Coffee	✓	
dbr:Coffin	dbr:Death	✓	
dbr:Coif	dbr:Hair	✓	
dbr:Coin	dbr:Money	✓	
dbr:College	dbr:University	✓	
dbr:Collegiality	dbr:Friendship	✓	
dbr:Colonel	dbr:Army	✓	
dbr:Color	dbr:Red	✓	
dbr:Coma	dbr:Sleep		✓
dbr:Comb	dbr:Hair	✓	
dbr:Comrade	dbr:Friendship	✓	
dbr:Concept	dbr:Idea	✓	
dbr:Confectionery	dbr:Candy	✓	
dbr:Conjunctivitis	dbr:Eye	✓	
dbr:Constable	dbr:Police	✓	
dbr:Constellation	dbr:Star	✓	
dbr:Construction	dbr:Building	✓	
dbr:Convent	dbr:Nun	✓	
dbr:Convict	dbr:Prison	✓	
dbr:Cookie	dbr:Biscuit	✓	
dbr:Corduroy	dbr:Trousers	✓	
dbr:Coroner	dbr:Death	✓	
dbr:Corporal	dbr:Army	✓	
dbr:Corps	dbr:Army	✓	
dbr:Corrosion	dbr:Rust	✓	
dbr:Cosmology	dbr:Star	✓	
dbr:Court	dbr:Law	✓	
dbr:Crate	dbr:Beer	✓	
dbr:Crayon	dbr:Pencil		✓
dbr:Crew	dbr:Ship		✓
dbr:Criminology	dbr:Police	✓	
dbr:Crocus	dbr:Flower		✓
dbr:Crop	dbr:Wheat		✓
dbr:Crow	dbr:Bird	✓	
dbr:Crowd	dbr:People	✓	
dbr:Crucifix	dbr:Christ	✓	
dbr:Crucifix	dbr:Cross	✓	
dbr:Cuisine	dbr:Food	✓	
dbr:Cuisine	dbr:Kitchen	✓	
dbr:Dagger	dbr:Knife	✓	
dbr:Darts	dbr:Pub	✓	
dbr:Data	dbr:Computer	✓	
dbr:Dative_case	dbr:Ablative_case	✓	
dbr:David	dbr:Goliath	✓	
dbr:Day	dbr:Week	✓	
dbr:Debt	dbr:Money	✓	
dbr:Deception	dbr:Lie	✓	
dbr:Deity	dbr:God	✓	
dbr:Dentist	dbr:Tooth		✓
dbr:Dentistry	dbr:Tooth	✓	
dbr:Dermis	dbr:Skin	✓	
dbr:Desire	dbr:Want	✓	
dbr:Detergent	dbr:Soap	✓	
dbr:Diabetes_mellitus	dbr:Sugar	✓	
dbr:Dianthus_caryophyllus	dbr:Flower		✓
dbr:Diaper	dbr:Infant	✓	
dbr:Digestion	dbr:Food	✓	
dbr:Document	dbr:Paper		✓
dbr:Dozen	dbr:Egg	✓	
dbr:Drawbridge	dbr:Castle	✓	
dbr:Dream	dbr:Sleep	✓	
dbr:Drought	dbr:Water	✓	
dbr:Dyspepsia	dbr:Pain	✓	
dbr:Ear	dbr:Nose	✓	
dbr:Economy	dbr:Money	✓	
dbr:Edinburgh	dbr:Scotland	✓	
dbr:Education	dbr:School	✓	
dbr:Election	dbr:Voting	✓	
dbr:Elm	dbr:Tree	✓	
dbr:Ember	dbr:Fire	✓	
dbr:Employment	dbr:Job	✓	
dbr:Enemy	dbr:Friendship	✓	
dbr:Engagement	dbr:Marriage	✓	
dbr:Engine	dbr:Automobile	✓	
dbr:Envy	dbr:Green	✓	
dbr:Erosion	dbr:Soil	✓	
dbr:Esophagus	dbr:Throat		✓
dbr:Estuary	dbr:River	✓	
dbr:Ethics	dbr:Morality	✓	
dbr:Evidence	dbr:Court	✓	
dbr:Expense	dbr:Money	✓	
dbr:Export	dbr:Import	✓	
dbr:Eye	dbr:Blue	✓	

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:F%C3%AAte	dbr:Garden	✓	
dbr:Falcon	dbr:Bird	✓	
dbr:Famine	dbr:Hunger	✓	
dbr:Fang	dbr:Tooth	✓	
dbr:Fare	dbr:Bus	✓	
dbr:Fascism	dbr:Adolf_Hitler	✓	
dbr:Feather	dbr:Bird		✓
dbr:Feces	dbr:Shit	✓	
dbr:Fee	dbr:Money	✓	
dbr:Felony	dbr:Crime	✓	
dbr:Femininity	dbr:Masculinity	✓	
dbr:Filly	dbr:Horse	✓	
dbr:Finch	dbr:Bird	✓	
dbr:Fir	dbr:Tree	✓	
dbr:Firearm	dbr:Gun	✓	
dbr:Fish	dbr:Sea		✓
dbr:Flame	dbr:Fire	✓	
dbr:Flavor	dbr:Taste	✓	
dbr:Flirting	dbr:Girl	✓	
dbr:Flood	dbr:Water	✓	
dbr:Flounder	dbr:Fish	✓	
dbr:Flue	dbr:Chimney	✓	
dbr:Fluid	dbr:Liquid	✓	
dbr:Fluid	dbr:Water	✓	
dbr:Foal	dbr:Horse	✓	
dbr:Foam	dbr:Beer	✓	
dbr:Fog	dbr:Mist	✓	
dbr:Food_grain	dbr:Wheat	✓	
dbr:Ford_Motor_Company	dbr:Automobile	✓	
dbr:Forest	dbr:Tree		✓
dbr:Forestry	dbr:Tree	✓	
dbr:Fortnight	dbr:Week	✓	
dbr:Foundry	dbr:Iron	✓	
dbr:Fountain	dbr:Water	✓	
dbr:Franc	dbr:France	✓	
dbr:Franc	dbr:Money	✓	
dbr:Freezing	dbr:Cold	✓	
dbr:Freight_transport	dbr:Cargo	✓	
dbr:Friendship	dbr:Enemy	✓	
dbr:Frock	dbr:Dress	✓	
dbr:Frost	dbr:Cold	✓	
dbr:Fuel	dbr:Gasoline	✓	
dbr:Furlong	dbr:Mile	✓	
dbr:Galaxy	dbr:Star	✓	
dbr:Gale	dbr:Wind	✓	
dbr:Gallon	dbr:Gasoline	✓	
dbr:Gallon	dbr:Pint	✓	
dbr:Gallows	dbr:Hanging	✓	
dbr:Gambling	dbr:Money	✓	
dbr:Garden	dbr:Flower	✓	
dbr:Gasoline	dbr:Automobile	✓	
dbr:Gelding	dbr:Horse	✓	
dbr:Gentleman	dbr:Lady	✓	
dbr:Geology	dbr:Rock_(geology)	✓	
dbr:Geometry	dbr:Mathematics	✓	
dbr:Geranium	dbr:Flower		✓
dbr:Geranium	dbr:Plant	✓	
dbr:Gibbon	dbr:Monkey	✓	
dbr:Gift	dbr:Christmas	✓	
dbr:Gingiva	dbr:Tooth	✓	
dbr:Giraffe	dbr:Neck	✓	
dbr:Girder	dbr:Steel	✓	
dbr:Glitter	dbr:Gold	✓	
dbr:Gloom	dbr:Darkness	✓	
dbr:Glove	dbr:Hand	✓	
dbr:Glucose	dbr:Sugar	✓	
dbr:Gnat	dbr:Fly	✓	
dbr:Goat	dbr:Milk	✓	
dbr:Gorilla	dbr:Ape	✓	
dbr:Gram	dbr:Weight	✓	
dbr:Granite	dbr:Rock_(geology)	✓	
dbr:Grape	dbr:Wine	✓	
dbr:Green	dbr:Grass	✓	
dbr:Greenhouse	dbr:Plant	✓	
dbr:Greyhound	dbr:Dog	✓	
dbr:Groin	dbr:Leg	✓	
dbr:Gull	dbr:Bird	✓	
dbr:Gull	dbr:Sea	✓	
dbr:Haggis	dbr:Scotland	✓	
dbr:Hair_roller	dbr:Hair	✓	
dbr:Hammock	dbr:Bed	✓	
dbr:Handkerchief	dbr:Nose	✓	
dbr:Harbor	dbr:Ship	✓	
dbr:Hare	dbr:Rabbit		✓

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:Harmony	dbr:Music	✓	
dbr:Harpoon	dbr:Spear	✓	
dbr:Harpoon	dbr:Whale	✓	
dbr:Harpsichord	dbr:Music		✓
dbr:Hatred	dbr:Love	✓	
dbr:Haze	dbr:Mist	✓	
dbr:Head_teacher	dbr:School		✓
dbr:Headache	dbr:Pain	✓	
dbr:Headgear	dbr:Hat	✓	
dbr:Hearse	dbr:Death	✓	
dbr:Heat	dbr:Cold	✓	
dbr:Heel	dbr:Shoe	✓	
dbr:Hemoglobin	dbr:Blood	✓	
dbr:Hemorrhoid	dbr:Blood	✓	
dbr:Herd	dbr:Cattle	✓	
dbr:Homicide	dbr:Death	✓	
dbr:Homicide	dbr:Murder	✓	
dbr:Homosexuality	dbr:Queer	✓	
dbr:Hoof	dbr:Horse	✓	
dbr:Hops	dbr:Beer		✓
dbr:Hornpipe	dbr:Dance	✓	
dbr:Hostel	dbr:Youth	✓	
dbr:Hostility	dbr:Enemy	✓	
dbr:Hue	dbr:Color	✓	
dbr:Human_gastrointestinal_tract	dbr:Gut_(anatomy)	✓	
dbr:Human_height	dbr:Height		✓
dbr:Humour	dbr:Laughter	✓	
dbr:Husband	dbr:Wife		✓
dbr:Hypothalamus	dbr:Brain	✓	
dbr:Icicle	dbr:Cold	✓	
dbr:Impressionism	dbr:Art	✓	
dbr:Imprisonment	dbr:Prison	✓	
dbr:Inch	dbr:Mile	✓	
dbr:Inn	dbr:Pub	✓	
dbr:Ireland	dbr:Green		✓
dbr:Irritation	dbr:Itch	✓	
dbr:Jaundice	dbr:Yellow	✓	
dbr:Jeep	dbr:Automobile	✓	
dbr:Jockey	dbr:Horse	✓	
dbr:Johannes_Brahms	dbr:Music	✓	
dbr:Joke	dbr:Laughter	✓	
dbr:Judge	dbr:Jury	✓	
dbr:Jug	dbr:Milk	✓	
dbr:Jug	dbr:Water	✓	
dbr:Juice	dbr:Fruit	✓	
dbr:July	dbr:August	✓	
dbr:June	dbr:July	✓	
dbr:Jupiter	dbr:Planet	✓	
dbr:Karate	dbr:Judo	✓	
dbr:Kennel	dbr:Dog		✓
dbr:Khaki	dbr:Army	✓	
dbr:Khaki	dbr:Shorts	✓	
dbr:Kipper	dbr:Fish		✓
dbr:Kitten	dbr:Cat	✓	
dbr:Knight	dbr:Armour	✓	
dbr:Knitting	dbr:Wool	✓	
dbr:Labyrinth	dbr:Maze	✓	
dbr:Lake	dbr:Water	✓	
dbr:Lampshade	dbr:Light	✓	
dbr:Lantern	dbr:Light	✓	
dbr:Lard	dbr:Butter	✓	
dbr:Lard	dbr:Fat	✓	
dbr:Larder	dbr:Food	✓	
dbr:Laundry	dbr:Washing	✓	
dbr:Leaf	dbr:Tree	✓	
dbr:Leak	dbr:Water		✓
dbr:Ledger	dbr:Book	✓	
dbr:Leek	dbr:Wales	✓	
dbr:Leg	dbr:Arm	✓	
dbr:Lemonade	dbr:Drink	✓	
dbr:Lent	dbr:Easter		✓
dbr:Lesson	dbr:School	✓	
dbr:Library	dbr:Book	✓	
dbr:License	dbr:Automobile	✓	
dbr:License	dbr:Driving	✓	
dbr:Lichen	dbr:Moss	✓	
dbr:Licking	dbr:Dog	✓	
dbr:Lie	dbr:Truth	✓	
dbr:Lieutenant	dbr:Army	✓	
dbr:Lightning	dbr:Thunder	✓	
dbr:Limp	dbr:Leg	✓	
dbr:Lion	dbr:Tiger		✓
dbr:Lip	dbr:Mouth	✓	
dbr:Literature	dbr:Book	✓	

