



UNIVERSITÄT MANNHEIM

DOKTORARBEIT

Symbolische Regelbasierte Wissensgraphkomplettierung

Autor:
Patrick BETZ

Betreuer:
Prof. Dr. Heiner STUCKENSCHMIDT

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

October 29, 2025

Dekan:
Prof. Dr. Claus Hertling

Gutachter:
Prof. Dr. Heiner Stuckenschmidt
Prof. Dr. Fabian Suchanek
Prof. Dr. Christian Bartelt



UNIVERSITY OF MANNHEIM

DOCTORAL THESIS

Symbolic Rule-Based Knowledge Graph Completion

Author:
Patrick BETZ

Supervisor:
Prof. Dr. Heiner STUCKENSCHMIDT

*A thesis submitted in fulfilment of the
requirements for the degree of Doctor of Science
at the University of Mannheim*

October 29, 2025

Abstract

Knowledge graphs are composed of relational facts that encode correct statements about the world. In the problem of knowledge graph completion, the goal is to augment a knowledge graph with missing but correct statements. Symbolic rule-based approaches perform this task by using logical clauses or rules. They are fully interpretable and efficient to use when using suitable inference mechanisms. A symbolic approach for performing knowledge graph completion can be separated into two stages. In the first stage, rules are learned inductively from the knowledge graph using a mining system. In the second stage, they must be applied to derive new facts. The focus of this thesis is on the latter stage which can become challenging when knowledge graphs are large and when many rules are learned. We explore and investigate how to effectively employ rules to make new fact predictions. We demonstrate in this context how rules make fully explainable predictions and how they can be used to explain other model classes. We analyse rule inference from a theoretical view and propose methods to make new predictions using supervision from a training knowledge graph. To have a meaningful evaluation, the rule-based approach and the proposed methods are compared with popular latent-based models that learn embeddings from the knowledge graph. Our results show that rule-based approaches are highly competitive when considering the predictive performance. Finally, we explore other model classes which inspires simple augmentation strategies to increase the expressiveness of the rule-based approach.

Zusammenfassung

Wissensgraphen bestehen aus relationalen Fakten, die korrekte Aussagen über die Welt beschreiben. Bei der Wissensgraphkomplettierung besteht das Ziel darin, einen Wissensgraphen mit fehlenden, aber korrekten Aussagen zu ergänzen. Symbolische regelbasierte Ansätze betreiben Wissensgraphkomplettierung durch die Verwendung logischer Klauseln oder Regeln. Sie sind vollständig interpretierbar und effizient, wenn geeignete Inferenzmechanismen verwendet werden. Ein symbolischer Ansatz zur Wissensgraphkomplettierung kann in zwei Phasen unterteilt werden. In der ersten Phase werden Regeln induktiv aus dem Wissensgraphen mit Hilfe eines Lernalgorithmus gesucht. In der zweiten Phase müssen sie angewendet werden, um neue Fakten abzuleiten. Der Fokus dieser Arbeit liegt auf der letztgenannten Phase, die herausfordernd werden kann, wenn Wissensgraphen groß sind und wenn viele Regeln gelernt werden. Wir untersuchen, wie man Regeln effektiv einsetzen kann, um neue Vorhersagen zu treffen. Wir demonstrieren, wie symbolische Regeln vollständig erklärbare Vorhersagen treffen und wie sie verwendet werden können, um auch andere Modellklassen zu erklären. Regelninferenz wird aus einer theoretischen Sicht untersucht und es werden Methoden präsentiert, um die Inferenz anhand von überwachtem Lernen basierend auf dem Wissensgraphen zu verbessern. Für eine aussagekräftige Bewertung werden der regelbasierte Ansatz und die vorgeschlagenen Methoden mit gängigen Modellen verglichen, welche latente Repräsentationen aus dem Wissensgraphen lernen. Unsere Ergebnisse zeigen, dass regelbasierte Ansätze hinsichtlich der Vorhersagequalität stark konkurrenzfähig sind. Außerdem untersuchen wir andere Modellklassen für das Problem, was einfache Erweiterungsstrategien zur Steigerung der Ausdrucksfähigkeit des regelbasierten Ansatzes inspiriert.

Acknowledgements

Back in 2020, after I completed my master's thesis at my parents' house due to a chaotic coronavirus year, I found myself sitting in a job interview with Heiner. The interview started with him skimming through my documents, then looking back up at me, subsequently forming a thumbs-up with his hand while commenting, "Not bad, not bad, Patrick!" It was a small gesture that had a great influence on my decision to pursue a PhD under his watch. The following years of cooperation had a similarly welcoming dynamic. Heiner gave me the flexibility to work freely on my research while funding it, supported me with valuable feedback and guidance when needed, and connected me with important contacts. I am truly grateful for these opportunities.

Through Heiner, I learned about the works of Fabian Suchanek, and I want to thank him for his valuable input regarding rule-based systems and for his incredible contributions to the knowledge graph community. Some years later, I was introduced to Christian Bartelt, and I want to thank him for teaching me so many valuable skills needed to be successful in the academic world.

Right at the beginning of my PhD, I was introduced to Christian Meilicke. We had actually known each other for a little while; in fact, I took his Relational Learning course in 2019 at the University of Mannheim. Of course, the exam was too hard and, unfortunately, my performance was less than glorious. Christian has the brilliant ability to break every complicated topic into small, comprehensible pieces. He became a good friend and was a great mentor, which made me the scientist I am today. For that, I am truly grateful.

Needless to say, pursuing a PhD can be quite a ride. While the majority of this journey is shared with your colleagues, parts of it are shared with your family. My parents have always believed in me and supported me emotionally and financially throughout my studies. The support and love they have shown is more than anyone could ever wish for. To my sister, her husband, and my sweet little nephew - even if I should truly visit you more often - I am thankful and proud that you are a part of my family.

Contents

Abstract	iii
Zusammenfassung	v
Acknowledgements	vii
1 Introduction	1
1.1 Rule-Based Knowledge Graph Completion	1
1.1.1 Knowledge Graph Completion	2
1.1.2 Rules for Knowledge Graphs	2
1.1.3 Inference with Rules	3
1.1.4 Challenges	4
1.2 Embedding-Based Completion	4
1.3 Research Questions and Contribution	5
1.4 Thesis Structure	7
2 Preliminaries	9
2.1 Knowledge Graphs	9
2.1.1 Taxonomies and Types	10
2.1.2 Constructing Knowledge Graphs	10
2.1.3 Assumptions About Knowledge Graphs	11
2.2 Knowledge Graph Completion	12
2.2.1 Ranking-Based Evaluation	13
2.2.2 Protocol and Metrics	15
2.2.3 Scientific Benchmarks for KGC	16
2.3 Rules for Knowledge Graphs	17
2.3.1 Rules	17
2.3.2 Rule Types	18
2.3.3 Formal Rule Types	19
2.3.4 Logical Formulation of Rules	20
2.4 Inference with Rules	20
2.4.1 Rule Predictions	20
2.4.2 Uncertainty and Rule Confidences	21
2.4.3 Rules and Reasoning	22
2.5 Rule Confidence Aggregation	23
2.5.1 Problem Setting	24
2.5.2 Aggregation Functions	24
2.5.3 Theoretical Considerations	25
2.5.4 Practical Considerations	26
2.6 Rule Learning	27
2.6.1 AnyBURL	27
2.6.2 AMIE	29
2.6.3 Other Rule Learning Systems	30

2.6.4	Rule-Based KGC: A Summary	30
2.7	Knowledge Graph Embedding Models	31
2.7.1	Scoring Functions	31
2.7.2	Negative Examples	33
2.7.3	Training	34
2.7.4	Variations and Hyperparameters	34
2.8	Graph Neural Networks	35
2.8.1	Relational Message Passing	36
2.8.2	Query-Dependent Message Passing	36
3	Comparing Rules and KGE Models	39
3.1	Related Work	40
3.2	Qualitative Comparison	41
3.2.1	Challenges of Model Comparison	41
3.2.2	Metric Computation	42
3.2.3	Selected Cases	42
3.2.4	Influential Facts	45
3.2.5	Summary	46
3.3	Quantitative Analysis	46
3.3.1	Differences in Ranking Calculation	47
3.3.2	KGE-Rescoring	47
3.4	Experiments	48
3.4.1	Experimental Setup	48
3.4.2	Summarized Comparison	49
3.4.3	Detailed Comparison	50
3.4.4	KGE-Rescoring Results	51
3.4.5	Averaging KGE Models	53
3.4.6	Rule-Based Ablations	54
3.5	Conclusion	54
4	Explaining KGE Models with Rules	57
4.1	Problem Setting	58
4.1.1	Attacks and Explanations	58
4.1.2	Attack Protocol	58
4.2	Related Work	59
4.3	Method	60
4.3.1	Abductive Reasoning	60
4.3.2	Calculating Explanations	61
4.3.3	Defining the Attack	61
4.3.4	Discussion	63
4.4	Experiments	63
4.4.1	Experimental Settings	64
4.4.2	Evaluation Protocol	64
4.4.3	Results	65
4.4.4	Rule Ablations	66
4.4.5	Understanding the Explanation	67
4.4.6	Performing Multiple Attacks	68
4.5	Conclusion	69

5	Confidence Aggregation: A Closer Look	71
5.1	Related Work	72
5.2	Probabilistic Confidence Aggregation	73
5.2.1	Formal Representation	74
5.2.2	Dealing with Uncertainty	74
5.2.3	Inference for Target Facts	75
5.2.4	Recovering Aggregation Functions	77
5.2.5	Discussion	79
5.2.6	Mixing Assumptions	80
5.3	ProbLog and Confidence Aggregation	81
5.3.1	ProbLog and Noisy-or Aggregation	81
5.3.2	Aggregating Logic Programs	82
5.3.3	Additional Examples	83
5.3.4	Discussion	84
5.4	Markov Logic and Confidence Aggregation	84
5.4.1	Representation	84
5.4.2	The Probability of a World	85
5.4.3	Incorporating Evidence and KGs	86
5.4.4	Discussion	87
5.5	Experiments	88
5.5.1	Experimental Setup	88
5.5.2	Results	89
5.5.3	Multi-Step Reasoning	90
5.6	Conclusion	91
6	Supervised Confidence Aggregation	93
6.1	Related Work	94
6.2	Learning from Noisy-or	95
6.2.1	Motivation	95
6.2.2	Differentiability	96
6.2.3	Base Model	96
6.2.4	Generalisation	98
6.2.5	Disjunct Data	99
6.2.6	Expressiveness	99
6.3	Incorporating the Max Assumption	100
6.3.1	Sparse Aggregation	100
6.3.2	Properties	101
6.3.3	Learning the Model	102
6.4	Dense Aggregation	104
6.5	Experimental Setup	105
6.5.1	Dataset Construction	105
6.5.2	Experimental Setting	106
6.5.3	Model Specifications	107
6.5.4	Training Details	107
6.6	Experimental Results	108
6.6.1	KGE Models vs. Learned Aggregation	109
6.6.2	Benefits of Learning the Aggregation	109
6.6.3	Other Works	110
6.6.4	WN18RR and GNNs	111
6.7	Conclusion	113

7	Comparisons Beyond KGE Models	115
7.1	The Synthetic Zoo Dataset	117
7.1.1	Experiments	117
7.1.2	Results	118
7.1.3	Discussion	119
7.2	Negative Patterns on WN18RR	120
7.2.1	Fine-Grained Model Comparison	120
7.2.2	Only-One-Tail Rule	121
7.2.3	Perturbation Experiments	122
7.2.4	Perturbation-Based Results	123
7.2.5	Augmenting the Rule-Based Approach	124
7.3	The Synthetic Uni Dataset	126
7.3.1	Experiments on the Uni KG	127
7.3.2	Discussion	128
7.4	Negative Pattern on Fb15k-237	128
7.4.1	Perturbation-Based Experiments	129
7.4.2	Perturbation-Based Results	129
7.4.3	Post-Hoc Experiments	130
7.5	Discussion and Conclusion	132
7.5.1	Differences in Predictive Performance	132
7.5.2	Improving the Rule-Based Approach	132
8	Conclusion and Outlook	135
8.1	Discussion and Limitations	135
8.2	Future Work	136
8.2.1	Number of Learned Rules	136
8.2.2	Additional Applications for Rules	136
8.2.3	Augmenting the Language Bias	137
	Bibliography	139
A	Appendix	147
A.1	Proofs For Chapter 5	147
A.2	Ablations for Chapter 6	153

List of Figures

2.1	A small knowledge graph.	9
3.1	The MRRs of the best KGE model on each dataset compared with Max+ aggregation.	49
3.2	MRRs of Max+, original KGE model (bottom), and KGE-Rescoring (top)	51
4.1	Results for the decline in H@1 for ComplEx on Fb15k-237 when deleting/adding multiple facts for each target fact.	69
5.1	ProbLog program based on a modification of Example 18.	83
5.2	ProbLog program where one rule is the more general form of another rule.	83
6.1	Left: Example dataset with two training facts and two rules. Right: The total loss as function of the Noisy-or product over both rules (right).	97
6.2	The best KGE model compared with the learned aggregation. The red lines depict Max+ aggregation results.	109
7.1	Summarized MRR results.	115
7.2	Graphical representation of the Zoo dataset. The task is to determine who follows the node <i>anna</i> . The correct answer <i>bobby</i> seems to be most intuitive as it visually closes the gap of the graph. However, it is less obvious which specific pattern in the data might support this conclusion and can be learned by a KGC model.	117
7.3	Relative relation-wise performance comparison for the head direction. NBFNet is one. The first four relations with the most facts on the test set are shown (largest first). <i>Hypernym</i> consists of 1251 facts while <i>has-part</i> only consists of 172 facts.	121
7.4	Graphical representation of the uni dataset. Each node is connected via the <i>member</i> relation (dashed arrows) to the <i>uni</i> node. The task is to move the red arrow pointing either to <i>bernd</i> or <i>anna</i> . The dataset regularities clearly suggest <i>anna</i> ; there is no reason why the arrow should be pointing to <i>bernd</i>	127
7.5	MRR results for WN18RR and Fb15k-237. The lower dashed lines depict the Max+ result. The upper dashed lines depict the LR results.	133
A.1	MRR in tail direction for five runs when training under the scaled/non-scaled gradient.	153
A.2	Valid MRR per epoch on Codex-M for Mean-Rank training (\star) and ordinary Cross-Entropy loss (+).	153