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:Litre	dbr:Pint	✓	
dbr:Locomotive	dbr:Train	✓	
dbr:Loft	dbr:Attic	✓	
dbr:Ludwig_van_Beethoven	dbr:Music	✓	
dbr:Mackerel	dbr:Fish		✓
dbr:Mackintosh	dbr:Rain	✓	
dbr:Mallet	dbr:Hammer	✓	
dbr:Mare	dbr:Horse	✓	
dbr:Margarine	dbr:Butter		✓
dbr:Martyr	dbr:Saint	✓	
dbr:Mat	dbr:Door	✓	
dbr:Matchbox	dbr:Match		✓
dbr:Matron	dbr:Hospital	✓	
dbr:Mattress	dbr:Bed	✓	
dbr:Mauve	dbr:Color	✓	
dbr:Mauve	dbr:Purple	✓	
dbr:Meal	dbr:Food	✓	
dbr:Mercenary	dbr:Money	✓	
dbr:Metre	dbr:Yard	✓	
dbr:Milkmaid	dbr:Cattle	✓	
dbr:Miner	dbr:Coal	✓	
dbr:Mire	dbr:Mud	✓	
dbr:Miser	dbr:Money	✓	
dbr:Mist	dbr:Fog	✓	
dbr:Monastery	dbr:Monk	✓	
dbr:Monkey	dbr:Ape	✓	
dbr:Month	dbr:Year	✓	
dbr:Moorland	dbr:Heath	✓	
dbr:Mortgage_loan	dbr:House	✓	
dbr:Mosaic	dbr:Pattern	✓	
dbr:Mount_Everest	dbr:Mountain	✓	
dbr:Mountain	dbr:Hill	✓	
dbr:Mourning	dbr:Death	✓	
dbr:Mousse	dbr:Chocolate		✓
dbr:Nap	dbr:Sleep	✓	
dbr:Napalm	dbr:Bomb	✓	
dbr:Napalm	dbr:Vietnam	✓	
dbr:Nape	dbr:Neck	✓	
dbr:Narcissus_(plant)	dbr:Yellow	✓	
dbr:Navigation	dbr:Ship	✓	
dbr:Need	dbr:Want	✓	
dbr:Netball	dbr:Game	✓	
dbr:Neurology	dbr:Brain	✓	
dbr:Newt	dbr:Frog	✓	
dbr:Nostril	dbr:Nose	✓	
dbr:Noun	dbr:Verb	✓	
dbr:Novel	dbr:Book	✓	
dbr:Oar	dbr:Boat	✓	
dbr:Ocean	dbr:Sea	✓	
dbr:Octave	dbr:Music	✓	
dbr:Octopus	dbr:8_(number)	✓	
dbr:Offspring	dbr:Child	✓	
dbr:Omelette	dbr:Egg	✓	
dbr:Onyx	dbr:Rock_(geology)		✓
dbr:Optimism	dbr:Hope	✓	
dbr:Optimism	dbr:Pessimism	✓	
dbr:Orange_(fruit)	dbr:Apple		✓
dbr:Orchard	dbr:Apple	✓	
dbr:Ore	dbr:Gold	✓	
dbr:Orphan	dbr:Child	✓	
dbr:Osprey	dbr:Bird	✓	
dbr:Ostrich	dbr:Bird	✓	
dbr:Ostrich	dbr:Feather	✓	
dbr:Overdraft	dbr:Bank	✓	
dbr:Oxygen	dbr:Atmosphere_of_Earth		✓
dbr:Pajamas	dbr:Bed	✓	
dbr:Palate	dbr:Mouth	✓	
dbr:Palate	dbr:Taste	✓	
dbr:Pansy	dbr:Flower	✓	
dbr:Pantry	dbr:Food	✓	
dbr:Parent	dbr:Father	✓	
dbr:Paris	dbr:France	✓	
dbr:Parking	dbr:Automobile	✓	
dbr:Parrot	dbr:Bird	✓	
dbr:Parson	dbr:Vicar	✓	
dbr:Patella	dbr:Leg	✓	
dbr:Patient	dbr:Hospital	✓	
dbr:Patriotism	dbr:Country	✓	
dbr:Paw	dbr:Cat	✓	
dbr:Paw	dbr:Dog		✓
dbr:Payment	dbr:Money	✓	
dbr:Peace	dbr:War	✓	
dbr:Pear	dbr:Apple	✓	
dbr:Peeler	dbr:Potato	✓	

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:Pen	dbr:Ink	✓	
dbr:Peninsula	dbr:Island	✓	
dbr:Penny	dbr:Money	✓	
dbr:Perfume	dbr:Odor	✓	
dbr:Perjury	dbr:Lie		✓
dbr:Person	dbr:People	✓	
dbr:Pet	dbr:Dog	✓	
dbr:Petal	dbr:Flower	✓	
dbr:Petrel	dbr:Bird	✓	
dbr:Physics	dbr:Chemistry	✓	
dbr:Pillow	dbr:Bed	✓	
dbr:Pillow	dbr:Sleep	✓	
dbr:Pineapple	dbr:Fruit	✓	
dbr:Pint	dbr:Beer	✓	
dbr:Pint	dbr:Milk		✓
dbr:Pistol	dbr:Gun	✓	
dbr:Plaice	dbr:Fish	✓	
dbr:Planet	dbr:Mars	✓	
dbr:Planet	dbr:Star	✓	
dbr:Plankton	dbr:Sea		✓
dbr:Plant_sap	dbr:Tree		✓
dbr:Playground	dbr:Child	✓	
dbr:Pleat	dbr:Skirt	✓	
dbr:Police_officer	dbr:Law	✓	
dbr:Pollen	dbr:Flower	✓	
dbr:Poly(methyl_methacrylate)	dbr:Glass	✓	
dbr:Polyethylene	dbr:Plastic	✓	
dbr:Pony	dbr:Horse	✓	
dbr:Porpoise	dbr:Fish	✓	
dbr:Porridge	dbr:Oat	✓	
dbr:Port	dbr:Ship		✓
dbr:Portrait	dbr:Image	✓	
dbr:Pottage	dbr:Soup	✓	
dbr:Prayer	dbr:God		✓
dbr:Prison_officer	dbr:Prison	✓	
dbr:Prisoner	dbr:Prison	✓	
dbr:Prisoner	dbr:War	✓	
dbr:Profession	dbr:Job	✓	
dbr:Prosperity	dbr:Wealth	✓	
dbr:Protestantism	dbr:Catholicism	✓	
dbr:Pub	dbr:Drink	✓	
dbr:Pump	dbr:Water	✓	
dbr:Pupil	dbr:Eye	✓	
dbr:Puppy	dbr:Dog	✓	
dbr:Putty	dbr:Window	✓	
dbr:Quadrupedalism	dbr:4_(number)	✓	
dbr:Quarantine	dbr:Dog	✓	
dbr:Quarto	dbr:Paper	✓	
dbr:Quilt	dbr:Bed	✓	
dbr:Quinine	dbr:Drug	✓	
dbr:Racket_(sports_equipment)	dbr:Tennis		✓
dbr:Rainbow	dbr:Color	✓	
dbr:Rayon	dbr:Nylon	✓	
dbr:Reagent	dbr:Chemistry	✓	
dbr:Recipe	dbr:Food	✓	
dbr:Recital	dbr:Music	✓	
dbr:Rectangle	dbr:Square		✓
dbr:Refectory	dbr:Food	✓	
dbr:Regiment	dbr:Army	✓	
dbr:Religion	dbr:God	✓	
dbr:Remittance	dbr:Money	✓	
dbr:Reply	dbr:Answer	✓	
dbr:Reptile	dbr:Snake	✓	
dbr:Reservoir	dbr:Water	✓	
dbr:Resort	dbr:Holiday	✓	
dbr:Resurrection	dbr:Christ		✓
dbr:Rhythm	dbr:Music	✓	
dbr:Ribbon	dbr:Hair	✓	
dbr:Rifle	dbr:Gun	✓	
dbr:Rifleman	dbr:Gun	✓	
dbr:Rigging	dbr:Ship	✓	
dbr:Rock_(geology)	dbr:Sea	✓	
dbr:Rodent	dbr:Rat	✓	
dbr:Roe	dbr:Fish	✓	
dbr:Rome	dbr:Italy	✓	
dbr:Romper_suit	dbr:Infant	✓	
dbr:Ruins	dbr:Castle	✓	
dbr:Saddle	dbr:Horse	✓	
dbr:Salad	dbr:Lettuce	✓	
dbr:Salary	dbr:Money	✓	
dbr:Sandal	dbr:Foot	✓	
dbr:Sandal	dbr:Shoe	✓	
dbr:Sarcasm	dbr:Wit	✓	
dbr:Sari	dbr:India	✓	

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:Satin	dbr:Silk	✓	
dbr:Saturn	dbr:Planet	✓	
dbr:Sausage	dbr:Meat	✓	
dbr:Saving	dbr:Money	✓	
dbr:Sawdust	dbr:Wood	✓	
dbr:Scaffolding	dbr:Building		✓
dbr:Scalpel	dbr:Knife	✓	
dbr:Scarf	dbr:Neck	✓	
dbr:Schooner	dbr:Ship	✓	
dbr:Scone	dbr:Butter	✓	
dbr:Sculpture	dbr:Art	✓	
dbr:Scythe	dbr:Grass	✓	
dbr:Seaman	dbr:Sailor	✓	
dbr:Seascape	dbr:Landscape	✓	
dbr:Seat	dbr:Chair	✓	
dbr:Seat_belt	dbr:Automobile	✓	
dbr:Seat_belt	dbr:Safety	✓	
dbr:Second	dbr:Time	✓	
dbr:Sect	dbr:Religion	✓	
dbr:Serfdom	dbr:Slavery		✓
dbr:Setter	dbr:Dog	✓	
dbr:Setter	dbr:Red		✓
dbr:Shilling	dbr:Penny		✓
dbr:Ship	dbr:Sea	✓	
dbr:Shivering	dbr:Cold	✓	
dbr:Shoe	dbr:Sock	✓	
dbr:Shooting	dbr:Gun	✓	
dbr:Shore	dbr:Sea	✓	
dbr:Shoulder	dbr:Arm	✓	
dbr:Shovel	dbr:Spade	✓	
dbr:Shrub	dbr:Tree	✓	
dbr:Sickle	dbr:Scythe	✓	
dbr:Signalman	dbr:Rail_transport	✓	
dbr:Ski	dbr:Snow	✓	
dbr:Skiing	dbr:Snow	✓	
dbr:Skull	dbr:Head	✓	
dbr:Sled	dbr:Snow	✓	
dbr:Sleep	dbr:Bed	✓	
dbr:Slipper	dbr:Shoe	✓	
dbr:Smog	dbr:Fog	✓	
dbr:Sneeze	dbr:Cough	✓	
dbr:Snipe	dbr:Bird		✓
dbr:Snoring	dbr:Sleep	✓	
dbr:Solicitor	dbr:Law	✓	
dbr:Solicitor	dbr:Lawyer	✓	
dbr:Somerset	dbr:Cider	✓	
dbr:Sonata	dbr:Music	✓	
dbr:Soot	dbr:Black	✓	
dbr:Soot	dbr:Chimney	✓	
dbr:Sound	dbr:Noise	✓	
dbr:Spade	dbr:Shovel		✓
dbr:Spelling	dbr:Word	✓	
dbr:Splinter	dbr:Wood	✓	
dbr:Spoon	dbr:Fork	✓	
dbr:Spouse	dbr:Husband	✓	
dbr:Spouse	dbr:Wife	✓	
dbr:Square	dbr:Circle	✓	
dbr:Stable	dbr:Horse	✓	
dbr:Stallion	dbr:Horse	✓	
dbr:Stallion	dbr:Mare	✓	
dbr:Stamen	dbr:Flower	✓	
dbr:Stanza	dbr:Poetry	✓	
dbr:Starling	dbr:Bird	✓	
dbr:Stationery	dbr:Paper	✓	
dbr:Strategy	dbr:Plan	✓	
dbr:Structure	dbr:Building	✓	
dbr:Suburb	dbr:Town	✓	
dbr:Suffering	dbr:Pain	✓	
dbr:Sugar	dbr:Sweetness	✓	
dbr:Suggestion	dbr:Idea	✓	
dbr:Suicide	dbr:Death	✓	
dbr:Summation	dbr:Mathematics	✓	
dbr:Summer	dbr:Winter	✓	
dbr:Sun	dbr:Moon	✓	
dbr:Swallow	dbr:Bird	✓	
dbr:Sweetness	dbr:Sugar	✓	
dbr:Symphony	dbr:Music	✓	
dbr:Syrup	dbr:Treacle	✓	
dbr:Tadpole	dbr:Frog	✓	
dbr:Tank	dbr:War	✓	
dbr:Tankard	dbr:Beer	✓	
dbr:Tar	dbr:Road	✓	
dbr:Tarmac	dbr:Road	✓	
dbr:Teacup	dbr:Saucer	✓	

Table B.1: Semantic Assocation Ground Truth Dataset (continued)

stimulus entity	response entity	train	test
dbr:Telegraphy	dbr:News	✓	
dbr:Terrain	dbr:Land	✓	
dbr:Thatching	dbr:Roof	✓	
dbr:The_Reverend	dbr:Vicar	✓	
dbr:Thermodynamics	dbr:Heat	✓	
dbr:Thermometer	dbr:Heat	✓	
dbr:Thermometer	dbr:Temperature	✓	
dbr:Thigh	dbr:Leg	✓	
dbr:Tickling	dbr:Laughter	✓	
dbr:Tide	dbr:Sea	✓	
dbr:Tiger	dbr:Lion	✓	
dbr:Tile	dbr:Roof	✓	
dbr:Timer	dbr:Clock	✓	
dbr:Tinsel	dbr:Christmas		✓
dbr:Toe	dbr:Foot	✓	
dbr:Tomato	dbr:Red	✓	
dbr:Tomb	dbr:Grave	✓	
dbr:Ton	dbr:Weight	✓	
dbr:Toothache	dbr:Dentist	✓	
dbr:Toothache	dbr:Pain	✓	
dbr:Toothbrush	dbr:Tooth	✓	
dbr:Tornado	dbr:Wind	✓	
dbr:Toy	dbr:Child	✓	
dbr:Tractor	dbr:Farm	✓	
dbr:Tram	dbr:Bus	✓	
dbr:Tranquillity	dbr:Peace	✓	
dbr:Transport	dbr:Bus	✓	
dbr:Tree	dbr:Leaf		✓
dbr:Troop	dbr:Army	✓	
dbr:Troop	dbr:Soldier	✓	
dbr:Tropical_cyclone	dbr:Wind	✓	
dbr:Trot	dbr:Horse		✓
dbr:Truth	dbr:Lie		✓
dbr:Tulip	dbr:Flower	✓	
dbr:Twig	dbr:Branch	✓	
dbr:Twig	dbr:Tree	✓	
dbr:Udder	dbr:Cattle	✓	
dbr:Umbrella	dbr:Rain	✓	
dbr:Uncle	dbr:Aunt	✓	
dbr:Underlay	dbr:Carpet	✓	
dbr:Uterus	dbr:Infant	✓	
dbr:Vacation	dbr:Holiday	✓	
dbr:Van	dbr:Automobile	✓	
dbr:Vase	dbr:Flower	✓	
dbr:Veal	dbr:Ham	✓	
dbr:Veal	dbr:Meat	✓	
dbr:Vehicle	dbr:Automobile	✓	
dbr:Velocity	dbr:Speed	✓	
dbr:Village	dbr:Town	✓	
dbr:Vine	dbr:Grape	✓	
dbr:Vinegar	dbr:Salt	✓	
dbr:Violin	dbr:Music	✓	
dbr:Vocation	dbr:Job	✓	
dbr:Wage	dbr:Money	✓	
dbr:Wallpaper	dbr:Paint	✓	
dbr:Wand	dbr:Fairy	✓	
dbr:Wardrobe	dbr:Clothing	✓	
dbr:Waste_container	dbr:Municipal_solid_waste	✓	
dbr:Wave	dbr:Sea	✓	
dbr:Wealth	dbr:Money	✓	
dbr:Weasel	dbr:Stoat	✓	
dbr:Weather	dbr:Rain	✓	
dbr:Week	dbr:Day		✓
dbr:Whisk	dbr:Egg	✓	
dbr:Whole_note	dbr:Music	✓	
dbr:Wife	dbr:Husband	✓	
dbr:Wig	dbr:Hair	✓	
dbr:Wildebeest	dbr:Animal	✓	
dbr:Wing	dbr:Bird	✓	
dbr:Winter	dbr:Snow	✓	
dbr:Wireless	dbr:Radio	✓	
dbr:Wolfgang_Amadeus_Mozart	dbr:Music	✓	
dbr:Woman	dbr:Man	✓	
dbr:Wool	dbr:Sheep	✓	
dbr:Yacht	dbr:Boat	✓	
dbr:Year	dbr:Month	✓	
dbr:Zoo	dbr:Animal	✓	
dbr:Zoology	dbr:Animal		✓

C

EVALUATION OF ALL RDF2VEC MODELS

Table C.1: Comparison of all RDF2Vec models used as baselines (cf. Section 10.4.2). The models are available under: http://data.dws.informatik.uni-mannheim.de/rdf2vec/models/DBpedia/ (All values in percent.)

			Recall@k							
method & model	MRR	NDCG	1	2	3	5	10	25	50	100
sim:										
2015-10/4depth/cbow/DB2Vec_cbow_200_5_5_2_500	7.4	10.8	4.2	6.9	8.3	12.5	12.5	16.7	18.1	**25.0**
2015-10/4depth/cbow/DB2Vec_cbow_500_5_5_2_500	**8.6**	**11.5**	4.2	**9.7**	**12.5**	**13.9**	**15.3**	**18.1**	**19.4**	22.2
2015-10/4depth/skipgram/DB2Vec_sg_200_5_5_15_2_500	7.4	10.3	4.2	6.9	9.7	11.1	13.9	13.9	16.7	22.2
2015-10/4depth/skipgram/DB2Vec_sg_500_5_5_15_2_500	7.8	10.7	4.2	6.9	**12.5**	12.5	13.9	15.3	16.7	22.2
2015-10/8depth/cbow/DB2Vec_cbow_200_5_5_4_500	7.2	8.8	**5.6**	6.9	8.3	8.3	9.7	12.5	13.9	15.3
2015-10/8depth/cbow/DB2Vec_cbow_500_5_5_4_50	5.0	8.0	2.8	5.6	6.9	6.9	6.9	9.7	15.3	22.2
2015-10/8depth/skipgram/DB2Vec_sg_200_5_5_15_4_500	7.8	10.3	**5.6**	6.9	8.3	11.1	12.5	13.9	18.1	20.8
2015-10/8depth/skipgram/DB2Vec_sg_500_5_5_15_4_500	6.2	8.1	4.2	5.6	6.9	9.7	11.1	12.5	15.3	15.3
2015-10/noTypes/db2vec/sg_200_5_25_5	3.7	5.4	2.8	2.8	4.2	4.2	5.6	9.7	11.1	12.5
2016-04/predicateFrequency/db2vec/sg_200_5_25_5	1.6	1.8	1.4	1.4	1.4	1.4	2.8	2.8	2.8	2.8
2016-04/uniform/db2vec/sg_200_5_25_5	3.7	5.4	1.4	5.6	5.6	5.6	5.6	8.3	8.3	12.5
pred:										
2015-10/4depth/cbow/DB2Vec_cbow_200_5_5_2_500	8.8	13.1	**5.6**	8.3	11.1	12.5	15.3	16.7	22.2	**31.9**
2015-10/4depth/cbow/DB2Vec_cbow_500_5_5_2_500	**10.3**	**14.4**	**5.6**	**12.5**	**13.9**	**15.3**	**16.7**	**19.4**	23.6	**31.9**
2015-10/4depth/skipgram/DB2Vec_sg_200_5_5_15_2_500	8.9	13.1	**5.6**	6.9	11.1	**15.3**	15.3	16.7	23.6	**31.9**
2015-10/4depth/skipgram/DB2Vec_sg_500_5_5_15_2_500	9.1	13.4	**5.6**	8.3	9.7	13.9	15.3	18.1	**25.0**	**31.9**
2015-10/8depth/cbow/DB2Vec_cbow_200_5_5_4_500	6.6	8.8	4.2	6.9	6.9	9.7	11.1	12.5	13.9	18.1
2015-10/8depth/cbow/DB2Vec_cbow_500_5_5_4_50	5.4	9.2	2.8	5.6	6.9	6.9	8.3	13.9	16.7	26.4
2015-10/8depth/skipgram/DB2Vec_sg_200_5_5_15_4_500	8.5	11.4	**5.6**	9.7	9.7	11.1	12.5	16.7	18.1	23.6
2015-10/8depth/skipgram/DB2Vec_sg_500_5_5_15_4_500	6.4	9.1	4.2	5.6	6.9	8.3	11.1	12.5	15.3	20.8
2015-10/noTypes/db2vec/sg_200_5_25_5	4.0	6.3	2.8	2.8	4.2	4.2	6.9	11.1	13.9	16.7
2016-04/predicateFrequency/db2vec/sg_200_5_25_5	1.4	1.9	1.4	1.4	1.4	1.4	1.4	1.4	2.8	4.2
2016-04/uniform/db2vec/sg_200_5_25_5	3.8	5.8	1.4	5.6	5.6	5.6	5.6	9.7	12.5	13.9
simpred:										
2015-10/4depth/cbow/DB2Vec_cbow_200_5_5_2_500	6.9	12.1	2.8	4.2	**9.7**	**11.1**	**16.7**	20.8	20.8	34.7
2015-10/4depth/cbow/DB2Vec_cbow_500_5_5_2_500	**8.4**	**13.7**	**5.6**	5.6	6.9	**11.1**	**16.7**	20.8	**31.9**	**36.1**
2015-10/4depth/skipgram/DB2Vec_sg_200_5_5_15_2_500	5.2	10.8	2.8	2.8	4.2	6.9	9.7	**22.2**	23.6	**36.1**
2015-10/4depth/skipgram/DB2Vec_sg_500_5_5_15_2_500	6.0	11.2	4.2	4.2	5.6	8.3	8.3	13.9	22.2	**36.1**
2015-10/8depth/cbow/DB2Vec_cbow_200_5_5_4_500	4.2	6.9	2.8	4.2	4.2	4.2	5.6	8.3	13.9	19.4
2015-10/8depth/cbow/DB2Vec_cbow_500_5_5_4_50	4.2	7.6	2.8	2.8	4.2	5.6	6.9	9.7	18.1	23.6
2015-10/8depth/skipgram/DB2Vec_sg_200_5_5_15_4_500	7.7	12.2	**5.6**	**6.9**	8.3	8.3	11.1	15.3	26.4	33.3
2015-10/8depth/skipgram/DB2Vec_sg_500_5_5_15_4_500	5.8	10.0	4.2	4.2	4.2	6.9	11.1	13.9	19.4	29.2
2015-10/noTypes/db2vec/sg_200_5_25_5	3.6	5.9	2.8	2.8	2.8	2.8	5.6	9.7	11.1	16.7
2016-04/predicateFrequency/db2vec/sg_200_5_25_5	0.3	0.9	0.0	0.0	0.0	0.0	1.4	1.4	1.4	4.2
2016-04/uniform/db2vec/sg_200_5_25_5	3.8	6.0	2.8	4.2	4.2	4.2	4.2	6.9	8.3	16.7

BIBLIOGRAPHY

[1] A. Abele, J. P. McCrae, P. Buitelaar, A. Jentzsch, and R. Cyganiak. *Linking Open Data cloud diagram*. 2017. URL: http://lod-cloud.net/ (cit. on pp. xvi, 3, 7, 18).