List of Tables

1.1	A subset of a knowledge graph.	2
2.1	A KG as a collection of triples.	10
2.2	A tail and head ranking and scores with respect to queries formed from the true base fact <i>speaks(bernd, english)</i>	14
2.3	KG benchmark summary statistics.	16
3.1	Hypothetical rankings for the query <i>speaks(lisa, ?)</i> with correct answer <i>english</i>	41
3.2	Rankings for the completion task <i>contains(?, Darwin)</i> based on Codex-M.	43
3.3	Ranking results for query <i>producedBy(TombRaiderI,?)</i> on Codex-M.	44
3.4	Ranking results for the query <i>influencedBy(schumann,?)</i> on Codex-M.	44
3.5	The impact of removing f_1 and f_2 on the scores and ranking position of King’s College.	46
3.6	Number of facts and learned rules on each benchmark.	48
3.7	Detailed results for the KGC task on Fb15k-237 and WN18RR.	50
3.8	Detailed results for the KGC task on Codex-M and Codex-L.	51
3.9	Detailed results for the rescoring experiments. The first column section shows the results when proposing candidates with the rules and rescoring them with the respective KGE model. The second column section shows the improvement over the respective original KGE results provided in Section 3.4.3.	52
3.10	The proportion of correct candidates missing from the top-100 answers for a Max+ ranking on the different benchmarks.	52
3.11	Example results for WN18RR when training models five times. Best and worst results are shown as well as an average ensemble of all runs (joint).	53
3.12	Ablation results for selected datasets. Performance results when using different subsets of rules are shown.	54
4.1	The degradation in MRR for ComplEx on Fb15k-237 in the <i>Del</i> setting when only following the existing protocol without actually performing an attack and without retraining (middle column).	65
4.2	Results for WNRR. All results are averages over five runs. Lower is better. The original MRR and H@1 values are 1.0 in all specifications.	66
4.3	Results for Fb15k-237. All results are averages over five runs. Lower is better. The original MRR and H@1 values are 1.0 in all specifications.	66
4.4	Ablation results for ComplEx on WN18RR. Lower is better. The original MRR and H@1 values are 1.0 in all specifications. In the parentheses, e.g., <i>l1</i> refers to a rule length of one.	67
5.1	Fine-grained probability calculation with ProbLog for target fact <i>speaks(marta, english)</i>	82

5.2	The possible world η_1	85
5.3	Results for MRR, H@1, and H@10.	89
5.4	Ablation results for multi-step reasoning on Fb15k-237 and Max+.	91
6.1	MRR, Hits@1, Hits@10 results for Fb15k-237, WN18RR, and Codex-M.	111
6.2	MRR, Hits@1, Hits@10 results for Fb15k-237 and WN18RR.	112
7.1	MRR results for the zoo dataset. We report averages (\pm std. dev.) over 10 runs. Random ranking in both directions results in an MRR of approximately 0.02.	118
7.2	MRR head (H) and tail (T) results for WN18RR.	120
7.3	Average head score changes on WN18RR after perturbation experiments with respect to target queries $_hypernym(?, entity_i)$ based on the test set.	123
7.4	Effects on the MRR for the head direction of the <i>hypernym</i> relation and overall when adding the influential fact or a random fact individually w.r.t each test fact.	124
7.5	Possible existence feature types that can be created for one target fact where p and q are relations. The overall number of features is $4 \mathcal{P} ^2$	125
7.6	MRR results for WN18RR. MRR (T) and MRR (H) are tail and head MRRs over all relations. The third column is the joint MRR.	126
7.7	MRR results for the uni dataset. We report averages (\pm std. dev.) over 10 runs.	127
7.8	Fb15k-237 (Tail direction) perturbation experiments for the default models trained with ROH=on.	130
7.9	MRR and Hits effects on Fb15k-237 when adding the influential fact or adding a random fact. Left: The original model specification. Right: The original specification while ROH is turned off. The add scenario essentially leaks the correct answer by a random connection. For the ROH specification this leads to a decrease in the performance (left) whereas it leads to an improvement when the hyperparameter is turned off (right).	131
7.10	MRR results for Fb15k-237 and post-hoc filter experiments for all model classes. The post-hoc filter does almost have no have effect for the original GNN specifications as the OOL rule is already exploited. If ROH is turned off, enforcing the OOL pattern by the filter leads to an increase in performance.	131
A.1	All probability values.	151

List of Abbreviations

BCE	Binary Cross-Entropy
BFS	Breadth First Search
BLP	Bayesian Logic Program
CE	Cross-Entropy
CWA	Closed World Assumption
DFS	Depth First Search
GCN	Graph Convolutional Neural Network
GNN	Graph Neural Network
ILP	Inductive Logic Programming
KG	Knowledge Graph
KGC	Knowledge Graph Completion
KGE	Knowledge Graph Embedding
LPLP	Lifted Probabilistic Logic Programming
MLN	Markov Logic Network
MR	Mean Rank
MRR	Mean Reciprocal Rank
OOL	Only One Link
OOT	Only One Tail
OWA	Open World Assumption
PCA	Partial Closed World Assumption
PLP	Probabilistic Logic Programming

List of Core Symbols

\mathbb{R}	real numbers
\mathbb{N}	natural numbers
\mathcal{G}	knowledge graph
\mathcal{E}	set of entities
\mathcal{P}	set of relations
$s, o, e \in \mathcal{E}$	entities
$p \in \mathcal{P}$	relation
$p(s, o) \in \mathcal{G}$	fact, where $s, o \in \mathcal{E}$ and $p \in \mathcal{P}$
$f \in \mathcal{G}$	fact (abbreviated)
ϕ	fact scoring function
$p(s, ?)$	tail query
$p(?, o)$	head query
q	query (abbreviated)
B	binary (cyclical) rule type
U_d	unary rule type
U_c	unary rule type
U_z	empty body rule type
τ	term
a	binary atom
X, Y, A, B, \dots	variables
θ	substitution
$conf$	any rule measure
\mathcal{R}	set of rules
$r \in \mathcal{R}$	one rule
$\mathcal{R}_f(\mathcal{G})$	subset of rules that predict fact f w.r.t. KG \mathcal{G}
\mathcal{R}_f	subset of rules that predict fact f (KG is implicit)
E	an explanation (set of facts)
I	index set
R_i	random variable representing rule r_i
P	probability distribution
π	marginal probability
$s \in \mathbb{R}^d$	embedding vector for entity s
$p \in \mathbb{R}^d$	embedding vector for relation p
$M^p \in \mathbb{R}^{d \times d}$	matrix embedding for relation p
l	loss function
Ω	model parameters
$\frac{dl}{d\Omega}$	gradient of the loss function w.r.t. model parameters
$y_f \in \{0, 1\}$	label for fact f
σ	sigmoid function
x_i	binary rule feature representing rule i
ω_i	learnable rule parameter

...

Chapter 1

Introduction

Relational data represents the world by modelling relationships between distinct objects. The granularity of these relations can range from encoding gene associations (Himmelstein et al., 2017), over certain semantic connections of specific word types (Miller, 1995), up to describing correspondences between countries (Bollacker et al., 2008). Inferring previously unknown statements about these domains requires to learn from relational data and remains a challenging problem in various research communities.

A popular form of storing relational data is given by knowledge graphs (Nickel et al., 2015). They are collections of symbolic facts that are composed of entities and relations and describe true statements about the respective domain. Knowledge graphs can contain millions or even billions of such facts, demonstrating their scalability for representing large and complex domains (Suchanek, Kasneci, and Weikum, 2007; Bollacker et al., 2008; Dong et al., 2014).

The process of verifying potential facts in a knowledge graph can be resource intensive. Therefore, most of the existing knowledge graphs are incomplete. That is, there exist facts that are correct in the domain but they are missing from the respective knowledge graph (Nickel et al., 2015). The focus of this thesis is the problem of knowledge graph completion which is concerned with deriving previously unknown but correct facts. Approaches that can complete the knowledge graph need to learn patterns from the existing facts and apply them to infer new facts. A main characteristic that distinguishes these approaches is given by the employed form of representation to encode the learned patterns.

1.1 Rule-Based Knowledge Graph Completion

Rule-based approaches employ symbolic representations to express patterns of a knowledge graph. They learn symbolic rules that describe human-readable logical implications (Galárraga et al., 2013; Meilicke et al., 2024). With the learned rules, new fact prediction can be derived by using a form of logical reasoning that requires their antecedents to be true when treating the existing facts as evidence. This approach is fully transparent and the resulting predictions are explainable.

Conceptually, a rule-based approach can be separated into two stages. In the learning stage, rules must be mined efficiently from the given KG. Subsequently, in the inference stage, the rules have to be applied for the given task at hand. There exists a considerable amount of research on how to learn rules from knowledge graphs (Galárraga et al., 2013; Galárraga et al., 2015; Ortona, Meduri, and Papotti, 2018; Meilicke et al., 2019; Sadeghian et al., 2019; Wu et al., 2022; Meilicke et al., 2024). Similarly, performing inference within different logical frameworks has been studied for decades (Bratko, 2001; De Raedt, 2008; Baader, Horrocks, and Sattler, 2008). Performing inference with rules for knowledge graph completion, however, comes

subject	relation	object
Anna	internAt	Google
Anna	expertIn	Computer Science
Bernd	livesIn	Mountain View
Microsoft	employerOf	Lisa
Lisa	internAt	Microsoft
Google	industrySector	Computer Science
...

TABLE 1.1: A subset of a knowledge graph.

with problem specific intricacies such as the sizes of knowledge graphs, the large number of learned rules, the involved uncertainty, and the relatedness of different rules. Overcoming these challenges is the main goal of this thesis and the problem setting will be introduced in the following.

1.1.1 Knowledge Graph Completion

Each fact in a knowledge graph consists of a subject entity, an object entity, and a relation specifying the connection between them. A small part from a knowledge graph is shown in Table 1.1. The facts describe persons and companies and different relationships between them. While it is typically assumed that each fact in the knowledge graph represents a correct statement (Nickel et al., 2015), it is not guaranteed that all correct statements about the domain are contained. For instance, due to this incompleteness, the knowledge graph might not explicitly include the fact that Anna is employed at Google, even if it is true in the real world.

The search for and evaluation of new facts based on external sources to complete a knowledge graph is cumbersome and expensive. A more efficient alternative involves leveraging the existing knowledge graph to derive new facts. Therefore, the focus of this thesis lies on approaches that learn patterns from an existing knowledge graph and apply them to complete the knowledge graph. For instance, an approach could learn that companies tend to hire individuals that are experts in their respective industry sectors and apply this pattern to infer possible employment relationships between persons and companies.

Example 1 (Knowledge Graph Completion). Let us assume that the knowledge graph contains the relation *employerOf* which describes actual employers of persons. Let us further assume that nothing is known about the employment positions of Anna in the knowledge graph. The task is to complete the knowledge graph in regard to this relation. That is, we search for plausible employers of Anna.

1.1.2 Rules for Knowledge Graphs

One possibility for proposing possible employers of Anna is to employ symbolic rules that are learned beforehand from the existing knowledge graph. They are composed of the existing relations as shown in the following example.

Example 2 (Rules). Let us assume that the following rules have been learned by a mining system from the knowledge graph:

$$\begin{aligned} r_1 [0.64]: & \text{employerOf}(X, Y) \leftarrow \text{internAt}(Y, X) \\ r_2 [0.32]: & \text{employerOf}(X, Y) \leftarrow \text{industrySector}(X, A), \text{expertIn}(Y, A) \\ r_3 [0.25]: & \text{employerOf}(\text{Google}, X) \leftarrow \text{livesIn}(X, \text{MountainView}) \end{aligned}$$

The symbols X, Y and A are variables. Precisely, *employerOf* describes that X is the employer of Y . The relation *industrySector* describes that an entity or company X operates in the sector A , the remaining relations have intuitive meanings. The numbers in brackets will be described further below.

The rules encode logical if-then statements in regard to the knowledge graph. In particular:

1. The first rule expresses that if somebody interns at a company, they might be employed at that company.
2. The second rule says that somebody who is an expert in a field might work for a company that operates in that field.
3. The last rule is a special type and it contains the entities Google and Mountain View from the knowledge graph. It says that if someone lives in Mountain View, they might work for Google.

The rule-mining approaches considered in this thesis are particularly designed for large knowledge graphs. This aspect, combined with the fact that the original knowledge graph is incomplete in the first place, induces intricacies: Unfortunately, the learned rules are subject to noise and uncertainty. The expressed if-then statements are not strictly deterministic. For instance, clearly, not every person that lives in Mountain View also works for Google. The numbers in brackets from Example 2 are termed the rule confidences and they allow to quantify the uncertainty of the respective rules. The values are estimated from the knowledge graph and they express the likelihood that a prediction made by a rule is true which will be discussed in the following.

1.1.3 Inference with Rules

With the rules, potential employers of Anna can be proposed by making rule predictions. For instance, one fact from the knowledge graph shown in Table 1.1 states that Anna interned at Google. Therefore, the first rule in Example 2 is activated as its right-hand side reflects an existing fact when substituting Y with *Anna* and X with *Google*. This substitution translates to the left-hand side of the rule, resulting in the fact *employerOf(Google, Anna)*. Thus, the rule predicts that Anna works for Google. With a similar approach, rule predictions can be obtained for more complicated rules. The predictions are fully explainable. For instance, if a potential user is curious about why the prediction was made, we can present the respective rule and the fact that Anna interned at Google as a reason for the activation of the rule.

Unfortunately, given the uncertainty associated with the rules, we cannot be certain that the predicted fact is true. To obtain an estimate for the likelihood of the prediction, the confidence of the rule could be used. However, Anna is also an expert in Computer Science and therefore the second rule likewise predicts Anna to be employed at Google due to the alignment of the industry sector. Therefore, we

have two distinct likelihood estimates for the same prediction and it is unclear what the final value should be.

1.1.4 Challenges

One of the challenges in completing a knowledge graph is given by the requirement of determining an order for predictions based on their likelihood or plausibility.

Example 3 (continued). Suppose that Bernd is predicted by the third rule from Example 2 to work for Google. Anna, on the other hand, is predicted by the first and second rule to work for Google. Which of the predictions is more plausible?

For comparing the predictions according to their plausibility, a strategy is required to translate the different rules that make the same prediction into one meaningful value. A typical approach is to define a function that aggregates the individual rule confidences. However, the task can become challenging as there could be hundreds of rules in a practical scenario that could predict Anna or Bernd to work for Google.

In Example 3, it might seem that the prediction for Anna is more likely as only a single rule predicts that Bernd works for Google. Nevertheless, relying on the count of predicting rules can be misleading as they can be partly redundant. For instance, consider the following rule:

$$r_4 [0.28]: \text{employerOf}(X, Y) \leftarrow \text{industrySector}(X, A), \text{graduatedIn}(Y, A)$$

The rule is similar to the second rule shown above as we expect a graduate in a field to also be an expert in that field. When learning rules automatically from a knowledge graph, many instances of similar rules can be learned.

If Anna is additionally predicted by this rule to work for Google, it may not count as substantial new evidence since a similar rule already made the prediction. Conversely, if the rule additionally predicted Bernd to work for Google, it may be perceived as complementary evidence and should increase the likelihood score of the prediction. Therefore, the rules cannot be considered to be independent of each other when making the same predictions. The problems considered in this thesis are particularly challenging since the respective knowledge graphs can be large with millions of learned rules. In these cases, we need to rely on automatic and data efficient approaches to infer new predictions and their likelihood scores.

1.2 Embedding-Based Completion

An alternative and popular approach for learning from knowledge graphs is given by sub-symbolic methods, also known as knowledge graph embedding models (Rossi et al., 2021; Ruffinelli, Broscheit, and Gemulla, 2020). In contrary to rule-based approaches that use explicit discrete structures, embedding-based models leverage continuous representations to capture latent and semantic information from a knowledge graph. In particular, embedding models encode each entity and relation into a continuous latent vector - commonly known as the embeddings. These embeddings are learned beforehand from a knowledge graph during the training stage. Once learned, the embeddings allow the calculation of fact scores by vector operations to determine the likelihood of unknown facts in the context of knowledge graph completion.

For instance, suppose that, after the training of an embedding model, we obtain the following embedding vector representations for Google, Anna, and the relation *employerOf*.

$$\overrightarrow{\text{Google}} = \begin{bmatrix} 1.3 \\ 0.2 \\ -0.9 \end{bmatrix}, \quad \overrightarrow{\text{Anna}} = \begin{bmatrix} 1.5 \\ -0.7 \\ 0.1 \end{bmatrix}, \quad \overrightarrow{\text{employerOf}} = \begin{bmatrix} 0.9 \\ -0.2 \\ 0.4 \end{bmatrix}$$

The vector entries are the model weights that are learned during the training stage. Although the values cannot be interpreted explicitly, they hopefully capture semantically rich characteristics such that related entities in the knowledge graph have a high vector similarity.

In the context of knowledge graph completion, embedding models employ a scoring function to calculate real-valued fact scores where highly plausible facts should obtain large values whereas nonsensical facts should obtain small values. For instance, following the last section, suppose that we seek to calculate the plausibility score that Anna is employed at Google. Therefore, we seek to calculate the score of the fact $f = \text{employerOf}(\text{Google}, \text{Anna})$. A popular scoring function ϕ is given by the DistMult model (Yang et al., 2015). It calculates the coordinate-wise product of the embeddings of Anna, Google, and *employerOf* and subsequently takes the sum:

$$\phi_{\text{Distmult}}(f) = \text{SUM} \begin{pmatrix} 1.3 \cdot 0.9 \cdot 1.5 \\ 0.2 \cdot -0.2 \cdot -0.7 \\ -0.9 \cdot 0.4 \cdot 0.1 \end{pmatrix} = \text{SUM} \begin{pmatrix} 1.76 \\ 0.03 \\ -0.04 \end{pmatrix} = 1.75$$

A higher value indicates a greater likelihood that the fact is correct and should be included in the knowledge graph. Additionally, the value can readily be compared with, say, the score that Bernd works for Google. To calculate this score, the embedding of Anna has to be substituted with the embedding of Bernd in the calculation above.

1.3 Research Questions and Contribution

The goal of this thesis is to explore rule-based approaches for knowledge graphs with the particular focus on how to effectively apply a previously learned collection of rules. We argue that, to truly uncover the applicability of the symbolic approach, it should be put into reference of and compared with another paradigm that has shown to work well for the respective tasks.

Therefore, we will additionally investigate the connection between rule-based approaches and embedding models in the context of knowledge graph completion. In particular, in this thesis, we seek to answer the following questions:

- (i) How does a rule-based approach compare to embedding models for knowledge graph completion? What are the similarities and differences in the learned patterns of both approaches?
- (ii) How can large sets of rules effectively be used to achieve a high predictive performance? How can the theoretical formalism be described and what are the practical means that need to be employed?
- (iii) What are the limitations of a rule-based approach? How can it be augmented to obtain a higher expressiveness?

Exploring these questions will lead to the following main contributions of the thesis.

- (i) We conduct empirical studies to compare embedding models and a rule-based approach. While we find differences of the two model classes, we also find that often they can learn similar patterns.
- (ii) We show that the explanatory power of a rule-based approach can be used to explain the predictions of an embedding model.
- (iii) We both theoretically and empirically explore how rules can be effectively used to make new fact predictions for knowledge graph completion. We can improve over previous methods and demonstrate how to close the gap to embedding models regarding the predictive performance.
- (iv) We identify patterns that cannot be expressed with the rules in comparison to newer graph neural network architectures and demonstrate how to augment the expressiveness of the rule-based approach.

Publications

The contents of this thesis are based on the following works.

- Meilicke, Christian, and Betz, Patrick, and Stuckenschmidt, Heiner (2021). "Why a naive way to combine symbolic and latent knowledge base completion works surprisingly well". In: 3rd Conference on Automated Knowledge Base Construction.
- Betz, Patrick, and Meilicke, Christian, and Stuckenschmidt, Heiner (2022a). "Adversarial explanations for knowledge graph embedding models". In: Proceedings of the 31th International Joint Conference on Artificial Intelligence.
- Betz, Patrick, and Meilicke, Christian, and Stuckenschmidt, Heiner (2022b). "Supervised knowledge aggregation for knowledge graph completion". In: Extended Semantic Web Conference.
- Ott, Simon, and Betz, Patrick, and Stepanova, Daria, and Gad-Elrab, Mohamed H., and Meilicke, Christian, and Stuckenschmidt Heiner (2023). "Rule-based knowledge graph completion with canonical models". In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management.
- Betz, Patrick, and Lüdtko, Stefan, and Meilicke, Christian, and Stuckenschmidt, Heiner (2024). "Rule confidence aggregation for knowledge graph completion". In: International Joint Conference on Rules and Reasoning.
- Betz, Patrick, and Stelzner, Nathanael, and Meilicke Christian, and Stuckenschmidt, Heiner and Bartelt, Christian (2024). "A*Net and NBFNet learn negative patterns on knowledge graphs". CoRR.

In each chapter, the respective reference work will be highlighted in the introduction. Furthermore, during our research, we developed the Python-based library PyClause with the goal of making rule-based functionalities easier accessible to a broader machine learning community.

- Betz, Patrick and Galárraga, Luis, and Ott, Simon, and Meilicke, Christian, and Suchanek, Fabian M., and Stuckenschmidt, Heiner (2024). "PyClause - Simple and efficient rule handling for knowledge graphs". In: Proceedings of the 31th International Joint Conference on Artificial Intelligence, demo track. Achieved best demo award.

Finally, the following two works are not part of this thesis but contributing to them helped me to gain a deeper understanding of rule-based approaches and embedding models.

- Meilicke, Christian and Chekol, Melisachew W. and Betz, Patrick and Fink, Manuel and Stuckenschmidt, Heiner (2024). "Anytime bottom-up rule learning for large-scale knowledge graph completion". In: The VLDB Journal.
- Broscheit, Samuel and Ruffinelli, Daniel and Kochsiek, Adrian and Betz, Patrick and Gemulla, Rainer (2020). "LibKGE-A knowledge graph embedding library for reproducible research". In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.

1.4 Thesis Structure

The next chapter introduces the required preliminaries and concepts needed throughout the thesis. A large part of the notation will be introduced in the chapter. Specific notation and symbols that are only relevant for particular parts will be introduced also in later chapters on the fly. The remainder of the thesis is structured as follows.

Rules and Embedding Models

Chapters 3 and 4 are concerned with exploring the relation and interplay of rule-based methods and embedding models. Initially, the two classes will be compared on real benchmarks and subsequently, we investigate how they rely on the same patterns when making predictions.

In Chapter 3, we will compare rule-based methods and embedding models on real benchmarks. We will start with presenting example cases that allow us to contrast the behaviour of both model classes. Subsequently, we will compare the models quantitatively based on the knowledge graph completion task. We also present a variety of custom experiments and ablations and seek to understand the difference or similarities of patterns the models use for performing the task. We will find that, despite the best embedding models performing slightly superior on average, the models learned similar patterns to some extent.

In Chapter 4, motivated by the previous chapter, we seek to quantify to which extent the models rely on the same regularities. We do this by proposing a method that allows to explain predictions of an embedding model with the rule-based approach. We compare this method with alternative approaches in the context of adversarial attacks. We will find that our method performs on par with the state-of-the-art despite being independent of the embedding model. Important facts, which influence the predictions of the embedding models, can be identified with the rules.

Improving Inference with Rules

The findings of the previous two chapters demonstrate that embedding models are slightly superior in regard to the predictive performance when completing a knowledge graph. However, they also suggest that the learned rules potentially should be sufficient to achieve equal performance. Therefore, Chapters 5 and 6 are concerned with improving the predictive capabilities of a given learned set of rules.

In Chapter 5, we will explore how rule inference can be improved based on a theoretical analysis of the prediction problem. From deriving a formal framework, we are able to compare different approaches in a unified probabilistic model. We will also answer the question if classical reasoning, which has been used for decades in the logical community, could be beneficial for the completion problem. We will find that the previously discussed approaches rely on assumptions about the rules that cannot hold in practice when rules are learned automatically.

Therefore, in Chapter 6, we propose to augment the rule-based approach with a classical supervised learning step. The used objective function is inspired by the training objective employed for embedding-based models. We will find that augmenting the rule application stage with supervised learning can improve the predictive performance. Therefore, we are able to show that the rule-based approach performs slightly superior to the embedding models.

Beyond Embedding Models

The experimental results in previous chapters point towards an alternative method for completing the knowledge graph given by Graph Neural Networks which are shown to be superior. To uncover the full potential or limitations of the rule-based approach, we have to go beyond the comparison against simple embedding models. To that end, Chapter 7 will analyse these results for improving the rule-based approach further and Chapter 8 provides an outlook for directions beyond the topics discussed in the thesis.

In Chapter 7, we have a close look at an alternative method given by Graph Neural Networks which report remarkable results on the completion task. We seek to answer the question if they rely on patterns that cannot be expressed with the rules or if rules can potentially achieve the same predictive performance. Based on synthetic datasets, we find cases where the embedding models and the rule-based approach behave similarly, whereas the Graph Neural Network behaves differently. Furthermore, we find that the models exploit negative patterns that allow them to disregard incorrect proposals. Although these patterns are not expressible with the vanilla rule-based approach, we discuss simple methods of how to imitate this behaviour.

In Chapter 8, we conclude the thesis and point towards different directions of how to overcome the discovered limitations of the rule-based approach. For instance, future work could investigate how the approach can be augmented to incorporate new rule types.

Chapter 2

Preliminaries

2.1 Knowledge Graphs

Knowledge Graphs (KGs) describe real-world domains by using structured predicate-subject-object facts. Facts are written in the form $p(s, o)$ or equivalently as triples (s, p, o) where s and o denote entities of the KG. We will call p the relation of the fact which is the common term in the context of KGs. From a logical perspective, p is a binary predicate. Furthermore, we call s the subject or the head entity of the fact and likewise o is the object or the tail entity of the fact.

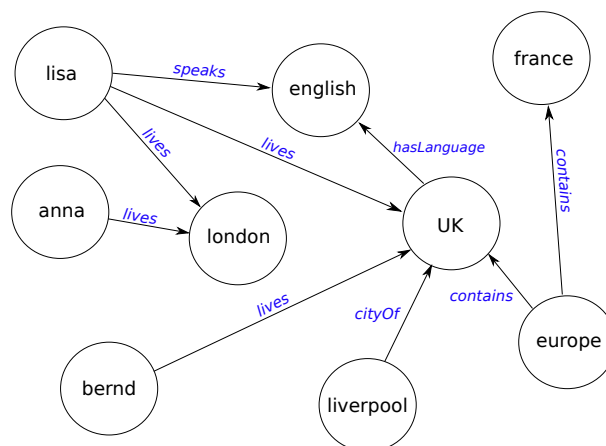


FIGURE 2.1: A small knowledge graph.

An entity in the KG can represent any real or abstract concept that can be described in the world. This includes persons or objects, such as Barack Obama, the ISS, or the Eiffel Tower, as well as abstract concepts like love, human, and building. The decision of what constitutes an entity depends exclusively on the designer of the KG.

The relations, on the other hand, are binary relations over the entities and describe certain relationships between them. Some possibilities matching the entities given above could be, for example, *has profession*, *likes*, or *contained in*. Also more abstract relations are possible such as *similar to*, *related to*, or *part of*. The facts contained in a KG are true statements about the world or the domain that it describes. Formally, a KG is defined as follows:

Definition 1 (Knowledge Graph). *Let \mathcal{E} be a set of entities and \mathcal{P} be a set of relations. A knowledge graph \mathcal{G} is a collection of facts, $p(s, o)$, where $s, o \in \mathcal{E}$ and $p \in \mathcal{P}$. Let $\mathcal{P} \times \mathcal{E} \times \mathcal{E}$ denote the set of all possible facts. Then, it holds that $\mathcal{G} \subseteq \mathcal{P} \times \mathcal{E} \times \mathcal{E}$.*

A KG can be represented as a directed graph where the subject of a fact is the source node of an edge, the object is the target node, and the relations describe different

edge labels. Figure 2.1 depicts an example KG in graph form and Table 2.1 shows the representation as a set of triples.

Knowledge Graph		
subject	predicate	object
lisa	speaks	english
lisa	lives	UK
lisa	lives	london
bernd	lives	london
europe	contains	UK
...

TABLE 2.1: A KG as a collection of triples.

2.1.1 Taxonomies and Types

The example KG in Figure 2.1 describes persons, cities and countries. One fact states that the person Lisa lives in the city London and another one states that she speaks the language English. Here, we have explicitly used the types *person*, *city*, and *language* to further describe the entities. These type descriptions could be included explicitly in the KG. For example, the KG could contain facts that state *hasType(lisa, person)* or *hasType(london, city)*. Furthermore, a KG might contain facts such as *subclassOf(politician, person)* that describe taxonomic relations between entities. In this case, we can say that type descriptions are *class* assertions and that the *class* politician is a subclass of person, i.e., every politician must be a person.

Collecting this kind of knowledge describing types and taxonomic relationships between classes is also described by the more general terms knowledge bases (e.g., Suchanek et al., 2019) or ontologies (e.g., Ding et al., 2007). Here, storing the taxonomic relations and class assertions is separately described by the specification of a *schema*. On the other hand, the definition of KGs from above does not explicitly take these considerations into account. In general, within this thesis, we distinguish a *Knowledge Graph* from a *Knowledge Base* by its focus on a flat collection of facts composed of entities and relations without the explicit modelling of a schema.

Although processing schema information explicitly can provide useful applications of KGs, constructing and maintaining consistent and coherent schemas is as expensive as constructing KGs in the first place. Moreover, the methods that we will explore in this thesis can readily be used to infer schema relations, for instance, inferring facts for the *subclass* or *type* relations. Finally, for some of the benchmarks that will be used, schema relations are contained within the KG. Therefore, they are implicitly used when learning rules on the KGs.

2.1.2 Constructing Knowledge Graphs

KGs can describe any potential topic or domain. For instance, in the biomedical KG Hetionet (Himmelstein et al., 2017), entities are biological concepts such as ailments, compounds, and anatomical structures. The facts describe possible treatments for ailments, associations between genes, and various other relationships between the concepts. While the creation of specific KGs potentially requires guidance by domain experts, there exist several initiatives to automatically create KGs that store general human world knowledge. For instance, projects like Yago (Suchanek, Kasneci, and

Weikum, 2007), Freebase (Bollacker et al., 2008), and DBpedia (Auer et al., 2007) create KGs by extracting structured data from the Wikipedia encyclopedia, among other sources. They use combinations of various techniques from different fields such as data mining, information extraction, and natural language processing.