[2] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Pasça, and A. Soroa. "A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches." In: *Proc. of the NAACL 2009*. June. Boulder, Colorado, US: Association for Computational Linguistics, 2009, pp. 19–27 (cit. on pp. 27, 31).

[3] L. von Ahn. "Games with a Purpose." In: *IEEE Computer Society - Computer* 39.6 (June 2006), pp. 92–94. DOI: 10.1109/MC.2006.196 (cit. on p. xv).

[4] L. von Ahn and L. Dabbish. "Labeling Images with a Computer Game." In: *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*. Vienna, Austria: ACM New York, NY, USA, 2004, pp. 319–326. URL: http://portal.acm.org/citation.cfm?id=985692.985733%7B%5C&%7Dtype=series (cit. on p. 42).

[5] L. von Ahn and L. Dabbish. "Designing games with a purpose." In: *Communications of the ACM* 51.8 (Aug. 2008), pp. 58–67. DOI: 10.1145/1378704.1378719 (cit. on pp. xv, 42, 45, 48, 59).

[6] J. R. Anderson. "A Spreading Activation Theory of Memory." In: *Journal of Verbal Learning and Verbal Behavior* 22.3 (June 1983), pp. 261–295. DOI: 10.1016/S0022-5371(83)90201-3 (cit. on p. 23).

[7] Aristotle. *De Memoria et Reminiscentia* (cit. on p. 19).

[8] V. Arvind, B. Das, and J. Köbler. "The Space Complexity of k-Tree Isomorphism." In: *Algorithms and Computation* 4835 (2007), pp. 822–833. URL: http://www.springerlink.com/index/h554826786533um6.pdf (cit. on p. 94).

[9] M. van Assem, A. Gangemi, and G. Schreiber. "Conversion of WordNet to a standard RDF/OWL representation." In: *International Conference on Languages Resources and Evaluation (LREC)*. 2006, pp. 237–242. URL: http://www.lrec-conf.org/proceedings/lrec2006/pdf/165_pdf.pdf (cit. on pp. 26, 27).

[10] L. Babai. *Graph Isomorphism in Quasipolynomial Time*. Tech. rep. 2016, pp. 1–89. arXiv: 1512.03547 (cit. on p. 94).

[11] L. Babai. "Graph Isomorphism in Quasipolynomial Time." In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing - STOC '16*. Cambridge, MA, USA: ACM, 2016, pp. 684–697. DOI: 10.1145/2897518.2897542 (cit. on p. 94).

[12] L. Babai and E. M. Luks. "Canonical labeling of graphs." In: *Proceedings of the fifteenth annual ACM symposium on Theory of Computing - STOC '83*. ACM, 1983, pp. 171–183. DOI: 10.1145/800061.808746 (cit. on p. 94).

[13] A. Baddeley, M. W. Eysenck, and M. C. Anderson. *Memory*. Psychology Press, 2009, p. 464 (cit. on pp. 4, 19).

[14] A. Balmin, V. Hristidis, and Y. Papakonstantinou. "ObjectRank: Authority-Based Keyword Search in Databases." In: *Proc. of the 13th International Conference on Very Large Data Bases, VLDB Endowment*. 2004, pp. 564–575 (cit. on p. 29).

[15] H. Bast, B. Buchhold, and E. Haussmann. "Relevance Scores for Triples from Type-Like Relations." In: *SIGIR 2015 - Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015), pp. 243–252. DOI: 10.1145/2766462.2767734 (cit. on p. 31).

[16] H. Bast, B. Buchhold, and E. Haussmann. "Semantic Search on Text and Knowledge Bases." In: *Foundations and Trends in Information Retrieval* 10.1 (2016), pp. 119–271. DOI: 10.1561/1500000032 (cit. on p. 31).

[17] D. Beckett. *RDF/XML Syntax Specification (Revised)*. W3C Recommendation. W3C, Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ (cit. on p. xvi).

[18] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, and S. Schlobach. "LOD Laundromat: A Uniform Way of Publishing Other People's Dirty Data." In: *Proceedings of the International Semantic Web Conference*. Vol. 8796. 2014, pp. 213–228. DOI: 10.1007/978-3-319-11964-9. arXiv: arXiv:1408.0926v1 (cit. on p. 27).

[19] T. Berners-Lee. "Plenary Talk: W3 future directions." In: *1st International World Wide Web Conference (WWW)*. CERN. Geneva, Switzerland: W3C, Sept. 1994. URL: https://www.w3.org/Talks/WWW94Tim/ (cit. on p. 13).

[20] T. Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. Harper Paperbacks, 2000, p. 256 (cit. on pp. xvi, 13).

[21] T. Berners-Lee. *Linked Data - Design Issues*. 2006. URL: http://www.w3.org/DesignIssues/LinkedData.html (cit. on pp. 6, 17, 18, 41).

[22] T. Berners-Lee, R. T. Fielding, and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax*. RFC 2396. RFC Editor, Aug. 1998. URL: http://www.rfc-editor.org/rfc/rfc2396.txt (cit. on pp. xvi, 7, 14).

[23] T. Berners-Lee, R. T. Fielding, and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. STD 66. RFC Editor, Jan. 2005. URL: http://www.rfc-editor.org/rfc/rfc3986.txt (cit. on pp. xvi, 7, 14).

[24] T. Berners-Lee, J. Hendler, and O. Lassila. "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities." In: *Scientific American* 284.5 (May 2001), pp. 34–43. URL: http://wayback.archive.org/web/20070713230811/http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 (cit. on pp. 3, 6, 13).

[25] M. Birbeck and S. McCarron. *CURIE Syntax 1.0*. W3C Note. W3C, Dec. 2010. URL: http://www.w3.org/TR/2010/NOTE-curie-20101216 (cit. on pp. xv, 7, 14).

[26] C. Bizer, T. Heath, and T. Berners-Lee. "Linked Data - The Story So Far." In: *International Journal on Semantic Web and Information Systems* 5.3 (Jan. 2009), pp. 1–22. DOI: 10.4018/jswis.2009081901 (cit. on pp. 3, 18).

[27] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. "DBpedia - A crystallization point for the Web of Data." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 7.3 (Sept. 2009), pp. 154–165. DOI: 10.1016/j.websem.2009.07.002 (cit. on pp. 3, 18).

[28] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation." In: *Advances in Neural Information Processing Systems*. 2002, pp. 601–608 (cit. on p. 32).

[29] T. Bobić, J. Waitelonis, and H. Sack. "FRanCo - A Ground Truth Corpus for Fact Ranking Evaluation." In: *SumPre 2015 - 1st International Workshop on Summarizing and Presenting Entities and Ontologies @ ESWC 2015*. 2015, pp. 1–12 (cit. on pp. 25, 27).

[30] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. "Freebase: a collaboratively created graph database for structuring human knowledge." In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1247–1250. DOI: 10.1145/1376616.1376746 (cit. on pp. 18, 43, 56).

[31] S. Brin and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." In: *Computer Networks and ISDN Systems* 30.1-7 (Apr. 1998), pp. 107–117. DOI: 10.1016/S0169-7552(98)00110-X (cit. on p. 29).

[32] A. Budanitsky and G. Hirst. "Evaluating WordNet-based Measures of Lexical Semantic Relatedness." In: *Computational Linguistics* 32.1 (Mar. 2006), pp. 13–47. DOI: 10.1162/coli.2006.32.1.13 (cit. on pp. 26, 30).

[33] J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. "NASARI: a Novel Approach to a Semantically-Aware Representation of Items." In: *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, Denver, Colorado, May 31 - June 5, 2015* (2015), pp. 567–577. URL: http://lcl.uniroma1.it/nasari/. (cit. on pp. 32, 33, 135).

[34] J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. "NASARI: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities." In: *Artificial Intelligence* 240 (2016), pp. 36–64. DOI: 10.1016/j.artint.2016.07.005 (cit. on pp. 32, 33, 135).

[35] M. Campbell, a. J. Hoane Jr., and F.-h. Hsu. "Deep Blue." In: *Artificial Intelligence* 134.1-2 (2002), pp. 57–83. DOI: 10.1016/S0004-3702(01)00129-1 (cit. on p. 3).

[36] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. "Toward an Architecture for Never-Ending Language Learning." In: *In Proceedings of the Conference on Artificial Intelligence (AAAI) (2010)*. 2010, pp. 1306–1313. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/download/1879/2201 (cit. on p. 18).

[37] C. Cattuto, D. Benz, A. Hotho, and G. Stumme. "Semantic Grounding of Tag Relatedness in Social Bookmarking Systems." In: *Proc. of the ISWC 2008*. Karlsruhe, Germany, 2008, pp. 615–631. DOI: 10.1007/978-3-540-88564-1_39 (cit. on p. 31).

[38] R. L. Cilibrasi and P. M. B. Vitányi. "The Google Similarity Distance." In: *IEEE Trans. Knowledge and Data Engineering* 19.3 (2007), pp. 370–383. arXiv: 0412098v3 [arXiv:cs] (cit. on pp. 27, 31, 62, 155).

[39] M. Cochez, P. Ristoski, S. P. Ponzetto, H. Paulheim, F. Fit, and S. Augustin. "Global RDF Vector Space Embeddings." In: *Proc. of the ISWC 2017*. Vienna, Austria (to appear), 2017 (cit. on pp. 32, 135).

[40] A. M. Collins and E. F. Loftus. "A Spreading-Activation Theory of Semantic Processing." In: *Psychological Review* 82.6 (1975), pp. 407–428. DOI: 10.1037/0033-295X.82.6.407 (cit. on pp. 4, 23).

[41] S. A. Cook. "The complexity of theorem-proving procedures." In: *ACM Symposium on Theory of Computing*. Shaker Heights, Ohio, USA: ACM New York, NY, USA, 1971, pp. 151–158. DOI: 10.1145/800157.805047 (cit. on p. 94).

[42] L. De Vocht, S. Coppens, R. Verborgh, M. Vander Sande, E. Mannens, and R. Van de Walle. "Discovering meaningful connections between resources in the web of data." In: *LDOW 2013*. Rio de Janeiro, Brazil: CEUR-WS Proceedings, 2013 (cit. on p. 165).

[43] M. Dean and G. Schreiber. *OWL Web Ontology Language Reference*. W3C Recommendation. W3C, Feb. 2004. URL: http://www.w3.org/TR/2004/REC-owl-ref-20040210/ (cit. on p. 15).

[44] R. Delbru, N. Toupikov, M. Catasta, G. Tummarello, and S. Decker. "Hierarchical Link Analysis for Ranking Web Data." In: *Proc. of the ESWC 2010*. Heraklion, Crete, Greece: Springer Berlin / Heidelberg, 2010, pp. 225–239. DOI: 10.1007/978-3-642-13489-0_16 (cit. on p. 29).

[45] B. Ding, Q. Wang, and B. Wang. "Leveraging Text and Knowledge Bases for Triple Scoring: An Ensemble Approach - The BOKCHOY Triple Scorer at WSDM Cup 2017." In: *WSDM Cup 2017 Notebook Papers, February 10, Cambridge, UK*. 2017. URL: http://www.wsdm-cup-2017.org/proceedings.html (cit. on p. 31).

[46] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. "Finding and Ranking Knowledge on the Semantic Web." In: *Proc. of the ISWC 2005*. Galway, Ireland: Springer Berlin / Heidelberg, 2005, pp. 156–170. DOI: http://dx.doi.org/10.1007/11574620_14 (cit. on p. 29).

[47] M. Duerst and M. Suignard. *Internationalized Resource Identifiers (IRIs)*. RFC 3987. RFC Editor, Jan. 2005. URL: http://www.rfc-editor.org/rfc/rfc3987.txt (cit. on pp. xvi, 8, 14).

[48] M. Ehrmann, F. Cecconi, D. Vannella, J. McCrae, P. Cimiano, and R. Navigli. "Representing Multilingual Data as Linked Data: the Case of BabelNet 2.0." In: *Proc. of LREC* (2014), pp. 401–408. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/810_Paper.pdf (cit. on pp. 18, 26, 27).

[49] B. Ell, D. Vrandečić, and E. Simperl. "Labels in the Web of Data." In: *Proc. of the ISWC 2011*. Bonn, Germany: Springer Berlin / Heidelberg, 2011, pp. 162–176. URL: http://www.springerlink.com/index/461072533UN13153.pdf (cit. on p. 30).

[50] EU Council. "EU Regulation 2016/679 General Data Protection Regulation (GDPR)." In: *Official Journal of the European Union* 59 (2016), pp. 1–88. URL: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679 (cit. on p. 5).

[51] S. Evert. "The Statistics of Word Cooccurrences - Word Pairs and Collocations." Dissertation. Universität Stuttgart, 2005, p. 353. URL: http://elib.uni-stuttgart.de/opus/volltexte/2005/2371/ (cit. on pp. 22, 31).

[52] *WordNet: An Electronic Lexical Database*. Cambridge, MA, USA: MIT Press, 1998. URL: http://wordnet.princeton.edu (cit. on pp. 18, 26, 27, 30, 71).

[53] J. D. Fernández, W. Beek, M. A. Martínez-Prieto, and M. Arias. "LOD-a-lot A Queryable Dump of the LOD Cloud." In: *Proc. of the ISWC 2017*. 2017. DOI: 10.1007/978-3-319-68204-4_7 (cit. on pp. 27, 166).

[54] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. "Binary RDF Representation for Publication and Exchange (HDT)." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 19 (2013), pp. 22–41. URL: http://www.websemanticsjournal.org/index.php/ps/article/view/328 (cit. on pp. 17, 166).

[55] D. Ferrucci et al. "Building Watson: An Overview of the DeepQA Project." In: *AI Magazine* 31.3 (2010), pp. 59–79. DOI: 10.1609/aimag.v31i3.2303 (cit. on p. 3).

[56] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP 1.1*. RFC 2068. RFC Editor, Jan. 1997. URL: http://www.rfc-editor.org/rfc/rfc2068.txt (cit. on pp. xvi, 18).

[57] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP 1.1): Message Syntax and Routing*. RFC 7230. RFC Editor, June 2014. URL: http://www.rfc-editor.org/rfc/rfc7230.txt (cit. on pp. xvi, 18).

[58] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP 1.1): Semantics and Content*. RFC 7231. RFC Editor, June 2014. URL: http://www.rfc-editor.org/rfc/rfc7231.txt (cit. on pp. xvi, 18).

[59] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP 1.1*. RFC 2616. RFC Editor, June 1999. URL: http://www.rfc-editor.org/rfc/rfc2616.txt (cit. on pp. xvi, 18).

[60] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. "Placing Search in Context: The Concept Revisited." In: *ACM Transactions on Information Systems (TOIS)* 20.1 (2002), pp. 116–131. DOI: 10.1145/503104.503110 (cit. on p. 27).

[61] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. "DEAP: Evolutionary Algorithms Made Easy." In: *Journal of Machine Learning Research* 13 (2012), pp. 2171–2175. DOI: 10.1.1.413.6512 (cit. on p. 98).

[62] T. Franz, A. Schultz, S. Sizov, and S. Staab. "TripleRank: Ranking Semantic Web Data by Tensor Decomposition." In: *Proc. of the ISWC 2009*. Chantilly, VA, USA: Springer Berlin / Heidelberg, 2009, pp. 213–228. DOI: 10.1007/978-3-642-04930-9_14 (cit. on p. 30).

[63] D. Galea and P. Bruza. "Deriving Word Association Networks from Text Corpora." In: *Proceedings of the EuroAsianPacific Joint Conference on Cognitive Science (EAPCogSci 2015)*. Torino, Italy: CEUR-WS Proceedings, 2015, pp. 252–257. URL: http://ceur-ws.org/Vol-1419/paper0038.pdf (cit. on pp. 23, 25, 136).

[64] F. Galton. "Psychometric Experiments." In: *Brain* 2.2 (1879), pp. 149–162. DOI: 10.1093/brain/2.2.149 (cit. on p. 20).

[65] P. Gearon, A. Passant, and A. Polleres. *SPARQL 1.1 Update*. W3C Recommendation. W3C, Mar. 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-update-20130321/ (cit. on p. 17).

[66] R. J. Gerrig and P. G. Zimbardo. *Psychology and Life*. 19th. Boston, USA: Allyn & Bacon, Pearson, 2010, p. 543 (cit. on pp. 19, 21, 22).

[67] Y. Goldberg and O. Levy. "word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method." In: *CoRR* (2014), pp. 1–5. arXiv: 1402.3722 (cit. on p. 32).

[68] B. Goodman and S. Flaxman. "EU regulations on algorithmic decision-making and a "right to explanation"." In: *ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*. 2016, pp. 1–9. arXiv: 1606.08813 (cit. on p. 5).

[69] A. Grover and J. Leskovec. "node2vec: Scalable Feature Learning for Networks." In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864. DOI: 10.1145/2939672.2939754. arXiv: 1607.00653 (cit. on p. 32).

[70] R. V. Guha and D. B. Lenat. "CYC: A Midterm Report." In: *AI Magazine* 11.3 (1990), pp. 32–59. DOI: 10.1609/aimag.v11i3.842 (cit. on p. 18).

[71] R. Guha and D. Brickley. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. W3C, Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/ (cit. on p. 15).

[72] R. Guha and D. Brickley. *RDF Schema 1.1*. W3C Recommendation. W3C, Feb. 2014. URL: http://www.w3.org/TR/2014/REC-rdf-schema-20140225/ (cit. on p. 15).

[73] S. Hacker and L. von Ahn. "Matchin: Eliciting User Preferences with an Online Game." In: *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*. Boston, MA, USA: ACM, 2009, pp. 1207–1216. URL: http://portal.acm.org/citation.cfm?id=1518701.1518882 (cit. on pp. 42, 45, 47).

[74] W. Hamilton. *The Works of Thomas Reid*. 8th ed. Vol. 2. Edinburgh, Scotland: Maclachlan and Stewart, 1880, p. 1034. URL: https://archive.org/details/worksnowfullyco02reiduoft (cit. on p. 19).

[75] S. Harris and A. Seaborne. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C, Mar. 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (cit. on pp. xv, xvi, 7, 9, 14, 16, 85, 86).

[76] Z. S. Harris. "Distributional structure." In: *Word* 10(23) (1954), pp. 146–162 (cit. on p. 31).

[77] Z. S. Harris. *Mathematical Structures of Language*. Wiley, 1968 (cit. on p. 31).

[78] A. Harth, S. Kinsella, and S. Decker. "Using Naming Authority to Rank Data and Ontologies for Web Search." In: *Proc. of the ISWC 2009*. Vol. 2. Chantilly, VA, USA: Springer Berlin / Heidelberg, 2009, pp. 277–292. DOI: http://dx.doi.org/10.1007/978-3-642-04930-9_18 (cit. on p. 29).

[79] J. Hees, B. Adrian, R. Biedert, T. Roth-Berghofer, and A. Dengel. "TSSort: Probabilistic Noise Resistant Sorting." In: *CoRR* abs/1606.0 (2016), pp. 1–10. arXiv: 1606.05289 (cit. on pp. 39, 47).

[80] J. Hees, R. Bauer, J. Folz, D. Borth, and A. Dengel. "An Evolutionary Algorithm to Learn SPARQL Queries for Source-Target-Pairs." In: *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016*. Vol. 10024. Bologna, Italy: Springer LNCS, Nov. 2016, pp. 337–352. DOI: 10.1007/978-3-319-49004-5_22. arXiv: 1607.07249 (cit. on pp. 83, 97, 127).

[81] J. Hees, R. Bauer, J. Folz, D. Borth, and A. Dengel. "Edinburgh Associative Thesaurus as RDF and DBpedia Mapping." In: *The Semantic Web - ESWC 2016 Satellite Events*. Vol. 9989 LNCS. Heraklion, Crete, Greece: Springer LNCS, May 2016, pp. 17–20. DOI: 10.1007/978-3-319-47602-5_4 (cit. on p. 65).

[82] J. Hees, R. Bauer, J. Folz, D. Borth, and A. Dengel. "Predicting Human Associations with Graph Patterns Learned from Linked Data." In: *ISWC 2017 Posters & Demonstrations and Industry Tracks*. Vol. 1963. Vienna, Austria: CEUR-WS Proceedings, 2017. URL: http://ceur-ws.org/Vol-1963/paper594.pdf (cit. on pp. 83, 122).

[83] J. Hees, M. Khamis, R. Biedert, S. Abdennadher, and A. Dengel. "Collecting Links between Entities Ranked by Human Association Strengths." In: *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC*. Vol. 7882. Montpellier, France: Springer LNCS, 2013, pp. 517–531. DOI: 10.1007/978-3-642-38288-8_35 (cit. on pp. 41, 54, 57).

[84] J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, and A. Dengel. "BetterRelations: Detailed Evaluation of a Game to Rate Linked Data Triples." In: *Proc. of the ISWC 2011 Ordering and Reasoning Workshop (OrdRing)*. Bonn, 2011. URL: http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/Workshops/OrdRing/paper_4_new.pdf (cit. on pp. 41, 44).

[85] J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, and A. Dengel. "BetterRelations: Using a Game to Rate Linked Data Triples." In: *KI 2011: Advances in Artificial Intelligence*. Berlin: Springer Berlin / Heidelberg, 2011, pp. 134–138. DOI: 10.1007/978-3-642-24455-1_12 (cit. on pp. 41, 44).

[86] J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, and A. Dengel. "BetterRelations: Collecting Association Strengths for Linked Data Triples with a Game." In: *Search Computing - Broadening Web Search*. Vol. 7538. Springer LNCS Berlin / Heidelberg, 2012, pp. 223–239. DOI: 10.1007/978-3-642-34213-4_15 (cit. on pp. 29, 41, 44).

[87] J. Hees, T. Roth-Berghofer, and A. Dengel. "Linked Data Games: Simulating Human Association with Linked Data." In: *LWA 2010*. Kassel, Germany, 2010, pp. 255–260. URL: http://www.kde.cs.uni-kassel.de/conf/lwa10/papers/wm2.pdf (cit. on pp. 24, 43).