The Freebase project (Bollacker et al., 2008) was an extensive effort to compile various sources into structured knowledge by Metaweb Technologies and it was later continued by Google. Popular topics are famous persons, media, countries, and general locations. In 2015, Freebase was discontinued and integrated into the Wikidata (Vrandečić and Krötzsch, 2014) initiative. Wikidata is a collaborative project that relies on automated and user curated information. Alongside the facts, Wikidata also aims to store their sources and it has thousands of users worldwide contributing to the project.

Yago was introduced originally by Suchanek, Kasneci, and Weikum (2007) and has received multiple updates (Pellissier Tanon, Weikum, and Suchanek, 2020; Hoffart et al., 2013; Mahdisoltani, Biega, and Suchanek, 2015) with the most recent version 4.5 (Suchanek et al., 2024). While the earlier versions of Yago directly extracted information from Wikipedia infoboxes and derived entities from Wikipedia page titles, newer versions rely on the entities from Wikidata. The taxonomic structure of Yago4 (Pellissier Tanon, Weikum, and Suchanek, 2020) is mostly based on Schema.org whereas the newest version integrates the taxonomy of Wikipedia and Schema.org and enforces rigorous semantic constraints for ensuring consistency.

The DBpedia project (Auer et al., 2007; Lehmann et al., 2015) extracts data from more than hundred different language editions of Wikipedia. The project is crowd-sourced and automatically extracts infoboxes to extract multiple versions of knowledge bases. Infoboxes from different languages are mapped into one shared knowledge base and overall the DBpedia project extract almost 1.5 billions of facts (Lehmann et al., 2015). As for the other mentioned projects, the respective datasets and sources are publicly available.

Although the automated and semi-automated processing of millions or billions of data points naturally will induce an error rate, when applying learning approaches on KGs, some theoretical assumptions have to be made. Within this thesis, the core assumption is the axiom that a fact of a KG is true which will be discussed in more detail in the next section.

2.1.3 Assumptions About Knowledge Graphs

Let \mathcal{G} be a KG. The base assumption that we make throughout this thesis is that each fact contained in \mathcal{G} denotes a true statement. There is no uncertainty involved with the statement made by the fact, or, in other words, the likelihood for the statement to be true is one. This is arguably a strong assumption which will fail in practice. There are other branches of research that are concerned with repairing KGs (Xue and Zou, 2022). Nevertheless, in the field of completing KGs, the assumption is required to model the KG in a suitable way (Nickel et al., 2015).

This leaves the question of how to treat facts which are not contained in \mathcal{G} . Symmetrical to the interpretation of existing facts, we could simply assume that non-existing facts represent an incorrect statement. This viewpoint is also called the **closed world assumption** (CWA). For example, in the KG shown in Figure 2.1, there does not exist a fact that states that Anna speaks English and therefore we can be certain that she does not speak English.

Open World Assumption (OWA)

The set of all possible facts can be quite large. Under the CWA, the correctness of each of these facts needs to be evaluated which is quite time consuming and not feasible for most domains. Under the OWA, on the other hand, the truth value of not-existing facts is not specified. In this case, despite the absence of *speaks(anna, english)* it still is possible that Anna speaks English.

The CWA and OWA are two different paradigms that determine how KGs can be used within knowledge acquisition pipelines. While the CWA is a stronger assumption (Anna does not speak English) than the OWA (Anna might speak english), it is also less realistic. In practice, most of the existing KGs are incomplete, i.e., they do not contain all true facts of the given domain. Under the OWA, the incompleteness is respected and prevents users from drawing false conclusions.

The assumptions made about a KG also determine the tasks that need to be performed to enhance its usability. For instance, the incompleteness of KGs is a central problem of this thesis and will be discussed in more detail in Section 2.2. However, while the CWA is too strong, the OWA introduces a substantial amount of uncertainty limiting the applicability of learning paradigms.

Partial Completeness Assumption (PCA)

Under the PCA (Galárraga et al., 2013), which is also termed the partial closed world assumption (Nickel et al., 2015), it is assumed that knowing one relation property about an entity means knowing all of them. Assume that a correct fact $p(s, o) \in \mathcal{G}$ is given. Under the PCA, we assume that every remaining correct fact $p(s, o')$ is also included in the KG already. Conversely, every fact with relation p that has subject entity s and is not contained in the KG, is assumed to be false. For example, in Figure 2.1, the information is given that Lisa speaks English. Therefore, she does not speak any additional language. Anna, on the other hand, might speak any particular language, as she does not appear in the subject slot of the *speaks* relation in any fact. The assumption is likewise made with respect to the object entity of existing facts. Additionally, it will be helpful in constructing negative examples when learning models on KGs which will be discussed further in Section 2.7.

2.2 Knowledge Graph Completion

In Section 2.1.3, it was mentioned that facts from a given KG \mathcal{G} are considered to be true statements but most of the existing KGs are incomplete. We say that \mathcal{G} is incomplete, if there is a fact $p(s, o) \notin \mathcal{G}$ which is true but does not exist in the graph. Incompleteness is only viable under the OWA as the CWA does not allow for not-existing true facts.

For example, let us consider the KG given in Figure 2.1. We know that Lisa lives in London. However, the graph does not contain the information that she also lives in the UK, which must be true given our background knowledge. Likewise, there is no information about Bernd and Anna speaking any language.

In the problem of knowledge graph completion (KGC), we need to find missing facts given an incomplete KG. There exist different directions of how to address the problem. We could collect external resources, for instance, data from the web, and add facts based on processing the collected data. In fact, this procedure is often used to construct KGs in the first place and was discussed in Section 2.1.2. On the other hand, for an already given KG one might also use the existing facts to derive new

facts by finding patterns and regularities. The focus of the thesis is based on the latter and, in the evaluation protocols, no additional external data will be allowed.

Throughout the following sections and chapters, we will use the term predictive performance to denote the quality of a model to perform KGC. There exist different evaluation protocols to measure the predictive performance (Wang et al., 2018). The most intuitive approach might be to define a simple classification problem for facts which is also termed triple classification. However, a KG does not contain explicit negative facts. Although one could sample negatives, it was already mentioned that evaluating the correctness of missing facts is costly. Additionally, it can result in tasks that are generally considered to be too trivial (Safavi and Koutra, 2020). Therefore, we will focus on ranking-based evaluation instead (Bordes et al., 2013b).

2.2.1 Ranking-Based Evaluation

Ranking-based protocols are based on the idea that a model should assign high plausibility or confidence scores to facts that exist in the KG, as they are known to be true, while providing lower scores to facts that do not exist in the KG. Although the protocols are not without merits, they can be performed by exclusively using the facts from the KG and do not rely on the construction of explicit negatives. They come with an intricacy in that they are based on answering queries opposed to directly predicting the truth values of target facts. The queries are formed from known facts, as shown in the following example based on Figure 2.1.

Example 4. From a correct fact $speaks(bernd, english)$, two queries can be formed. A tail query $speaks(bernd, ?)$ and a head query given by $speaks(?, english)$. For the tail query, one true answer is $english$ and for the head query it is $bernd$.

We always assume that one head and one tail query is formed from a correct base fact as shown in the example. Although there may be multiple correct answers to each of the queries, the base fact provides one definite true answer for each direction. This answer will be termed the current true answer throughout the following, i.e., the correct answer provided from the fact which was used to form the query in the first place. In general, a query does not necessarily be based on an existing base fact. However, we will stick with this convention as it aligns with the practical procedure of the protocol that will be introduced below.

Remark 1. Further below, we will encounter the definition of a rule head. It denotes the atom on the left-hand side of a rule and we distinguish it from a head query, which simply denotes a query direction as shown in the example above.

For defining ranking-based protocols, we require the concept of a fact scoring function.

Definition 2 (Scoring Function). Given entities \mathcal{E} and relations \mathcal{P} , a scoring function ϕ calculates real-valued fact scores, i.e., $\phi : \mathcal{P} \times \mathcal{E} \times \mathcal{E} \rightarrow \mathbb{R}$.

The scoring function takes a fact as input and outputs a real-valued score. It is interpreted as a plausibility value of the correctness of the fact. When the scores are normalized to probabilities, it may represent the probability that a fact is true which will be discussed in more detail in later chapters. Throughout the thesis, different approaches for KGC will be introduced by describing their mechanism of calculating fact scores.

Queries are implicitly answered by calculating fact scores where the source entity of the query is held constant. For example, for the query $p(s, ?)$ with source entity s , scores are calculated for multiple (or all) facts $p(s, o')$ with $o' \in \mathcal{E}$.

	Tail Ranking		Head Ranking	
Rank	$speaks(bernd, ?)$		$speaks(?, english)$	
#1	german	2.3	anna	3.4
#2	french	1.9	lisa	2.1
#3	italian	1.8	bernd	1.4
#4	english	1.4	-	

TABLE 2.2: A tail and head ranking and scores with respect to queries formed from the true base fact $speaks(bernd, english)$.

Definition 3 (Candidate Score). Let \mathcal{E} denote a set of entities, let ϕ be a fact scoring function, and let $p(s, ?)$ be a query with $p \in \mathcal{P}$ and $s \in \mathcal{E}$. The candidate score of an entity $o' \in \mathcal{E}$ for the query is the score of the fact that completes the query with o' , i.e., $\phi(p(s, o'))$. For brevity, the candidate score will also be written as $\phi(o')$ where the reference to the query and fact score will be clear from the context.

Candidates can be sorted according to their scores in descending order which is termed a candidate ranking throughout the thesis. The position of a candidate in a ranking can be easily obtained by counting how many distinct other candidates achieved a higher score, which is termed the rank of the candidate.

Definition 4 (Rank). Given entities \mathcal{E} , relations \mathcal{P} , and an evaluation fact $p(s, o)$, the tail rank of o is defined as

$$rk_t(o|p, s) = |\{o' \in \mathcal{E} : \phi(p(s, o')) \geq \phi(p(s, o))\}| + 1, \quad (2.1)$$

and the head rank $rk_h(s|p, o)$ of s is defined symmetrical.

From the perspective of evaluating model quality for KGC, it is desired that the correct candidate answers achieve a low rank and respectively a top position in the ranking. Table 2.2 shows an exemplary ranking with fact scores with respect to the base fact $speaks(bernd, english)$. The current true answers $english$ (tail direction) and $bernd$ (head direction) have the ranks 4 and 3, respectively.

Remark 2. Throughout the remaining chapters, the term *top- m ranking* will be used occasionally (e.g., *top-100 ranking*). This corresponds to a ranking in which at most m candidates with the highest scores are considered. The ranking potentially can contain less than m candidates.

A common procedure is to additionally filter the rankings and remove all known true answers except of the current true answers. In regard to Table 2.2, let us assume that the fact $speaks(bernd, german)$ is known to be true. Indeed, in the tail ranking, $german$ is the top answer. However, the current correct answers is $english$ based on the base evaluation fact and it has rank four. If this is used as the final rank, the model is penalized for providing a high score to a correct answer. Therefore, from the raw rankings, all correct answers which are different from the current correct answer are filtered out. The new rank for the answer $english$ is therefore three. In the next section, we will define how the evaluation facts and the facts used for filtering are specified.

Remark 3. Throughout this thesis, when not specified otherwise, ranking-based evaluation metrics are always based on filtered rankings.

2.2.2 Protocol and Metrics

We will now describe the common protocol and subsequently introduce the used metrics formally. For the purpose of evaluation, the facts of a KG are split into disjoint training, validation, and testing subsets, e.g., $\mathcal{G}^{Train} \cap \mathcal{G}^{Valid} \cap \mathcal{G}^{Test} = \emptyset$, where each split contains entities and relations from the global sets \mathcal{E} and \mathcal{P} as shown in Definition 1.

Remark 4. *Each subset regarding the splitting is itself a KG. We will use the terms, for instance, training KG, training set or training split interchangeably throughout the thesis.*

A model or approach can learn from the training KG possibly by using guidance from the validation graph (e.g., for hyperparameter tuning). After this stage, each fact of the test KG is used as an evaluation fact. The scoring function ϕ of the model is used to calculate the respective head and tail rankings for each query formed from every evaluation fact, as described in the previous section. Each ranking is filtered with all the facts from all three splits. Based on the definitions from above, we can define the following metrics.

Definition 5. *The mean rank (MR), mean reciprocal rank (MRR), and Hits@k metrics are defined as*

$$\begin{aligned} \text{MR} &= \frac{1}{2|\mathcal{G}^{test}|} \sum_{p(s,o) \in \mathcal{G}^{Test}} \left(\text{rk}_h(s|p,o) + \text{rk}_t(o|p,s) \right), \\ \text{MRR} &= \frac{1}{2|\mathcal{G}^{test}|} \sum_{p(s,o) \in \mathcal{G}^{Test}} \left(\frac{1}{\text{rk}_h(s|p,o)} + \frac{1}{\text{rk}_t(o|p,s)} \right), \\ \text{Hits@k} &= \frac{1}{2|\mathcal{G}^{Test}|} \sum_{p(s,o) \in \mathcal{G}^{Test}} \left(\text{Ind}\{\text{rk}_h(s|p,o) \leq k\} + \text{Ind}\{\text{rk}_t(o|p,s) \leq k\} \right). \end{aligned}$$

The MR is the average of all the ranking positions of the correct answers. The MRR is closely related. It is between zero and one (higher is better) and it gives higher weights for better ranking positions. For instance, an improvement by one position has higher effect on the MRR if the improvement is from position two to one compared to an improvement from 100 to 99. The Hits@k metrics measures in how many percent of rankings the true answer was within the top-k candidates. The Hits@1 metric can also be interpreted as the multi-class accuracy. The section is concluded with an example regarding the defined metrics.

Example 5. We assume that the test KG consists of only one fact given by $\text{speaks}(\text{bernd}, \text{english})$. The rankings from Table 2.2 are taken as reference and are filtered with the facts $\{\text{speaks}(\text{bernd}, \text{german}), \text{speaks}(\text{lisa}, \text{english})\}$. This leads to rank three for correct answer english in tail direction and to a rank of two for correct answer bernd in head direction. Then we have that

$$\begin{aligned} \text{MRR} &= \frac{1}{2 \cdot 1} \left(\frac{1}{3} + \frac{1}{2} \right) = 0.42, \\ \text{MR} &= \frac{1}{2 \cdot 1} (3 + 2) = 2.5, \\ \text{Hits@1} &= 0, \\ \text{Hits@2} &= 0.5 \quad . \end{aligned}$$

2.2.3 Scientific Benchmarks for KGC

Scientific benchmark KGs are datasets that are derived from public KGs and adapted for the evaluation of the KGC task. In the following, the KG benchmarks that will be used in the experimental parts of this thesis will be introduced. The benchmark dataset were created by first collecting a base KG and subsequently splitting it into separate training, validation, and testing KGs as described above. The different splits are predefined by the authors of the respective publications that introduced a benchmark and are not changed within this thesis. Summary statistics for the benchmarks are shown in Table 2.3.

Benchmark	Entities	Relations	Number of Facts		
			Train	Valid	Test
Fb15k-237	14 505	237	272 115	17 535	20 466
WN18RR	40 559	11	86 835	3 034	3 134
Codex-M	17 050	51	185 584	10 310	10 311
Codex-L	77 951	69	551 193	30 622	30 622
Yago3-10	123 182	37	1 079 040	5 000	5 000

TABLE 2.3: KG benchmark summary statistics.