[88] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann, and T. Stegemann. "RelFinder: Revealing Relationships in RDF Knowledge Bases." In: *Semantic Multimedia, 4th International Conference on Semantic and Digital Media Technologies, SAMT 2009*. Vol. 5887. LNCS. Graz, Austria: Springer, 2009, pp. 182–187. DOI: 10.1007/978-3-642-10543-2_21 (cit. on p. 34).

[89] P. Heim, S. Lohmann, and T. Stegemann. "Interactive Relationship Discovery via the Semantic Web." In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010*. Vol. 6088. LNCS. Heraklion, Crete, Greece: Springer, 2010, pp. 303–317. DOI: 10.1007/978-3-642-13486-9_21 (cit. on p. 34).

[90] S. Heindorf, M. Potthast, H. Bast, B. Buchhold, and E. Haussmann. "WSDM Cup 2017: Vandalism Detection and Triple Scoring." In: *Tenth ACM International Conference on Web Search and Data Mining (WSDM)*. Cambridge, UK: ACM, 2017, pp. 827–828. DOI: 10.1145/3018661.3022762 (cit. on p. 31).

[91] R. Herbrich, T. Minka, and T. Graepel. "TrueSkill(TM): A Bayesian Skill Rating System." In: *Advances in Neural Information Processing Systems*. Vol. 19. Cambridge, MA, USA: MIT Press, 2007, pp. 569–576. URL: http://books.nips.cc/papers/files/nips19/NIPS2006_0688.pdf (cit. on p. 47).

[92] S. Hertling, M. Schröder, C. Jilek, and A. Dengel. "Top-k Shortest Paths in Directed Labeled Multigraphs." In: *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016*. 2016, pp. 200–212. DOI: 10.1007/978-3-319-46565-4_16 (cit. on p. 165).

[93] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. "KORE: Keyphrase Overlap Relatedness for Entity Disambiguation." In: *Proceedings of the 21st ACM International Conference on Inofrmation and Knowledge Management (CIKM)*. Maui: ACM, 2012, pp. 545–554. DOI: 10.1145/2396761.2396832 (cit. on pp. 27, 152).

[94] T. Hofmann. "Probabilistic Latent Semantic Analysis." In: *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence*. Stockholm, Sweden, 1999, pp. 289–296 (cit. on p. 32).

[95] A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. "Weaving the Pedantic Web." In: *Proc. of the WWW2010 Workshop on Linked Data on the Web (LDOW)*. Raleigh, USA: CEUR Workshop Proceedings, 2010 (cit. on p. 18).

[96] I. Iacobacci, M. T. Pilehvar, and R. Navigli. "SensEmbed: Learning Sense Embeddings for Word and Relational Similarity." In: *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 2015, pp. 95–105. DOI: 10.3115/v1/P15-1010 (cit. on pp. 32, 33).

[97] P. Jain, P. Hitzler, P. Z. Yeh, K. Verma, and A. P. Sheth. "Linked Data Is Merely More Data." In: *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*. Palo Alto, California: AAAI,

2010, pp. 82–86. URL: http://www.aaai.org/ocs/index.php/SSS/SSS10/paper/view/1130 (cit. on pp. 18, 28).

[98] W. James. "The Principles of Psychology, Vol I." In: *American Science Series* 1 (1890), p. 689. URL: https://archive.org/stream/theprinciplesofp01jameuoft (cit. on p. 20).

[99] T. Joachims. "Optimizing Search Engines using Clickthrough Data." In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*. ACM, 2002. DOI: 10.1145/775066.775067 (cit. on p. 121).

[100] C. G. Jung. "The Association Method." In: *The American Journal of Psychology* 21.2 (1910), pp. 219–269. DOI: 10.2307/1413002 (cit. on p. 20).

[101] G. Kiss, C. Armstrong, R. Milroy, and J. Piper. "An associative thesaurus of English and its computer analysis." In: *The Computer and Literary Studies*. Edinburgh, UK: Edinburgh University Press, 1973, pp. 153–165. URL: http://www.eat.rl.ac.uk/Kiss_EAT_Chapter_1973.pdf (cit. on pp. xv, 4, 19, 20, 26, 27, 39, 65, 74, 162).

[102] J. M. Kleinberg. "Authoritative sources in a hyperlinked environment." In: *Journal of the ACM* 46.5 (1999), pp. 604–632. URL: http://dl.acm.org/citation.cfm?id=324140 (cit. on p. 29).

[103] T. Kliegr. "Linked hypernyms: Enriching DBpedia with Targeted Hypernym Discovery." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 31 (2015), pp. 59–69. DOI: 10.1016/j.websem.2014.11.001 (cit. on p. 79).

[104] J. C. Klyne and Graham. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. W3C, Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (cit. on pp. xv, xvi, 3, 6, 8, 14).

[105] E. Kny, S. Kölle, G. Töpper, and E. Wittmers. *WhoKnows?* Tech. rep. Potsdam: Hasso-Plattner-Institut, Oct. 2010, p. 43. URL: http://www.emilia-wittmers.de/paper/WhoKnows.pdf (cit. on pp. 25, 43).

[106] K. J. Kochut and M. Janik. "SPARQLeR: Extended Sparql for Semantic Association Discovery." In: *Proceedings of the 4th European conference on The Semantic Web: Research and Applications - ESWC '07*. 2007, pp. 145–159. DOI: 10.1007/978-3-540-72667-8_12 (cit. on p. 165).

[107] B. J. Kuipers. "On Representing Commonsense Knowledge." In: *Associative Networks: The Representation and Use of Knowledge by Computers*. New York, NY, USA: Academic Press, 1979, pp. 393–408. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.9010%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf (cit. on p. 18).

[108] T. K. Landauer and S. T. Dumais. "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge." In: *Psychological Review* 104.2 (1997), pp. 211–240 (cit. on p. 32).

[109] A. Ławrynowicz and J. Potoniec. "Pattern based feature construction in semantic data mining." In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.1 (2014), pp. 27–65. DOI: 10.4018/978-1-4666-8751-6.ch036 (cit. on p. 35).

[110] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539. arXiv: arXiv:1312.6184v5 (cit. on p. 3).

[111] J. Lehmann and L. Bühmann. "AutoSPARQL: Let users query your knowledge base." In: *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece*. Vol. 6643. LNCS. Heraklion, Crete, Greece: Springer, 2011, pp. 63–79. DOI: 10.1007/978-3-642-21034-1_5 (cit. on p. 34).

[112] J. Lehmann, J. Schüppel, and S. Auer. "Discovering Unknown Connections – the DBpedia Relationship Finder." In: *CSSW 2007*. Leipzig, Germany: CEUR-WS Proceedings, 2007, pp. 99–110. URL: http://ceur-ws.org/Vol-301/Paper_9_Lehmann.pdf (cit. on p. 33).

[113] O. Levy and Y. Goldberg. "Neural Word Embedding as Implicit Matrix Factorization." In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2177–2185. URL: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization (cit. on p. 32).

[114] H. Lieberman, D. A. Smith, and A. Teeters. "Common Consensus: a web-based game for collecting commonsense goals." In: *Workshop on Common Sense for Intelligent Interfaces, ACM International Converence on Intelligent User Interfaces (IUI-07)*. Hawaii, USA, Jan. 2007. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.2969%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf (cit. on p. 43).

[115] H. Liu and P. Singh. "ConceptNet - A Practical Commonsense Reasoning Tool-Kit." In: *BT Technology Journal* 22.4 (Oct. 2004), pp. 211–226. DOI: 10.1023/B:BTTJ.0000047600.45421.6d (cit. on p. 18).

[116] A. Mallea, M. Arenas, A. Hogan, and A. Polleres. "On Blank Nodes." In: *The Semantic Web - ISWC 2011*. Vol. 7031. LNCS. Bonn: Springer Berlin / Heidelberg, 2011, pp. 421–437. DOI: 10.1007/978-3-642-25073-6_27 (cit. on p. 15).

[117] C. D. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Online edi. Cambridge, England: Cambridge University Press, 2009, p. 581. URL: https://nlp.stanford.edu/IR-book/ (cit. on p. 131).

[118] F. Manola and E. Miller. *RDF Primer*. W3C Recommendation. W3C, Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (cit. on p. 15).

[119] J. P. McCrae, C. Fellbaum, and P. Cimiano. "Publishing and Linking WordNet using lemon and RDF." In: *Proceedings of the 3rd Workshop on Linked Data in Linguistics*. 2014, pp. 1–4. URL: https://pub.uni-bielefeld.de/download/2732779/2732785 (cit. on pp. 26, 27).

[120] J. P. McCusker. "WebSig: A Digital Signature Framework for the Web." PhD thesis. Rensselaer Polytechnic Institute, Troy, NY, 2015, p. 126. URL: http://gradworks.umi.com/3727015.pdf (cit. on pp. 94, 107).

[121] B. D. McKay and A. Piperno. "Practical graph isomorphism, II." In: *Journal of Symbolic Computation* 60.January 2013 (2014), pp. 94–112. DOI: 10.1016/j.jsc.2013.09.003. arXiv: 1301.1493 (cit. on p. 94).

[122] C. McKinstry, R. Dale, and M. J. Spivey. "Action Dynamics Reveal Parallel Competition in Decision Making." In: *Psychological Science* 19.1 (2008), pp. 22–24. DOI: 10.1111/j.1467-9280.2008.02041.x (cit. on p. 18).

[123] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. "DBpedia Spotlight: Shedding Light on the Web of Documents." In: *Proc. of the I-SEMANTICS*. Graz, Austria: ACM, 2011. URL: http://www.wiwiss.fu-berlin.de/en/institute/pwo/bizer/research/publications/Mendes-Jakob-GarciaSilva-Bizer-DBpediaSpotlight-ISEM2011.pdf (cit. on p. 56).

[124] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and their Compositionality." In: *NIPS*. Oct. 2013, pp. 1–9. arXiv: 1310.4546 (cit. on pp. 32, 135).

[125] G. A. Miller. "WordNet: A Lexical Database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748 (cit. on pp. 18, 26, 27, 30).

[126] D. Milne and I. H. Witten. "An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links." In: *AAAI WS Wikipedia and Artificial Intelligence: An Evolving Synergy*. 2008, pp. 25–30. URL: https://aaai.org/Papers/Workshops/2008/WS-08-15/WS08-15-005.pdf (cit. on pp. 29, 133).

[127] R. Mirizzi, A. Ragone, T. D. Noia, and E. D. Sciascio. "Ranking the Linked Data: The Case of DBpedia." In: *Proc. of the ICWE 2010*. Vienna, Austria: Springer Berlin / Heidelberg, 2010, pp. 337–354. DOI: 10.1007/978-3-642-13911-6_23 (cit. on p. 31).

[128] T. M. Mitchell et al. "Never-Ending Learning." In: *AAAI 29th Conference on Artificial Intelligence*. 2015, pp. 2302–2310. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/download/10049/9557 (cit. on p. 18).

[129] E. T. Mueller. *Natural Language Processing with ThoughtTreasure*. New York: Signiform, 1998 (cit. on p. 18).

[130] R. Navigli and S. P. Ponzetto. "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network." In: *Artificial Intelligence* 193 (2012), pp. 217–250. DOI: 10.1016/j.artint.2012.07.001 (cit. on pp. 18, 26, 27).

[131] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber. *The University of South Florida word association, rhyme, and word fragment norms*. 1998. URL: http://www.usf.edu/FreeAssociation/ (cit. on pp. xvi, 21, 26, 27, 74, 165).

[132] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber. "The University of South Florida free association, rhyme, and word fragment norms." In: *Behavior Research Methods, Instruments, & Computers* 36.3 (2004), pp. 402–407. DOI: 10.3758/BF03195588 (cit. on pp. xvi, 21, 26, 27, 74, 165).

[133] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. "A Review of Relational Machine Learning for Knowledge Graphs." In: (2015), pp. 1–23. arXiv: 1503.00759 (cit. on pp. 28, 33).