WN18 and Fb15k

The benchmarks WN18 and Fb15k were initially used to evaluate the KGE model TransE which will be introduced in Section 2.7. WN18 was proposed by Bordes et al. (2013a) and Fb15k by Bordes et al. (2013b). Although the benchmarks will not be part of the experiments of this thesis, they are important for understanding the context of the used datasets. WN18 is a subset of the KG WordNet (Miller, 1995). WordNet is also described as a lexical database. It describes English word types such as nouns, verbs, and adjectives and links them with semantic and grammatical relations (Miller, 1995). WN18 was created by taking 18 relations from WordNet and selecting all entities, connected to one of these relations, that appear in at least 15 facts. Fb15k is a subset of Freebase and it was created by selecting facts based on entities that appeared at least 100 times in Freebase and also had a Wikipedia entry.

Fb15k-237

This benchmark is a direct subset of the Fb15k dataset. Toutanova and Chen (2015) showed that a simply baseline based on binary features, which describe if two entities are connected via some relation, could achieve state-of-the-art results for Fb15k and WN18. Therefore, they proposed a more challenging subset, Fb15k-237, which was initially termed Fb15k-Selected. The dataset was created by first only selecting the most frequent relations from Fb15k. Subsequently, they searched for relation pairs that were almost identical or the inverse of each other and filtered out one relation of the pair to remove trivial patterns. Finally, they removed each fact from the validation and testing splits where the two entities already had a direct connection in the training set, i.e., appeared in any fact together.

Remark 5. *The last step of the procedure, i.e., removing all facts from the validation and testing splits where the two entities have a direct connection in the training set, will be important in the analysis of Chapter 7. In particular, it will be discussed in more detail in Section 7.4.*

WN18RR

WN18RR is direct subset of WN18 and was proposed by Dettmers et al. (2018) while introducing the KGE model ConvE. The authors motivated the construction of WN18RR with the findings of Toutanova and Chen (2015). They describe that the same procedure was used that was also used to create Fb15k-237 (Dettmers et al., 2018). However, it is important to note that the last step of the procedure described above apparently has not been executed. Entities that appear together in a fact in the training set can also appear in a fact together in the validation and test sets which is more realistic.

Yago3-10

This benchmark is introduced by Dettmers et al. (2018) and it is a subset of Yago3 (Mahdisoltani, Biega, and Suchanek, 2015). It contains mostly descriptions about popular persons such as their professions, citizenships and gender.

Codex Benchmark

The Codex benchmark (Safavi and Koutra, 2020) consists of three different evaluation KGs, Codex-S, Codex-M, and Codex-L. The authors criticise that Fb15k-237 is based on an outdated version of Freebase and propose to base a high quality benchmark on Wikidata from which they sampled an initial set of entities. This resulted in a KG with over 380 thousand entities and more than a million of facts. From this KG, they extracted different k-core subgraphs, that is, subgraphs in which each entity has at least k connections (Batagelj and Zaveršnik, 2011). They chose k to be 15, 10, and 5 for Codex-S, Codex-M, and Codex-L, respectively. We will focus on the two larger benchmarks within this thesis and examples for facts from Codex-M will be presented in Chapter 3.

2.3 Rules for Knowledge Graphs

Symbolic rules are the central building block of this thesis. KGs represent structured information by the use of facts composed of entities and relations. This inherently symbolic data representation aligns naturally with how we describe rules. We will first introduce rules generally in this section. Subsequently, in Section 2.4, we will describe how to make predictions with rules and how to use them for the task of query answering. Finally, as the concept of rule learning requires some of the previously made definitions, we will briefly discuss how rules can be learned from KGs in Section 2.6.

2.3.1 Rules

We will base the definitions of rules on logical syntax. First, we define the most basic building block of a rule, a logical atom.

Definition 6 (Atom). *An **atom** a of arity two is an expression of the form $a = p(\tau_1, \tau_2)$ where p is a binary relation and τ_1, τ_2 are terms. A term is either a constant or a variable.*

Only atoms of arity two are considered as higher values are not needed throughout the thesis. Likewise, functions as terms are not needed. Under the KG definitions, entities from \mathcal{E} are treated as logical constants and the binary relations are from \mathcal{P} .

Then, the facts of a KG describe ground atoms, i.e., atoms containing two constants. Rules are composed of atoms:

Definition 7 (Rule). Let a_0, \dots, a_n be atoms over binary relation. A **rule** is a formula of the form

$$a_0 \leftarrow \bigwedge_{i=1}^n a_i.$$

We say that a_0 is the **head** of the rule and $\bigwedge_{i=1}^n a_i$ is the **body** of the rule. The rule **length** n is the number of body atoms.

There are a few things to consider regarding the definition. First, we do not allow negated atoms in this form; the considered rules are horn rules. They can alternatively be written as a disjunction of atoms where all body atoms are negated whereas the head atom is not negated. Second, for better readability, we write the head of the rule on the left-hand side. Third, when we write out specific rules, we will chain the body atoms simply with a comma instead of the conjunction symbol as shown in the following example:

Example 6. Consider the following rules where X, Y, Z and A are variables and *london* and *english* are constants,

$$\begin{aligned} r_1: & \text{speaks}(X, Y) \leftarrow \text{lives}(X, A), \text{hasLanguage}(A, Y) \\ r_2: & \text{speaks}(X, \text{english}) \leftarrow \text{lives}(X, \text{london}) \\ r_3: & \text{contains}(X, Y) \leftarrow \text{contains}(X, A), \text{cityOf}(Y, A) \\ r_4: & \text{profession}(X, \text{actor}) \leftarrow \text{starrsIn}(X, Z). \end{aligned}$$

The first rule says that a person speaks a language if they live at a place (country) which has this language as official language. The second rule says that somebody speaks English if they live in London. The third rule is recursive and says that a place is contained in an area if that place is a city of another place that is already contained in the area. Finally, the last rule has one free variable and says that a person is an actor if they play the main role in some undefined movie.

We can observe that the variable order within the body atoms is arbitrary. For example, in the first rule, A is positioned before Y and vice versa for the third rule. However, flipping the positions of two variables within a rule also changes the meaning of the rule.

2.3.2 Rule Types

Instead of using arbitrary horn rules, we restrict the discussion in this thesis to specifically defined types. For instance, as described above, only atoms of arity two are considered for the alignment with the facts of a KG. We will impose further constraints on the syntax in the following by introducing four rule types. They correspond to the rules that can be learned by the system AnyBURL (Meilicke et al., 2024), which will be introduced in Section 2.6. We will use the rules from Example 6 to introduce the different types.

The first type that we are concerned with is given by **B**-rules:

$$\mathbf{B}\text{-rule} : \text{speaks}(X, Y) \leftarrow \text{lives}(X, A), \text{hasLanguage}(A, Y)$$

The shown rule has length two. It represents the most well-known rule type and often these rules are also termed cyclical rules. To see this, the body of the rule describes a path from X to Y via A . Combined with the head, the rule represents a cycle in the graph.

The second rule type is given by U_c -rules. They contain two constants, that is, two entities from the KG:

$$U_c\text{-rule : } \textit{speaks}(X, \textit{english}) \leftarrow \textit{lives}(X, \textit{london})$$

They always contain exactly two constants or entities even if their length is larger than one. They allow to describe more specific regularities. It is allowed that the entities are identical; in this case, the rule type represents a half-grounded B -rule.

The third rule type is given by U_d -rules. They contain a free variable in the rule body:

$$U_d\text{-rule : } \textit{profession}(X, \textit{actor}) \leftarrow \textit{starsIn}(X, Z)$$

The shown rule has length one. By the free variable in the body, these rules allow to implicitly encode types which relates to the schema discussion presented in Section 2.1.1. For instance, the body of the rule implicitly defines a type of entities that appear in movies. Another example could be $\textit{playsInstrument}(X, Z)$ to encode the type musician when Z is free.

Finally, the last rule type that can be learned by the AnyBURL system is given by U_z -rules. The type is treated as a special case and its behaviour deviates slightly from a logic interpretation since it is particularly designed for query answering. We write this rule with a head and an empty body, such as $\textit{lives}(X, \textit{germany}) \leftarrow$. The rule allows to predict the entity in its head for a query. For example, if a tail query is given by $\textit{lives}(\textit{anna}, ?)$ then the shown rule can predict the fact $\textit{lives}(\textit{anna}, \textit{germany})$. In general, the rule type can be helpful to predict the default distribution in rare cases but its benefits are mostly negligible.

Remark 6. *The types B and U_c represent the most important rules within this thesis. They alone determine most of the predictive performance of rule-based KGC. Moreover, the maximum length of the U_c -rules and U_d -rules is always set to one to express small specific patterns or type relationships. We will show an ablation study in Chapter 3, exploring the benefits of the individual rule types.*

2.3.3 Formal Rule Types

While we have introduced the rule types with practical examples in the last section, they can also be described more generally for any given rule length. Based on the previously made definitions, we will introduce the formal descriptions in the following. They are based on our study regarding large-scale rule learning (Meilicke et al., 2024). However, the examples from the previous section are sufficient for the understanding of the remaining contents of this thesis. The rule types can be described formally as follows:

$$\begin{array}{ll}
\mathbf{B} & h(A_0, A_n) \leftarrow \bigwedge_{i=1}^n b_i(A_{i-1}, A_i) \\
\mathbf{U}_d & h(A_0, e) \leftarrow \bigwedge_{i=1}^n b_i(A_{i-1}, A_i) \\
\mathbf{U}_c & h(A_0, e) \leftarrow \left(\bigwedge_{i=1}^{n-1} b_i(A_{i-1}, A_i) \right) \wedge b_n(A_{n-1}, e') \\
\mathbf{U}_z & h(A_0, e) \leftarrow
\end{array}$$

We let A_i denote variables and h and b_i are relations. The constants e and e' represent entities from a KG. As previously mentioned, we allow that $e = e'$. Likewise, recursive rules are permitted where, e.g., $h = b_i$ for some i . A recursive case is given by the third rule in Example 6. Moreover, as explained above, the order of the variables within the body is arbitrary. For the heads of the rules containing a constant, it can either be written on the left-hand or on the right-hand side of the atom.

2.3.4 Logical Formulation of Rules

Let us consider the first rule of Example 6. In first-order logic, we could write the rule as

$$\forall x, y, z : \text{lives}(x, z) \wedge \text{hasLanguage}(z, y) \rightarrow \text{speaks}(x, y). \quad (2.2)$$

The head is written on the right-hand side. Additionally, the variable quantification is explicitly described. We will assume that throughout the work, all variables are implicitly universally quantified if not specified otherwise.

In Expression 2.2, it is permitted that distinct variables will refer to the same constants. While this is commonly allowed in logic, we make a strict assumptions and force distinct variables to be unequal. This is denoted *object identity* and becomes relevant when introducing substitutions in the next section. In expression (2.2), we would need to add $x \neq y \wedge x \neq z \wedge y \neq z$ to the formula.

2.4 Inference with Rules

To utilize rules for finding missing facts in a KG, we need to perform inference with them and we will mostly focus on making direct fact predictions. However, as it will be described in the following, rules can be associated with uncertainty and therefore we rely on certain strategies to quantify the likelihood of predictions.

2.4.1 Rule Predictions

Performing inference with rules means to apply them onto some base KG to derive new facts. The core concept for applying rules is given by the substitution of variables with entities from the KG.

Definition 8 (Substitution). *Assume that we are given a domain of variables $\{A, X, Y, \dots\}$ and a set of constants or entities \mathcal{E} . A (ground) **substitution** θ is a mapping from the variables to constants, e.g., $\theta = \{A \mapsto e_1, X \mapsto e_2, Y \mapsto e_3, \dots\}$ where $e_1, e_2, e_3 \in \mathcal{E}$.*

We will only consider substitutions under strict object identity, that is, two distinct variables can never be mapped onto the same entity. Substitutions are applied to the head and to the tail of a rule resulting in ground facts.

Definition 9 (Grounding). *Let r be a rule and θ a substitution. The **head grounding** of r with respect to θ is the fact that results when applying θ to the rule head. The **body grounding** is the set of facts resulting from applying θ to the rule body.*

Example 7 (continued). Let X, Y, A be variables and *bernd*, *english* and *UK* be constants. Consider the first rule from Example 6,

$$\text{speaks}(X, Y) \leftarrow \text{lives}(X, A), \text{hasLanguage}(A, Y).$$

One possible substitution θ for the variables is given by

$$\{X \mapsto \text{bernd}, A \mapsto \text{UK}, Y \mapsto \text{english}\}.$$

Applying the substitution to the rule gives the head grounding $\text{speaks}(\text{bernd}, \text{english})$ and the body grounding $\{\text{lives}(\text{bernd}, \text{UK}), \text{hasLang}(\text{UK}, \text{english})\}$.

We can now define the main inference mechanism that will be used throughout. It is based on the one-time application of rules based on simple substitutions.

Definition 10 (One-step entailment \models_1). *Let \mathcal{R} denote a set of rules, $f = p(s, o)$ be a fact, and \mathcal{G} be a KG. The fact f is **one-step entailed** by $\mathcal{R} \cup \mathcal{G}$, written as $\mathcal{R} \cup \mathcal{G} \models_1 f$, iff there is a rule in \mathcal{R} for which a substitution exists such that the resulting body grounding is contained in \mathcal{G} and the head grounding is equal to f .*

An alternative definition to describe the one-time application of rules can be made with the immediate consequence operator (e.g., Tena Cucala, Cuenca Grau, and Motik, 2022). It takes as input a set of rules and a KG and outputs all the facts that are one-step entailed. We choose Definition 10 for ease of readability and because of the natural alignment to (general) logical entailment denoted by \models (De Raedt, 2008). In fact, in Section 2.4.3, we will describe that one-step entailment implies entailment but the reverse direction is not necessarily true.

Throughout this thesis, we will often simply say that a rule predicts a certain fact, which we can define formally based on one-step entailment.

Definition 11 (Prediction). *Let \mathcal{G} be a KG, \mathcal{R} a set of rules, and f be a fact. A rule $r \in \mathcal{R}$ **predicts** f with respect to \mathcal{G} iff $\{r\} \cup \mathcal{G} \models_1 f$.*

For brevity, we will also write $r \models_1 f$ where the reference to the KG will be clear from the context. The definition provides a precise understanding of a rule prediction. A rule predicts a fact if it individually one-step entails the fact. In other words, if the fact is an immediate consequence of the rule with respect to the KG. We will conclude with an example regarding one-step entailment and predictions.

Example 8 (continued). Consider the fact $f = \text{speaks}(\text{lisa}, \text{english})$, the three rules $\mathcal{R} = \{r_1, r_2, r_3\}$ from Example 6 and the KG in Figure 2.1. Following the previously made definitions, the first rule predicts f , the second rule predicts f , and all the subsets of \mathcal{R} except of $\{r_3\}$ one-step entail f with respect to the KG.

2.4.2 Uncertainty and Rule Confidences

To compare distinct rules and their predictions, we need some statistical metric that can be assigned to an individual rule expressing its certainty value. This goes beyond

the paradigm of logic programming where uncertainty is not expressed. Different approaches are discussed in the literature under the terms *rule measures*, *quality scores* and *confidences* (Galárraga et al., 2013; Tanon et al., 2017; Zupanc and Davis, 2018; Meilicke et al., 2019). We will start with the support of a rule.

Definition 12 (Support). *The **support** of a rule r with respect to a KG \mathcal{G} is defined as*

$$\text{support}(r) = |\{f : \{r\} \cup \mathcal{G} \models_1 f \wedge f \in \mathcal{G}\}|. \quad (2.3)$$

The support counts the number of predictions of a rule that exist already in the reference KG. We will loosely call them the *true predictions* throughout the following. Next, we will define the main measure used throughout this thesis, the (standard) rule confidence.

Definition 13 (Confidence). *The **confidence** conf of a rule r with respect to a KG \mathcal{G} is the number of true predictions divided by the number of all predictions with respect to \mathcal{G} ,*

$$\text{conf}(r) = \frac{\text{support}(r)}{|\{f : \{r\} \cup \mathcal{G} \models_1 f\}|}. \quad (2.4)$$

This is the standard definition of the confidence and it originates from association rules (Agrawal, Imieliński, and Swami, 1993). The most intuitive viewpoint describes it as the number all true predictions divided by the number of all predictions of the rule which shows the resemblance to precision. In general, the confidence serves as a quality measure for the rules. Therefore, when we define rules for a KG or when we learn them automatically, we want to focus on the rules with high confidences.

Remark 7. *Throughout the experiment sections of this thesis, the confidence will be augmented with a smoothing parameter that is a natural number. It will be added in the denominator of equation 2.4 and therefore adds artificial negative example predictions.*

There exist different variations of the confidence in the literature. For instance, Galárraga et al. (2013) proposed a formulation based on the PCA paradigm:

Definition 14 (PCA Confidence). *The **PCA confidence** of a rule r with respect to a KG \mathcal{G} is defined as*

$$\text{conf}_{\text{PCA}}(r) = \frac{\text{support}(r)}{|\{p(s, o) : \{r\} \cup \mathcal{G} \models_1 p(s, o) \wedge \exists o' : p(s, o') \in \mathcal{G}\}|}. \quad (2.5)$$

The definition is explicitly based on the PCA assumption that was introduced in Section 2.1.3. By construction, true predictions are always included in the denominator. A prediction that is not in the KG, however, is only included in the denominator, if the subject entity already appears as subject with the respective relation.

2.4.3 Rules and Reasoning

While we focus on the one-time application of rules expressed by one-step entailment, rules can also be employed to perform classical inference in the form of (general) entailment, denoted by \models . The exact definition of entailment depends on the respective underlying logical formalism. For the setting in this thesis, we can consider the definitions employed by De Raedt (2008):

Definition 15 (Entailment \models). *A set of clauses φ entails a clause α iff every model for φ is also a model for α .*

While the formal semantics of models and interpretations are beyond the scope of this thesis, the definition means that under entailment it holds that whenever φ is true or satisfied then also α must be true. Moreover, in the definitions of the previous sections, we distinguish between facts and rules. On the other hand, with respect to Definition 15, both are considered to be clauses. For instance, a fact is an atomic clause that only contains constants.

In the following, an example demonstrating the differences between one-step entailment \models_1 and entailment \models will be presented.

Example 9. Consider the following KG

$$\mathcal{G} = \left\{ \begin{array}{l} \text{lives}(\text{bernd}, \text{UK}) \\ \text{lives}(\text{anna}, \text{london}) \end{array} \right\},$$

and the following rules:

$$\begin{array}{l} r_1: \text{speaks}(X, \text{english}) \leftarrow \text{lives}(X, \text{UK}) \\ r_2: \text{lives}(X, \text{UK}) \leftarrow \text{lives}(X, \text{london}) \end{array}$$

In the viewpoint of Definition 15, the KG contains background knowledge, that is, atomic clauses that must hold true. This also aligns with our assumptions about KGs from Section 2.1.3. Now, given the definitions from Section 2.4.1, we have that $\{r_1\} \cup \mathcal{G} \models_1 \text{speaks}(\text{bernd}, \text{english})$ and also $\{r_1, r_2\} \cup \mathcal{G} \models_1 \text{speaks}(\text{bernd}, \text{english})$. But, on the other hand, we have $\{r_1, r_2\} \cup \mathcal{G} \not\models_1 \text{speaks}(\text{anna}, \text{english})$.

For general entailment, we have that $\{r_1, r_2\} \cup \mathcal{G} \models \text{speaks}(\text{bernd}, \text{english})$ because one-step entailment implies entailment. This follows directly from the respective definitions. Intuitively, the reason is that the direct prediction of r_1 , $\text{speaks}(\text{bernd}, \text{english})$, is exactly the fact that makes r_1 true (it is contained in every model for r_1). Therefore, $\text{speaks}(\text{bernd}, \text{english})$ must be true as otherwise r_1 cannot be true.

The difference between \models_1 and \models becomes apparent when we consider that $\{r_1, r_2\} \cup \mathcal{G} \models \text{speaks}(\text{anna}, \text{english})$. The statement holds because first it must hold that $\text{lives}(\text{anna}, \text{UK})$ is true. It is a direct prediction of r_2 and therefore it must hold true to satisfy r_2 . However, if $\text{lives}(\text{anna}, \text{UK})$ is true, then also $\text{speaks}(\text{anna}, \text{english})$ must be true because it is directly predicted by r_1 . Therefore, in every assignment of truth values to the facts (an interpretation) where r_1, r_2 are satisfied (the interpretation is a model), the fact $\text{speaks}(\text{anna}, \text{english})$ is true.

In the example, we used two steps for showcasing the mechanics of entailment. However, it can require an arbitrary number of steps and is therefore also termed multi-step reasoning opposed to the one-step reasoning performed by \models_1 . From a computational perspective, one-step entailment is significantly cheaper to calculate, especially when KGs are large.

2.5 Rule Confidence Aggregation

Throughout large parts of the following chapters, the goal will be to assign plausibility scores to fact predictions for ranking them in regard to the KGC evaluation protocols. Along these lines, a central topic that will be discussed is given by the rule confidence aggregation problem.

2.5.1 Problem Setting

The rule confidences defined in Section 2.4.2 can readily be assigned to fact predictions providing them with a certainty value. That is, we can define the fact scoring function, defined in Section 2.2, based on the rule confidence. For example, let f be a fact predicted by a rule r based on some KG. Now we can define $\phi(f) = \text{conf}(r)$.

However, this leaves the question unanswered of how to proceed when f was predicted by multiple distinct rules simultaneously as demonstrated in the introduction section of this thesis. One possibility is to aggregate the confidences into one hopefully meaningful prediction score, which we term *confidence aggregation*. We proceed with an example which is an adapted version of the example shown in the introduction.

Example 10. Let *locIn* denote the relation *locatedIn*. Consider the following rules,

$$\begin{aligned} r_1 [0.64]: & \text{ employerOf}(X, Y) \leftarrow \text{internAt}(Y, X) \\ r_2 [0.44]: & \text{ employerOf}(X, Y) \leftarrow \text{locIn}(X, B), \text{locIn}(B, A), \text{studentAt}(Y, B), \\ r_3 [0.41]: & \text{ employerOf}(X, Y) \leftarrow \text{cooperatesWith}(X, A), \text{studentAt}(X, A) \end{aligned}$$

The numbers in brackets are the rule confidences. The first and third rule are quite intuitive. The second rule expresses that a person might work for a company if that company is located at the same place where this person went to university. Now assume that all three rules predict Anna to work for Google. And the first two rules predict Lisa to work for Google. Is Anna more likely to work for Google than Lisa? The rule confidence aggregation problem aims to find a final value for the two predictions. It should be meaningful in the sense that it allows to rank the predictions according their plausibility or likelihood.

The number of rules that predict a candidate fact simultaneously can be large and there can be redundancies between the predicting rules. For example, if the second rule from Example 10 predicts Anna to work for Google already, the question arises whether the third rule provides additional evidence for this prediction. The rules make the prediction for seemingly similar reasons, as it is more likely for a university and a company to cooperate when they are located in the same location.

2.5.2 Aggregation Functions

Let \mathcal{R} be a set of rules and \mathcal{G} be a KG. Throughout the remainder of this thesis, we let $\mathcal{R}_f(\mathcal{G}) \subseteq \mathcal{R}$ denote the subset of rules that predict f with respect to \mathcal{G} , i.e., $\mathcal{R}_f(\mathcal{G}) = \{r \in \mathcal{R} : \{r\} \cup \mathcal{G} \models_1 f\}$. For brevity, we will write this often short as \mathcal{R}_f . The reference to \mathcal{G} will always be clear from the context. We will now define the two most common aggregation strategies.

Definition 16 (Max-aggregation). *Let f be a fact, \mathcal{R} a set of rules, \mathcal{G} a KG and $\mathcal{R}_f \subseteq \mathcal{R}$ as defined above. The Max-aggregation score ϕ_M is calculated according to the rule with the highest confidence,*

$$\phi_M(f) = \max\{\text{conf}(r) \mid r \in \mathcal{R}_f\}. \quad (2.6)$$

Max-aggregation is robust against large sets of rules since all rules are ignored except of the rule with the highest confidence. From a fuzzy-logic perspective, it is based on a T-norm. Nevertheless, the discussions within this thesis closer resemble distributional semantics opposed to fuzzy semantics. This will also be discussed in Chapter 5.

The aggregation function was adapted by Meilicke et al. (2019) for the particular purpose of the KGC task. When two candidates have the same score calculated by Max-aggregation, a tie handling strategy has to be employed to assign ranking positions to the candidates. The authors use a heuristic that will be termed Max+ aggregation in the remainder of this thesis. If a tie is discovered, the candidates will be compared based on their predicting rules with the second highest confidence. If the tie is not resolved, the process is continued with the next rules, i.e., the aggregation strategy performs a lexicographic ordering of the candidates according to the confidences of their predicting rules.

Definition 17 (Max+ Ranking). *A Max+ ranking orders candidate predictions lexicographically with respect to the confidences of their predicting rules.*

Only if two candidates are predicted by identical sets of rules, a real tie is discovered. The strategy is quite similar to Max-aggregation as the rule with the highest confidence has still the largest influence.

Remark 8. *In the literature, the term Max-aggregation is commonly used to describe the Max+ ranking strategy. In this thesis, we will distinguish the two approaches where Max-aggregation relies on random tie handling. That is, if two candidates are tied, they will be randomly ordered in the ranking.*

The Noisy-or aggregation function, on the other hand, takes into account all predicting rules. The score increases in the number of predicting rules, even if their confidences are low.

Definition 18 (Noisy-or aggregation). *Let f be a fact, \mathcal{R} a set of rules, \mathcal{G} a KG and $\mathcal{R}_f \subseteq \mathcal{R}$ as defined above. The Noisy-or score ϕ_{NO} is calculated as the Noisy-or product over all predicting rules,*

$$\phi_{NO}(f) = 1 - \prod_{r \in \mathcal{R}_f} (1 - \text{conf}(r)). \quad (2.7)$$

Max-aggregation and Noisy-or aggregation were first discussed in the context of KGC by Galárraga et al. (2015) under the term *joint prediction*. The two functions are based on assumptions which lie on opposite ends of a spectrum. Intuitively, this can be seen by the fact that Noisy-or takes into account all the rules whereas Max-aggregation only takes into account one rule. This will be discussed and formalized in detail in Chapter 5 where we will also introduce an alternative aggregation function which lies in between the two strategies.

Example 11 (continued). Let us assume that Anna is predicted to work for Google by all rules from Example 10 while Lisa is predicted by only the second and third rule to work for Google. The Max-aggregation and Noisy-or scores for Anna are 0.64 and 0.88, respectively. For Lisa, they are 0.44 and 0.67.

Remark 9. *The particular choice of rule measure for conf is independent of the aggregation function.*

2.5.3 Theoretical Considerations

The confidence aggregation problem is closely related to fields such as probabilistic reasoning and reasoning under uncertainty. For instance, the Noisy-or product originates from Bayesian Networks where it is used to express independent causes (Pearl,

1988). Likewise, it is commonly used in the context of probabilistic logic programming frameworks (Taisuke, 1995; De Raedt, Kimmig, and Toivonen, 2007; Riguzzi et al., 2017). We will briefly highlight some of these connections in the following.

Probabilistic Treatment

The connection to probabilistic reasoning is most intuitively exemplified when considering the equation given by the Noisy-or product presented in Definition 18. For instance, we could substitute the confidences in the equation with Bernoulli probabilities. Let π_j be the probability that some binary random variable j is true. Then the expression $1 - \prod_i (1 - \pi_i)$ has a well-defined probabilistic interpretation. It calculates one minus the probability that all the variables are false. This is equivalent to the probability that at least one variable is true.

While this probabilistic interpretation is insightful, it raises several important questions. How to incorporate the rule application into the probabilistic treatment? Additionally, in contrast to the Noisy-or product, Max-aggregation lacks a probabilistic interpretation and appears to be a computational heuristic. Can it nevertheless be described within a probabilistic framework? These questions will be discussed in more detail in Chapter 5 of this thesis.

Classical Reasoning and Probabilities

There exist many frameworks that extend classical logical reasoning and entailment (compare Section 2.4.3) with uncertainty (Kersting and De Raedt, 2001; De Raedt, Kimmig, and Toivonen, 2007; Richardson and Domingos, 2006; Riguzzi, 2016). Within these frameworks, the logical structures are labelled with individual probabilities. Then, they allow to calculate final target probabilities for entailed facts. Frameworks such as ProbLog (De Raedt, Kimmig, and Toivonen, 2007) and Markov Logic Networks (MLNs) (Richardson and Domingos, 2006) provide inference mechanisms that could potentially be employed directly when using rule confidences for the individual rule probabilities. Clearly, some form of aggregation has to be performed by these frameworks to calculate the final target probabilities. Therefore, in Chapter 5, we will have a close look at these mechanisms and discuss if they would provide benefits for our settings. As the details of the approaches are specific to only that chapter, we will postpone the detailed introduction of these approaches until Section 5.3.

2.5.4 Practical Considerations

Independent of the theoretical considerations, when defining a confidence aggregation function, there exist practical challenges that have to be taken into account.

Number of Learned Rules

While we will describe rule learning in the next section, for the benchmark KGs considered in this thesis, the number of learned rules can be large. In most cases, the sets of learned rules contain more rules than existing facts in the training KG. For instance, in our standard setting, on the Fb15k-237 benchmark, the number of learned rules is higher than five millions.

Varying Numbers of Predicting Rules

Given the overall large amount of learned rules, some target facts f can be predicted by hundreds of rules. In other cases, a target fact might be predicted by only one or a few rules. This leads to the question of how to compare these cases, for example, how to compare a candidate that is predicted by hundreds of rules with low confidences with a candidate that is predicted by only one rule with a high confidence.

Rule Redundancies

Many of the mined rules are not independent and make candidate predictions for similar reasons. We have seen cases in the introductory chapter and in Example 10. If one fact is predicted by multiple rules that express mostly similar patterns, an aggregation function should be able to take this into account to not overestimate the final prediction score.