[134] M. Nickel, V. Tresp, and H.-P. Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data." In: *Proc. of the 28th International Converence on Machine Learning (ICML)*. Bellevue, WA, US, 2011 (cit. on p. 30).

[135] M. Nickel, V. Tresp, and H.-P. Kriegel. "Factorizing YAGO." In: *Proceedings of the 21st International Conference on World Wide Web*. Lyon, France, 2012, pp. 271–280. DOI: 10.1145/2187836.2187874 (cit. on p. 30).

[136] C. K. Ogden and I. A. Richards. *The Meaning of Meaning: A study of the influence of thought and of the science of symbolism*. New York: Harcourt, Brace & World, 1923 (cit. on p. 21).

[137] H. Paulheim. "DBpediaNYD - A silver standard benchmark dataset for semantic relatedness in DBpedia." In: *CEUR Workshop Proceedings* 1064 (2013), pp. 1–5. URL: http://ceur-ws.org/Vol-1064/Paulheim_DBpediaNYD.pdf (cit. on pp. 27, 155).

[138] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2012), pp. 2825–2830. DOI: 10.1007/s13398-014-0173-7.2. arXiv: 1201.0490 (cit. on p. 120).

[139] J. Pennington, R. Socher, and C. D. Manning. "GloVe: Global Vectors for Word Representation." In: *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162 (cit. on p. 32).

[140] J. Potoniec. "An On-Line Learning to Query System." In: *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016)*. Vol. 1690. Kobe, Japan: CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1690/paper54.pdf (cit. on p. 35).

[141] E. Prud'hommeaux and C. Buil-Aranda. *SPARQL 1.1 Federated Query*. W3C Recommendation. W3C, Mar. 2013. URL: https://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/ (cit. on p. 17).

[142] E. Prud'hommeaux and G. Carothers. *RDF 1.1 Turtle*. W3C Recommendation. W3C, Feb. 2014. URL: http://www.w3.org/TR/2014/REC-turtle-20140225/ (cit. on p. 14).

[143] R. Qian. *Understand Your World with Bing*. 2013. URL: http://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing (cit. on pp. 4, 18).

[144] M. R. Quillian. "Semantic Memory." In: *Semantic Information Processing*. Cambridge, MA, USA: MIT Press, 1968, pp. 216–260 (cit. on p. 23).

[145] Y. Raimond and G. Schreiber. *RDF 1.1 Primer*. W3C Note. W3C, June 2014. URL: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/ (cit. on p. 15).

[146] R. Rapp. "The Computation of Word Associations: Comparing Syntagmatic and Paradigmatic Approaches." In: *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*. COLING. Taipei, Taiwan: Association for Computational Linguistics, 2002. DOI: 10.3115/1072228.1072235 (cit. on pp. 23, 25, 136).

[147] R. Řehůřek and P. Sojka. "Software Framework for Topic Modelling with Large Corpora." In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, 2010, pp. 45–50. URL: https://is.muni.cz/publication/884893/en (cit. on p. 135).

[148] A. Rettinger, U. Lösch, V. Tresp, C. D'Amato, and N. Fanizzi. "Mining the Semantic Web: Statistical learning for next generation knowledge bases." In: *Data Mining and Knowledge Discovery* 24.3 (2012), pp. 613–662. DOI: 10.1007/s10618-012-0253-2 (cit. on pp. 18, 28, 33).

[149] P. Ristoski, C. Bizer, and H. Paulheim. "Mining the Web of Linked Data with RapidMiner." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 35.3 (2015), pp. 142–151. DOI: 10.1016/j.websem.2015.06.004 (cit. on p. 35).

[150] P. Ristoski and H. Paulheim. "RDF2Vec: RDF graph embeddings for data mining." In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference*. Vol. 9981. Kobe, Japan: Springer LNCS, 2016, pp. 498–514. DOI: 10.1007/978-3-319-46523-4_30 (cit. on pp. 32, 135).

[151] P. Ristoski and H. Paulheim. "Semantic Web in data mining and knowledge discovery: A comprehensive survey." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 36 (2016), pp. 1–22. DOI: 10.1016/j.websem.2016.01.001 (cit. on pp. 28, 33).

[152] P. Ristoski, J. Rosati, T. D. Noia, R. D. Leone, and H. Paulheim. "RDF2Vec: RDF Graph Embeddings and Their Applications." In: *Semantic Web* (under review) (2017). URL: http://www.semantic-web-journal.net/system/files/swj1643.pdf (cit. on pp. 32, 135).

[153] S. Rudolph, B. Parsia, P. Patel-Schneider, M. Krötzsch, and P. Hitzler. *OWL 2 Web Ontology Language Primer (Second Edition)*. W3C Recommendation. W3C, Dec. 2012. URL: http://www.w3.org/TR/2012/REC-owl2-primer-20121211/ (cit. on p. 15).

[154] G. Schreiber and F. Gandon. *RDF 1.1 XML Syntax*. W3C Recommendation. W3C, Feb. 2014. URL: http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/ (cit. on p. xvi).

[155] A. Seaborne and G. Carothers. *RDF 1.1 N-Triples*. W3C Recommendation. W3C, Feb. 2014. URL: http://www.w3.org/TR/2014/REC-n-triples-20140225/ (cit. on p. 14).

[156] A. Seaborne and E. Prud'hommeaux. *SPARQL Query Language for RDF*. W3C Recommendation. W3C, Jan. 2008. URL: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/ (cit. on pp. xv, xvi, 7, 9, 14, 16, 85, 86).

[157] N. Shadbolt, W. Hall, and T. Berners-Lee. "The Semantic Web Revisited." In: *IEEE Intelligent Systems* 21.3 (2006), pp. 96–101. DOI: 10.1109/MIS.2006.62. arXiv: 1204.6441 (cit. on p. 17).

[158] D. Silver et al. "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961 (cit. on p. 3).

[159] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. "Open Mind Common Sense: Knowledge Acquisition from the General Public." In: *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE: Confederated International Conferences CoopIS, DOA, and ODBASE 2002 Proceedings*. Davis 1990. Springer Berlin / Heidelberg, 2002, pp. 1223–1237. DOI: 10.1007/3-540-36124-3_77 (cit. on p. 18).

[160] A. Singhal. *Introducing the Knowledge Graph: things, not strings*. 2012. URL: http://googleblog.blogspot.pt/2012/05/introducing-knowledge-graph-things-not.html (cit. on pp. 4, 18).

[161] K. Siorpaes and M. Hepp. "OntoGame: Towards Overcoming the Incentive Bottleneck in Ontology Building." In: *Proc. of the 3rd International IFIP Workshop On Semantic Web & Web Semantics (SWWS), OTM-WS 2007, Part II. LNCS*. Vilamoura, Portugal: Springer, Heidelberg, 2007, pp. 1222–1232. URL: http://members.deri.at/~katharinas/files/publications/SiorpaesHepp-OntoGame-camready-completelyfinal.pdf (cit. on p. 43).

[162] K. Siorpaes and M. Hepp. "Games with a Purpose for the Semantic Web." In: *IEEE Intelligent Systems* 23.3 (2008), pp. 50–60. URL: http://www.heppnetz.de/files/gwap-semweb-ieee-is.pdf (cit. on p. 43).

[163] N. Soares and B. Fallenstein. "Agent Foundations for Aligning Machine Intelligence with Human Interests: A Technical Research Agenda." In: *The Technological Singularity: Managing the Journey*. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2017, pp. 103–125. DOI: 10.1007/978-3-662-54033-6_5 (cit. on p. 3).

[164] R. Speer and C. Havasi. "Representing General Relational Knowledge in ConceptNet 5." In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. 2012, pp. 3679–3686. URL: http://conceptnet.io/ (cit. on p. 18).

[165] M. Strube and S. P. Ponzetto. "WikiRelate! Computing Semantic Relatedness Using Wikipedia." In: *Proc. of the AAAI 2006*. February. Boston, MA, USA: AAAI Press, 2006, pp. 1419–1424. URL: http://www.aaai.org/Papers/AAAI/2006/AAAI06-223.pdf (cit. on p. 30).

[166] F. M. Suchanek, G. Kasneci, and G. Weikum. "YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia." In: *WWW*. Banff, Alberta, Canada: ACM, 2007, pp. 697–706. URL: http://www.www2007.org/papers/paper391.pdf (cit. on p. 18).

[167] A. Thalhammer. "Linked Data Entity Summarization." Dissertation. Karlsruhe Institute of Technology (KIT), 2016. URL: https://d-nb.info/1124903089/34 (cit. on p. 29).

[168] A. Thalhammer, M. Knuth, and H. Sack. "Evaluating Entity Summarization Using a Game-Based Ground Truth." In: *ISWC 2012*. Vol. 7650. Boston, MA, USA: Springer LNCS, 2012, pp. 350–361. DOI: 10.1007/978-3-642-35173-0_24 (cit. on p. 43).

[169] A. Thalhammer, N. Lasierra, and A. Rettinger. "LinkSUM: Using Link Analysis to Summarize Entity Data." In: *Web Engineering: ICWE 2016*. Vol. 9671. Lugano, Switzerland: Springer LNCS, 2016, pp. 244–261. DOI: 10.1007/978-3-319-38791-8_14 (cit. on p. 29).

[170] A. Thalhammer and A. Rettinger. "Browsing DBpedia Entities with Summaries." In: *ESWC 2014 SE*. Vol. 8798. Crete, Greece: Springer LNCS, 2014, pp. 511–515. DOI: 10.1007/978-3-319-11955-7_76 (cit. on p. 29).

[171] A. Thalhammer and A. Rettinger. "PageRank on Wikipedia: Towards General Importance Scores for Entities." In: *ESWC 2016*. Vol. 9989. Heraklion, Crete, Greece: Springer LNCS, 2016, pp. 227–240. DOI: 10.1007/978-3-319-47602-5_41 (cit. on pp. 30, 133).

[172] G. Tummarello, R. Delbru, and E. Oren. "Sindice.com: Weaving the Open Linked Data." In: *Proc. of the ISWC 2007*. Springer Berlin / Heidelberg, 2007, pp. 552–565. DOI: 10.1007/978-3-540-76298-0_40 (cit. on p. 29).

[173] R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert. "Triple Pattern Fragments: A Low-cost Knowledge Graph Interface for the Web." In: *Journal of Web Semantics* 37-38.2016 (2016), pp. 184–206. DOI: 10.1016/j.websem.2016.03.003 (cit. on pp. 17, 166).

[174] D. Vrandečić and M. Krötzsch. "Wikidata: A Free Collaborative Knowledgebase." In: *Communications of the ACM* 57.10 (2014), pp. 78–85. DOI: 10.1145/2629489 (cit. on pp. 18, 74).

[175] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation. W3C, Dec. 2012. URL: http://www.w3.org/TR/2012/REC-owl2-overview-20121211/ (cit. on p. 15).

[176] W3C Semantic Web Interest Group. *The Semantic Web Layer Cake*. 2008. URL: https://www.w3.org/2001/sw/layerCake.svg (cit. on p. 14).

[177] W3C SPARQL Working Group. *SPARQL 1.1 Overview*. W3C Recommendation. W3C, Mar. 2013. URL: http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/ (cit. on p. 16).

[178] J. Waitelonis and H. Sack. "Towards Exploratory Video Search Using Linked Data." In: *Proc. of the IEEE International Symposium on Multimedia (ISM) 2009*. San Diego, CA, USA: IEEE, 2009, pp. 540–545. DOI: 10.1109/ISM.2009.111 (cit. on p. 31).

[179] H. C. Warren. *A History of the Association Psychology*. Princeton, New Jersey: Scribner Press, 1921, p. 328. URL: https://archive.org/details/historyoftheasso007979mbp (cit. on p. 19).

[180] J. Washtell and K. Markert. "A Comparison of Windowless and Window-Based Computational Association Measures as Predictors of Syntagmatic Human Associations." In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09)*. August. Singapore, 2009, pp. 628–637. URL: http://www.aclweb.org/anthology/D09-1066 (cit. on pp. 23, 25, 136).