Max-aggregation and Noisy-or aggregation provide different solutions to these challenges. The focus on the rule with the highest confidence allows Max-aggregation to be robust against a large number of rules and their potential redundancies. Noisy-or, on the other hand, is able to take into account different numbers of predicting rules. Nevertheless, both aggregation functions are not optimal which will be shown formally in Chapter 5. Subsequently, in Chapter 6, we propose to learn the aggregation functions directly from the data instead of relying on the pre-computed confidences.

2.6 Rule Learning

Rules can be obtained by learning them inductively from a given KG and the challenge is to find good rules effectively. As the number of entities and relations from a KG is finite, when assuming a maximal rule length, the space of all possible rules can easily be determined. The difficult task is to traverse this space such that rules representing strong regularities can be found. The experimental parts of this thesis focus on the rule learner AnyBURL (Meilicke et al., 2024). It is the successor of RuleN (Meilicke et al., 2018) which was built as a symbolic baseline and comparison approach against latent-based KGC.

2.6.1 AnyBURL

The rule learning approach of AnyBURL is based on a bottom-up algorithm that starts with specific expressions without variables and generalizes them into rules. The algorithm performs multiple steps repeatedly and stops after a previously defined runtime is reached. One main characteristic of the system is that any specified runtime will lead to a high quality set of rules as a scheduling mechanism allocates resources effectively, such that a wide range of rules can be learned. The algorithm repeatedly samples paths from the KG and converts them into bottom rules, generalizes them into rule types, and estimates their confidences.

Path Sampling

The path sampling step results in a bottom rule, that is, a rule that only consists of entities instead of variables. The procedure starts with sampling a fact which will be the head of the bottom rule, i.e., it will determine the head relation. Subsequently, a

random walk takes a number of specified steps over the graph starting at either of the entities of the head fact. Steps can be taken along the relation direction or in the inverse direction. After the specified number of steps is reached, the resulting path is converted into a bottom rule, e.g.,

$$speaks(anna, english) \leftarrow lives(anna, UK), hasLanguage(UK, english).$$

For this example, the created bottom rule is cyclic. Nevertheless, most of the bottom rules will not result into such a pattern and are acyclic. Therefore, there exists an additional setting in which a heuristic approach is used for the extraction of cyclic paths. In the last step of the random walk, if it is possible to complete a cycle with a certain step, it will be chosen deterministically.

Bottom Rule Generalization

In a bottom-up fashion, the created bottom rule will be generalized by repeatedly either 1) deleting an atom from the body or 2) replacing an entity with a variable. Conceptually this can be performed with a generalization lattice where each child node represents the resulting expression after performing one of the two operations. Not all child node expressions will result in useful rules. Meilicke et al. (2019) argue that the beneficial expressions are the rule types \mathbf{B} , \mathbf{U}_d , \mathbf{U}_c as described in Section 2.3.2. For example, from the bottom rule shown above, performing operation 2) three times in succession will result in the binary rule that was shown already in Example 6:

$$speaks(X, Y) \leftarrow lives(X, A), hasLanguage(A, Y).$$

On the other hand, dropping the last body atom and subsequently substituting Anna with a variable results in the \mathbf{U}_c rule $speaks(X, english) \leftarrow lives(X, UK)$.

Confidence Estimation

The two steps explained above will result in a large number of created rule candidates and not all of them represent strong regularities. The core idea is to select rules that have a confidence higher than a specified threshold where AnyBURL operations are always based on the standard confidence definition (compare Section 2.4.2).

Especially for longer rules, calculating the confidences can induce a bottleneck in the learning process as it requires a depth-first search (DFS) over the body atoms of the rules. An alternative is to stop the DFS after a fixed number of body groundings has been calculated and use only this subset for the confidence calculation. However, this may lead to a bias of the estimated confidence. Some entities can have thousands of connections, leading to the fact that only specific parts of the search tree would be visited - particularly those parts that branch off these hub entities. Therefore, whenever a new branch is visited in the search tree, the sampling algorithm selects one path randomly instead of following the DFS paradigm.

Scheduling

The learning algorithm assigns tasks to a specific number of worker threads. Each worker performs all the three steps described above independently and stores the discovered rules into a global index. Rules that are re-discovered are discarded. Each worker is assigned with a specific profile for sampling paths. A path profile is

characterised by the step length of the random walk and the path type which can be cyclic or acyclic.

All workers learn rules based on their assigned profile for a short timespan (e.g. a few seconds). After the timespan ends, a scheduling mechanism assigns a new path profile to each worker. Earlier versions of AnyBURL select the assigned profiles by a saturation based approach. If most of the discovered rules already exist the step length of the path profiles can be increased (Meilicke et al., 2019). The current version of AnyBURL defines a bandit problem in which a reward is calculated for every profile that was used in the previous time span. The assignment of new profiles is based on a policy that mixes between assigning profiles randomly and distributing them proportionally over all workers according to their achieved rewards.

The overall learning process of AnyBURL can be configured by a substantial amount of learning parameters such as the overall learning time and the maximal rule lengths of each individual rule type. A main strength of AnyBURL, however, is that it has been shown to work well over many benchmarks by using the same default parameters (Meilicke et al., 2024).

2.6.2 AMIE

AMIE (Galárraga et al., 2013) is a well-known rule mining system and it was updated to AMIE+ (Galárraga et al., 2015) and AMIE3 (Lajus, Galárraga, and Suchanek, 2020).

The base algorithm AMIE uses a breadth-first search paradigm (BFS) to traverse the search space of all possible rules. It starts with initializing all possible rules that have no body. During the BFS, rules are modified by a refinement operator. This operator creates new rules by adding every possible atom. Based on certain user-defined selection criteria, refined rules will be written to an output list. While this strategy in general allows to traverse the whole search space, a pruning mechanism is used. First, it ensures that rules that are not closed are not added to the output list. A rule is closed if every variable appears at least two times (Galárraga et al., 2013). Note that the rule type U_d defined in Section 2.3.2 is not closed. Second, rules are discarded that do not have a higher PCA confidence than the previously mined rules with a subset of body atoms.

In comparison to the rule types defined in Section 2.3.2, AMIE can learn **B** and U_c -rules. Moreover, it also learns rule types that cannot be learned by AnyBURL. One example is given by

$$\text{citizenOf}(X, Y) \leftarrow \text{hasChild}(A, X), \text{lives}(X, Y), \text{livesIn}(A, Y).$$

In the body of the shown rule, each of the variables X and Y appears twice. In other words, the body describes two separate paths from X to Y . A direct path via the *lives* relation and a path that goes through A . While this rule does not adhere to any of the rule types defined in Section 2.3.2, it can be also expressed by two separate **B**-Rules:

$$\begin{aligned} \text{citizenOf}(X, Y) &\leftarrow \text{lives}(X, Y), \\ \text{citizenOf}(X, Y) &\leftarrow \text{hasChild}(A, X), \text{livesIn}(A, Y). \end{aligned}$$

2.6.3 Other Rule Learning Systems

A well known and closely related field in the context of learning logical structures from examples is given by inductive logic programming (ILP) (Muggleton, 1991; Muggleton and De Raedt, 1994). Specifically, the task is to mine a logic program given a set of positive and negative examples. The examples are, in the context of this thesis, ground facts where predicates can have an arbitrary arity. The obtained program should entail each positive example but should not entail any of negative examples. Popular systems are given by, for instance, FOIL (Quinlan, 1990) or Aleph (Srinivasan, 2000). Nevertheless, for learning from KGs, negative examples cannot be provided easily and ILP approaches are known to be computationally expensive due to their expressiveness (Blockeel et al., 1999).

There also exist other rule mining approaches that are directly applied to KGs (Fan et al., 2022; Ortona, Meduri, and Papotti, 2018; Pirrò, 2020). For instance, RuDiK (Ortona, Meduri, and Papotti, 2018) focuses on learning positive and negative rules. The learning of negative rules is based on a set of negative facts, similar to the discussion above. The negative facts are created from the original KG. Precisely, they employ the PCA paradigm (compare Section 2.1.3) to construct the negatives. However, the final rule quality is evaluated on rules individually without evaluating the joint predictive performance. For instance, the authors do not discuss the (confidence) aggregation setting when multiple positives or a positive and a negative rule make the same prediction. The rule learner TyRule (Wu et al., 2022) allows for unary predicates (types) within the rule bodies to represent type constraints within the rules.

An alternative branch of work focuses on differentiable rule learning (Rocktäschel and Riedel, 2017; Yang, Yang, and Cohen, 2017; Sadeghian et al., 2019; Minervini et al., 2020). These approaches learn parameterized rules by optimizing a training objective that is based on fact scores similar to KGE training. While these works are not the focus of this thesis, they have either not shown to scale to the datasets used within the work, or the respective results will be included in the experimental sections.

2.6.4 Rule-Based KGC: A Summary

We can now put all the pieces together to perform rule-based KGC. In Algorithm 1, we show the two stages of the procedure based on calculating predictions for a head query and using Max-aggregation. The algorithm uses the symbols introduced in the previous sections.

In the learning stage, rules are learned on the training KG, and their confidences are estimated on the training KG. In the application stage, rules are applied to make predictions for the queries formed from the evaluation facts of the test KG. For each query, the candidate predictions (i.e., facts that complete the query) are calculated, and for each prediction, the set of predicting rules is stored. Based on the predicting rules, the scores of the candidates can be calculated by a confidence aggregation function. It is important to emphasize that the rules are applied with respect to the training KG to calculate the predictions for the queries based on the test KG. Thus, the test KG is not used as evidence when applying the rules.

The algorithm shows a conceptual simplification of the process. For instance, in practice, confidences might be estimated during the rule learning stage. Moreover, the computation of the candidate predictions and the storing of their predicting rules are done simultaneously.

Algorithm 1 Rule-Based KGC (head direction)

Input: Training KG \mathcal{G}^{Train} , Test KG \mathcal{G}^{Test} , entities \mathcal{E} , relations \mathcal{P}

- 1: **procedure** LEARNING STAGE
- 2: $\mathcal{R} \leftarrow \text{learnRules}(\mathcal{G}^{Train})$
- 3: *// store rule confidences w.r.t. training KG*
- 4: **for each** $r \in \mathcal{R}$ **do**
- 5: $conf \leftarrow \text{calculateConf}(r, \mathcal{G}^{Train})$ *// equation 2.4*
- 6: **procedure** APPLICATION STAGE
- 7: **for each** $f = p(s, o) \in \mathcal{G}^{Test}$ **do**
- 8: $q_{head} \leftarrow p(?, o)$
- 9: $Cands \leftarrow \{p(s', o) \mid \mathcal{G}^{Train} \cup \mathcal{R} \models_1 p(s', o)\}$ *// calculate candidate answers*
- 10: **for each** $f' \in Cands$ **do**
- 11: $\mathcal{R}_{f'} \leftarrow \text{calculate predicting rules}$
- 12: $\phi(f') \leftarrow \text{Max-aggregation}(\mathcal{R}_{f'})$ *// calculate score, e.g., Def 18 or 16*
- 13: *// sort scores to obtain query ranking*
- 14: *// repeat for the tail direction*

2.7 Knowledge Graph Embedding Models

Knowledge graph embedding (KGE) models are a popular method for completing KGs. They embed the entities and relations from a KG in a low dimensional vector space. Different KGE models are characterised by their particular scoring function that takes as input the embeddings and outputs real-valued fact scores.

2.7.1 Scoring Functions

In the following, we will introduce some of the most common KGE models. While there exists a plethora of model variations and extensions in the literature, it has been shown that predictive performance is similar when models are trained under a joint codebase and training protocol (Ruffinelli, Broscheit, and Gemulla, 2020).

Let $s, o \in \mathcal{E}$ be entities and $p \in \mathcal{P}$ be a relation. At the core of KGE models is the embedding of the entities and relations into a vector space. We will denote vectors with bold letters. To that end, we define the entity embeddings as $\mathbf{s} \in \mathbb{R}^d$ and $\mathbf{o} \in \mathbb{R}^d$ where d is the vector dimensionality. The relation embedding $\mathbf{p} \in \mathbb{R}^d$ is defined likewise. It can also be represented as a matrix, i.e., we let $M^p \in \mathbb{R}^{d \times d}$ denote a matrix embedding for p . In general, embeddings are interpreted as latent features that hopefully, after they are learned, describe meaningful characteristics of the respective entities or relations. For instance, the semantic similarity of two embeddings representing politicians, measured by the cosine similarity, should be higher than the similarity of a politician and a non-related entity such as a building.

In the following, we assume that we are given a fact $f = p(s, o)$, and we will introduce the respective scoring functions $\phi(f)$ for selected models.

RESCAL

The scoring function ϕ_{RES} for the RESCAL model (Nickel, Tresp, and Kriegel, 2011) is defined as

$$\phi_{RES}(f) = \mathbf{s}^T M^p \mathbf{o} = \sum_{i=1}^d \sum_{j=1}^d s_i M_{ij}^p o_j, \quad (2.8)$$

where \mathbf{s}^T denotes the vector transpose and, for example, $s_i \in \mathbb{R}$ is the i -th entry of the vector and $M_{ij}^p \in \mathbb{R}$ is the matrix at position (i, j) . The scoring function can be interpreted as summing all elements of the outer product of the entities and weighting each individual product with the respective entry of M^p .

DistMult

For the DistMult model (Yang et al., 2015), the scoring function has already been introduced in the introduction. Compared to RESCAL, the relation embeddings are simplified to a matrix where all off-diagonal elements are zero. Equivalently, instead of using matrix embeddings, the relations are represented with a single vector like the entities. The scoring function then boils down to the simple dot product of three vectors,

$$\phi_{DIST}(f) = \mathbf{s}^T \mathbf{I} \mathbf{p} \mathbf{o} = \sum_{i=1}^d s_i p_i o_i, \quad (2.9)$$

where $\mathbf{I} \in \mathbb{R}^{d \times d}$ is the identity matrix. The simplification regarding the relation embedding leads to a smaller number of parameters (embedding entries) and a smaller computational cost for computing fact scores.

TransE

TransE (Bordes et al., 2013b) defines the scoring function as the negative vector distance between the subject embedding, translated with the relation, and the object embeddings.

$$\phi_{TRA}(f) = -\|\mathbf{s} + \mathbf{p} - \mathbf{o}\|, \quad (2.10)$$

where we use vector addition and subtraction and $\|\cdot\|$ denotes the euclidean norm. Intuitively, for facts that are likely true, the subject embedding should be translated by the relation embedding to a vector that is very close to the object embedding. The fact is most plausible (the score is maximal) when a distance of zero is reached, i.e., when $\mathbf{s} + \mathbf{p} = \mathbf{o}$.

Other KGE Models

The introduced scoring functions can be seen as seminal formulations of a large class of KGE models. As mentioned above, many variations and augmentations exist in the literature (e.g., Rossi et al., 2021). Two well known models that are relevant for this thesis are ComplEx (Trouillon et al., 2016) and ConvE (Dettmers et al., 2018). The

ComplEx scoring formulation augments DistMult by representing the embedding vectors with complex numbers. DistMult and RESCAL provide symmetrical fact scores, i.e., the score of $p(s, o)$ is the same as the score for $p(o, s)$. However, this is an undesired property as the score, for example, for a fact $profession(obama, politician)$ should not be the same as $profession(politician, obama)$. Using complex vectors implies that fact scores are not symmetrical.