[181] L. Wolf, M. Knuth, J. Osterhoff, and H. Sack. "RISQ! Renowned Individuals Semantic Quiz - A Jeopardy like Quiz Game for Ranking Facts." In: *Proc. of the I-SEMANTICS*. Graz, Austria: ACM, 2011, pp. 71–78. DOI: 10.1145/2063518.2063528 (cit. on p. 43).

[182] D. Wood, M. Lanthaler, and R. Cyganiak. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C, Feb. 2014. URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ (cit. on pp. xv, xvi, 3, 6, 8, 14).

(text galeabruza), 136
(text rapp), 136
(text washtell), 136
?source, 9
?target, 9

adaboost, 121
AGI, 3, 4, 11, 161
AI, 3, 4, 13
all datasets, 129
ambiguity, 22
analogies, 32
any nb, 133
ARD regression, 121
ASK query, 17, 86, 105
association, 5, 19, 21, 65
association network, 20, 23
association strength, 6, 24
AutoSPARQL, 34
average lifetime play, 48, 59

BabelNet, 26
BetterRelations, 39, 45, 75, 161
BGP, 9, 16, 28, 33, 85, 86, 93–95, 97, 99, 107, 117, 162, 164
bidi nb, 133
BNode, 8, 15, 21, 86, 94
BNodes B, 86

canonical form of a pattern canon(gp), 94
canonical graph patterns, 151
clustering approaches, 115
complete pattern, 9
connected, 91
context, 6, 22
core dataset, 78, 129
COUNT query, 16
coverage matrix C, 116
coverage vector $\mathbf{c}_{t_i}$, 116
covers, 87

CURIE, 7

DBpedia Relationship Finder, 33
DBpediaNYD, 27
DBpediaRanking, 31
defaultPlaceHolder, 13, 16, 19, 21, 22, 42–44, 50, 54, 57, 75, 78, 84, 89, 105, 108, 113, 116, 127, 136, 138–140, 144, 145, 155–157
degrees, 29
DESCRIBE query, 41
disconnected, 91
dominant parent, 100

EAT, 20, 26, 27, 39, 65–68, 70–77, 136, 152, 162, 164, 165
edge, 7
edge-only connected, 91
execution time, 89
expected contribution, 48, 59
explainability, 33
extended dataset, 78, 129

f measures, 119
f measures precisions, 119
fact, 7
fitness, 97, 99
frequency, 66
fulfilled, 9
  gp fulfils (s_i, t_i), 87
fusion model, 120
fusion of target candidates, 117

gain, 88
game, 54
Genderizer, 44
generation, 97

GloRDFVe, 32
GloVe, 32
Gold Standard list, 50
gp precisions, 119
gp precisions precisions, 119
graph pattern gp, 8, 86
Graph Pattern Learner gpl($\mathcal{GT}$, G), 86, 97, 162
ground truth, 61
ground truth $\mathcal{GT}$, 10, 77, 86
ground truth precision coverage, 113
guess, 55
GWAP, 10, 42–45, 48, 58, 64, 75

Harris' Distributional Hypothesis, 31
HITS, 29
hits, 133
homomorphism, 93
HTTP, 18, 128
human communication, 24
human performance, 136, 142

in nb, 133
incomplete pattern, 9
indeg, 133
individuals, 97
Intra-dataset learning, 139
IQR, 144
IRI, 8, 107
isomorphism, 92, 94
isomorphism class, 94
items, 45

knowledge base, 7
knowledge graph G, 7, 85
Knowledge Test Game, 39, 54, 76, 161
KORE, 27
KRETR, 35

label, 7
Linked Data, 6, 18
Linked Data mirror, 7
Literal, 8, 14, 15, 86, 165
Literals L, 86
LOD, 6, 7, 35
LOD Cloud, 3, 7, 18, 26, 69, 128, 166
logistic regression, 121

MAP, 130, 131, 140, 144, 163
mate, 97
max precision coverage vector $\mathbf{mpv}_{GP_e}$, 113
max precision loss, 115
max precision vector sum mpvs(GP_e), 114
Milne-Witten Relatedness, 29, 133
model, 10
models, 87
MRR, 121, 130, 131, 136, 140, 144, 146, 155–157, 163
MSE, 51–53
mutate, 97
mw wl, 134
mw wl top5overlap, 134

NASARI, 33
nasari, 135
NDCG, 62, 63, 130–132, 136, 140, 144, 147, 153, 155–157, 163
NER, 22
node, 7
node connected, 92
node-edge joint, 92
Normalised Google Distance, 31
normalised max precision vector sum nmpvs(GP_e), 114

object, 7, 14
occurrences, 66
OntoGame, 43
other box, 57
out nb, 133
outdeg, 133
over-fitting, 89

PageRank, 29
pagerank, 133
path patterns, 103, 150

pattern length, 89
population, 97
POS, 26
precision, 88
precision vector $\mathbf{pv}_{gp}$, 113
precisions, 119
pred, 134
predicate, 7, 14
prediction, 86
primary word associations, 20, 65

random forest, 121
RankSVM, 121
raw associations, 65
RDF, 3, 5, 6, 13, 14, 18, 25, 28, 39, 65–68, 74, 80, 90, 91, 94, 95, 106, 164, 165
RDF term, 8
RDF2Vec, 32
rdf2vec, 135
recall, 88
Recall@k, 130
recessive parent, 100
referent, 21
related triples, 44
relation, 7
relation $\mathcal{R}$, 85
RelFinder, 34
response, 6, 65
resulting graph patterns GP_r, 115
RISQ, 43
round, 45
runs, 98

scores, 118
scores precisions, 119
scoring function, 46
SELECT query, 9, 16, 86
self loops, 91
semantic association, 8, 25, 39, 54, 55, 68, 74
semantic entity, 7
semantic network, 23
Semantic Web, 6
semi-automatic mapping approach, 39, 76
Semiotic Triangle, 21
sim, 134
simpred, 134
single player mode, 46
source node s, 85
source-target pairs, 10
SPARQL, 3, 7–9, 13, 16, 18, 28, 33–35, 41, 56, 84–87, 91, 95, 97, 99, 117, 122, 150, 162, 164–166
SPARQL endpoint, 7
statement, 14
stimulus, 6, 65
stimulus-response pair, 6
strong associations, 6, 65
subject, 7, 14
suggestions-box, 56
surface form, 8
symbol, 21
symbolic form, 8, 47
symbolic indirection, 24

target candidates $TC_{GP_r}(s)$, 116
target node t, 85
target occs, 118
tensor factorisation, 30
thought, 21
thought association, 21
throughput, 48, 59
time, 89
timeout, 89
topic, 45, 50
topic done, 58
topic of interest, 44, 54
training data, 10
triple, 7, 14
Typewriter, 43

unique associations, 65
URI, 7, 8, 15, 18, 21, 24, 67, 72, 73, 86, 94, 136, 149, 156
URIs U, 86
USFA, 21, 26, 27, 74, 165

valid mapping, 72

variable, 8, 9, 16
variable count, 89
Variables V, 86
variables in graph pattern
vars(gp), 93

W3C, 6
WhoKnows?, 43
WhoKnows? Movies, 43
wiki doc sim, 135
WikiRelate, 30
wl, 133
word association, 6, 21
word association norms, 20
Word2Vec, 32
word2vec, 135
WordNet, 26, 30
WordSim353, 27
WWW, 13, 24, 29

XML, 14

ACADEMIC CURRICULUM VITÆ: JÖRN HEES

CONTACT INFORMATION

Website: http://joernhees.de
Email: diss@joernhees.de

EDUCATION

TU Kaiserslautern *11/2010 - 04/2018*
PhD Student / Researcher, Knowledge-based Systems Group, Prof. Dengel
Main areas: Semantic Web, Linked Data, Data Mining, Machine Learning
PhD Topic: *Simulating Human Associations with Linked Data*

TU Kaiserslautern *10/2007 - 12/2015*
Scholar of the Computer Science Department's PhD Program

TU Kaiserslautern *10/2007 - 02/2011*
Master of Science in Computer Science
Master Thesis: *Better Relations: Using a Game to Rate Linked Data Triples*

German National Academic Foundation *05/2006 - 09/2010*
Scholar of the *Studienstiftung des deutschen Volkes*

Bonn-Rhine-Sieg University of Applied Sciences *09/2004 - 09/2007*
Bachelor of Science in Computer Science
Bachelor Thesis: *Generating Ontologies from Flickr Tags*

EXPERIENCE

German Research Center for Artificial Intelligence (DFKI) *since 01/2016*
Smart Data and Knowledge Services Department, Prof. Dengel
Full-time Researcher, Main Projects: MOM, DeFuseNN

OpenSource Software Library RDFLib *since 05/2013*
Library to work with RDF in Python
Maintainer & Developer

German Research Center for Artificial Intelligence (DFKI) *10/2008 - 12/2015*
Knowledge Management Department, Prof. Dengel
Tutor, Guest Researcher, Main Projects: myCBR, NEXUS & MOM

SELECTED PUBLICATIONS

What do Deep Networks Like to See? *CVPR 2018*
S. Palacio, J. Folz, **J. Hees**, F. Raue, D. Borth, A. Dengel

Simplified SPARQL REST API *ESWC 2018*
M. Schröder, **J. Hees**, A. Bernardi, D. Ewert, P. Klotz, S. Stadtmüller

Predicting Human Associations with Graph Patterns Learned from Linked Data *ISWC 2017 P&D*
J. Hees, R. Bauer, J. Folz, D. Borth, A. Dengel

RDF Spreadsheet Editor: Get (G)rid of Your RDF Data Entry Problems *ISWC 2017 P&D*
M. Schröder, C. Jilek, **J. Hees**, S. Hertling, A. Dengel

EKAW 2016
An Evolutionary Algorithm to Learn SPARQL Queries for Source-Target-Pairs
J. Hees, R. Bauer, J. Folz, D. Borth, A. Dengel

ESWC 2016 SE
Edinburgh Associative Thesaurus as RDF and DBpedia Mapping
J. Hees, R. Bauer, J. Folz, D. Borth, A. Dengel

CoRR 2016
TSSort: Probabilistic Noise Resistant Sorting
J. Hees, B. Adrian, R. Biedert, T. Roth-Berghofer, A. Dengel

ESWC 2013
Collecting Links between Entities Ranked by Human Association Strengths
J. Hees, M. Khamis, R. Biedert, S. Abdennadher, A. Dengel

ACM MM 2013
Analysis and Forecasting of Trending Topics in Online Media Streams
T. Althoff, D. Borth, **J. Hees**, A. Dengel

Book Chapter: Search Computing 2012
BetterRelations: Collecting Association Strengths for Linked Data Triples with a Game
J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, A. Dengel

ETRA 2012
A Robust Realtime Reading-Skimming Classifier
R. Biedert, **J. Hees**, A. Dengel, G. Buscher

ISWC 2011 OrdRing WS
BetterRelations: Detailed Evaluation of a Game to Rate Linked Data Triples
J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, A. Dengel

KI 2011
BetterRelations: Using a Game to Rate Linked Data Triples
J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, A. Dengel

EKAW 2010
Epiphany: Adaptable RDFa Generation Linking the Web of Documents to the Web of Data
B. Adrian, **J. Hees**, I. Herman, M. Sintek, A. Dengel

LWA 2010
Linked Data Games: Simulating Human Associations with Linked Data
J. Hees, T. Roth-Berghofer, A. Dengel

CHI EA 2010
Text 2.0
R. Biedert, G. Buscher, S. Schwarz, **J. Hees**, A. Dengel

KI 2009
iDocument: Using Ontologies for Extracting and Annotating Information from Unstructured Text
B. Adrian, **J. Hees**, L. van Elst, A. Dengel