ConvE, on the other hand, first performs a 2D convolution between the concatenated entity and relation embedding. Subsequently, the output is projected into a new vector and the inner product with the object embedding is taken. While the approach is slightly more complicated, it achieves a similar predictive performance compared to other model types (Ruffinelli, Broscheit, and Gemulla, 2020).

Finally, in the experimental parts of this thesis, the HittER no-context model will be included (Chen et al., 2021) which is based on a vanilla BERT encoder architecture (Devlin et al., 2019). The model calculates directed fact scores, that is, a fact score for a tail query is different than the fact score for a head query. For a tail query, the fact score is calculated by feeding subject and relation embeddings together with the embedding of a classifier token into the BERT encoder while also adding for each of the three inputs one specific type embedding. The (directed) fact score is calculated by the dot product of the output embedding of the classifier token and the respective raw embedding of the candidate tail entity. This no-context version of the model is included within the class of KGE models because the fact score computation is exclusively based on the input embeddings opposed to using additional graph context.

2.7.2 Negative Examples

In Section 2.2, it was defined how KGC can be performed with fact scoring in regard to queries formed from evaluation facts. With respect to KGE models, we need to find suitable values for the embedding vectors such that the models will predict high scores for correct evaluation facts and low scores for incorrect or not-existing facts. The embeddings are learned by training the models on a labeled dataset consisting of correct facts and incorrect facts. However, as discussed previously, KGs do only contain correct facts. Therefore, negative facts have to be created artificially. In the following, we will describe the three most common strategies to create negative examples, which are closely related to the assumptions made about KGs that have been discussed in Section 2.1.3. We assume a given KG \mathcal{G} with entities \mathcal{E} and relations \mathcal{P} for the discussion.

Negative Sampling

A correct target fact $p(s, o)$ is perturbed to create negatives. In particular, entities s' and o' are sampled from \mathcal{E} to create the pseudo negatives $p(s', o)$ and $p(s, o')$. The number of sampled entities can be higher; commonly it is treated as a training hyperparameter. If the sampled facts exist in \mathcal{G} , they may be filtered out to prevent labeling them erroneously as negatives. Then, negative sampling is based on the PCA paradigm: Knowing $p(s, o)$ means that every non-existing $p(s, o')$ must be false (the same assumption is made for the head direction of the fact). As only some negative facts are sampled, the underlying assumption is also termed *stochastic PCA* (Ali et al., 2021). However, due to the computational cost, often the filtering step is omitted.

1vsAll Strategy

For a correct fact $p(s, o)$, all possible facts $\{p(s', o) : s' \in \mathcal{E} \wedge s' \neq s\}$ and all facts $\{p(s, o') : o' \in \mathcal{E} \wedge o' \neq o\}$ are taken as negatives without further consideration. One can also view this procedure as describing a multi-class setting. For example, for the tail direction, o is the correct class label for query $p(s, ?)$ while the number of possible classes is given by $|\mathcal{E}|$.

KvsAll

Similar to the 1vsAll strategy, based on the correct fact $p(s, o)$, the two sets $\{p(s', o) : s' \in \mathcal{E} \wedge s' \neq s\}$ and $\{p(s, o') : o' \in \mathcal{E} \wedge o' \neq o\}$ are considered. However, in this case, only facts are selected as negatives that truly do not exist in \mathcal{G} . As for negative sampling with filtering, this is based on the PCA paradigm.

2.7.3 Training

Based on the selected strategy of how to handle negative examples, model training is performed by minimizing a loss with respect to the model parameters, which are given by all entity and relation embeddings. The overall objective is the sum over all individual example losses based on a selected loss function.

The precise formulation of the objective depends on the handling of negatives, the selected loss function, and further implementation aspects. We will therefore show a simplified formulation that is based on negative sampling and the binary cross-entropy (BCE) loss. Let \mathcal{G}^+ denote positive facts, given by the original KG, and \mathcal{G}^- negative facts obtained by negative sampling. Let Ω denote all model parameters, i.e., all embeddings. The overall objective is given by

$$Loss(\Omega) = \sum_{f \in \mathcal{G}^+ \cup \mathcal{G}^-} l(f, y_f, \Omega), \quad (2.11)$$

where l is a loss function and $y_f \in \{0, 1\}$ is the label of fact f . It is defined as $y_f = 1$ if $f \in \mathcal{G}^+$ and $y_f = 0$ if $f \in \mathcal{G}^-$. Choosing the BCE loss results in

$$l(f, y_f, \Omega) = y_f \log \sigma(\phi(f)) + (1 - y_f) \log(1 - \sigma(\phi(f))). \quad (2.12)$$

In equation 2.12, σ denotes the Sigmoid function that normalizes the fact scores $\phi(f)$ into the range $[0, 1]$.

Other loss functions have also been proposed in the literature. For example, the original RESCAL model was trained on the squared error between the fact label and the score (Nickel, Tresp, and Kriegel, 2011) and for TransE, margin-ranking and a hinge loss was used (Bordes et al., 2013b). Moreover, Kadlec, Bajgar, and Kleindienst (2017) proposed to use the cross-entropy (CE) loss and it has been shown that this loss function leads to a strong predictive performance for all models classes (Ruffinelli, Broscheit, and Gemulla, 2020).

2.7.4 Variations and Hyperparameters

Equation 2.11 is a simplified formulation of one particular loss function. In practical terms, the loss minimization is performed by batch-wise gradient-descent and usually positive and negative example are structurally tied together in these batches. That is,

when $f = p(s, o)$ appears in a batch, then also the perturbed negatives based on f are commonly within the same batch. Moreover, often parameter updates are made separately for batches based on tail queries and for batches based on head queries.

Differences are also induced when different loss functions are used. For example, the CE loss under the 1vsAll strategy calculates simply the Softmax loss based on every correct fact of a batch. That is, for the tail direction, the individual fact loss is

$$l(f) = -\log \frac{\exp \{ \phi(p(s, o)) \}}{\sum_{o'} \exp \{ \phi(p(s, o')) \}} \quad (2.13)$$

The loss decreases when the normalized score of the correct current fact increases. The smallest possible loss of zero is reached when the fraction becomes 1, i.e., all mass is centered on the current fact.

Moreover, there exist various hyperparameters when training KGE models. Some of these are model specific such as the filter size of the convolutions of ConvE and others are relevant for all models such as the learning rate, embedding dimensionality, or the type of parameter regularization used. The parameter values are determined from the validation sets by performing hyperparameter optimization techniques (Yu and Zhu, 2020). Within the thesis, we will mostly report results of pre-trained models under optimal hyperparameters based on the work by Ruffinelli, Broscheit, and Gemulla (2020). The models are available from the libKGE library (Broscheit et al., 2020).

2.8 Graph Neural Networks

In Chapter 7 of this thesis, we will discuss and analyse the predictive performance of an alternative approach for KGC that is based on graph neural networks (GNNs) (e.g., Bacciu et al., 2020). This model class operates directly on graphs by the communication of neighbouring nodes. While KGE models calculate fact scores exclusively based on the learned embeddings, GNNs take into account the graph context explicitly. Originally, they work on standard graphs composed of nodes and edges where the edges do not contain semantic information such as the different relations of a KG. The computations are distributed along different timesteps or iterations in which hidden feature representations are calculated by aggregating neighbouring representations. Common applications are given by, for instance, node classification (Shchur et al., 2018; Barceló et al., 2020) or, when the whole graph is aggregated into one feature representation, graph classification (Errica et al., 2020).

For each node in the graph, a GNN calculates and updates a hidden representation for L iterations or message passing rounds. Often, the iterations are termed the layers of the GNN because the computations are parameterized independently for every step. The hidden representation $z_i \in \mathbb{R}^d$ for a node i is a feature vector with dimensionality d . GNNs are characterised by the computation that updates the hidden representation of the node in layer l . In the most general form, the update can be described by three distinct functions *Aggregate*, *Message*, and *Combine* that can each be parameterized separately in every layer. The hidden representation for node i in layer l is calculated as:

$$z_i^{(l)} = \text{Combine} \left(z_i^{(l-1)}, \text{Aggregate} \left(\{ \text{Message}(z_j^{(l-1)}) \mid j \in \mathcal{N}(i) \} \right) \right) \quad (2.14)$$

Where $\mathcal{N}(i)$ denotes all the neighbouring nodes of node i . Each neighbouring node sends a message to node i . After all messages are aggregated into one vector, they are combined with the previous representation of node i to calculate the updated representation. Simple function choices could be, for instance, a feed-forward layer for Message, the sum or the mean for Aggregate, and the addition of $z_i^{(l-1)}$ with aggregated messages followed by a ReLU activation (Agarap, 2018) for Combine.

2.8.1 Relational Message Passing

The graph convolutional neural network (GCN) is a special case of equation 2.14 and parameterizes the message function according to the normalized graph laplacian (Kipf and Welling, 2016). A popular extensions of GCNs to the relational case is given by RCGN (Schlichtkrull et al., 2018) for which we show the update function of the hidden representations in the following. As before let $p \in \mathcal{P}$ be a relation and the nodes of the graph are the entities of the KG. Let $W_p^{(l)}$ be a learnable weight matrix for relation p in layer l , and let $W_0^{(l)}$ be a learnable weight matrix. The hidden representation for node (entity) i is calculated as:

$$z_i^{(l)} = \sigma \left(\sum_{p' \in \mathcal{P}} \sum_{j \in \mathcal{N}_{p'}(i)} \frac{1}{\gamma_{i,p'}} W_{p'}^{(l)} z_j^{(l-1)} + W_0^{(l)} z_i^{(l-1)} \right) \quad (2.15)$$

The factor $\gamma_{i,p}$ is described as a problem specific scaling constant that can either be learned or chosen in advance (Schlichtkrull et al., 2018). Furthermore, σ is the Sigmoid function and $\mathcal{N}_p(i)$ denotes all nodes that are connected with node i by relation p . The equation is a concrete implementation of the general form given by equation (2.14) while performing relation specific message passing. The Aggregate function is the sum, the Message function is the scaling and the transformation with the relation specific weight matrix, and the Combine function adds $W_0^{(l)} z_i^{(l-1)}$ onto the aggregated messages and applies the Sigmoid function subsequently.

In the first layer of message passing ($l=1$) each node is initialized with a learned feature vector, which resembles the entity embeddings of KGE models. For calculating fact scores, Schlichtkrull et al. (2018) employ the DistMult scoring function based on the hidden representations of the nodes (entities) $z_i^{(L)}$ in the last message passing layer and separately learned relation embeddings.

2.8.2 Query-Dependent Message Passing

While RCGN allows for relation specific message passing, the messages themselves are constant with respect to different fact scores. In other words, the message that is send from one entity to another, although being parameterized by the connecting relation, is always the same independent of the fact that is currently scored.

In Chapter 7 of this thesis, we will focus on a different GNN architecture given by the model NBFNet (Zhu et al., 2021). Conceptionally, the most important difference between this architecture and RCGN is given by the fact that message passing is query dependent. That means, the message a node sends to another node depends directly on the current fact score to be calculated based on either a head or a current tail query. Query dependent message passing is induced by two main characteristics, 1) a boundary condition at the first round of message passing and 2) query dependent

Message functions. While there exist various implementations differences, we will focus on these two characteristics in the following.

Boundary Condition

Let $p(s, o)$ be a target fact to be scored based on a tail query. As in equation 2.14, NBFNet calculates the final hidden representation $z_i^{(l)}$ of an entity based on message passing in L layers. However, in the first round of message passing ($l=1$), NBFNet initializes the hidden representations of all entities with a zero vector except of the representation of the source entity s . The representation of s is initialized with a learned embedding \mathbf{p} of the target relation p . Therefore, in the message passing layers for $l > 1$, the graph is populated successively starting from the source entity of the query in dependence of the relation of the target fact.

Query-Dependent Message Functions

Let $p(s, o)$ be the current target fact to be scored based on the tail query $p(s, ?)$. Additionally, let $q(s', o')$ be any fact in the KG with q being a relation. We are concerned with the message function from equation (2.14) that sends a message, for instance, from s' to o' according to $q(s', o')$, i.e., s' being a neighbour of o' . The message can be calculated as

$$\text{Message}_{s' \rightarrow o'} = \mathbf{z}_{s'}^{(l-1)} * (W_q \mathbf{p} + \mathbf{b}_q), \quad (2.16)$$

where $*$ denotes pointwise multiplication, W_q is a learnable weight matrix and \mathbf{b}_q an intercept term for relation q . Furthermore, \mathbf{p} is the relation embedding for relation p .

The weight matrix W_q and intercept term \mathbf{b}_q enable relation specific message passing, as in the message function of RGCN shown above. The matrix vector product with \mathbf{p} , however, makes the message specific to the current target fact to be scored.

Fact Scoring and Training

Query dependent message passing is enabled due to the descriptions above, independent of the choice of the Combine or Aggregate functions. Nevertheless, for Aggregate, the authors employ principal neighbourhood aggregation (Corso et al., 2020) followed by a feed-forward layer and a non-linear activation function for Combine. Fact scores are calculated by feeding the hidden representation of a candidate entity into a feed-forward layer to calculate a real-valued score. Similar to some KGE models, the model calculates directed fact scores. That is, the fact score for $p(s, o)$ is different when based on the tail query $p(s, ?)$ compared to it being calculated based on a head query. The reason is the boundary condition, i.e., the graph is populated by starting with the source entity of a query. Training of the model can be performed similar to KGE model training by defining a fact loss and a strategy for negative examples. In particular, the model is trained with negative sampling under the BCE loss.

Chapter 3

Comparing Rules and KGE Models

A rule-based approach and KGE models conceptually follow two fundamentally different paradigms for completing a KG. KGE models leverage continuous embeddings of the entities and relations of a KG. Patterns and regularities learned from the graph need to be encoded within these representations. They offer greater flexibility since the embeddings can also be employed in downstream tasks as feature vectors (Baier, Ma, and Tresp, 2017). At inference time, scoring functions are employed that perform vector operations over the embeddings. Therefore, fact inference can be performed independently of the original training KG. Rule-based approaches, on the other hand, encode the patterns of a KG explicitly with human-readable logical rules. They rely on the original KG to perform rule application for calculating fact predictions. This process ensures that rule-based predictions are fully transparent and explainable. Each new prediction made is directly caused by facts from the training KG in conjunction with the respective rule. A natural question that arises after contrasting both approaches conceptually, is how they compare empirically, both qualitatively (in terms of model behaviour) and quantitatively (in terms of their predictive performance).

Context and Goals

This chapter serves as an initial comparison of KGE models and the rule-based approach. After providing the basic definitions of terms and model characteristics in the previous chapter, in the following sections, a thorough comparison based on real scientific benchmarks will be provided. In particular, we seek to answer the question which of the model classes performs superior regarding the KGC task and how they compare on a qualitative level.

Therefore, we will first provide a qualitative comparison in which individual model predictions will be analysed and contrasted based on real examples. The goal is to provide the reader with a base intuition about the respective model behaviour. The examples will motivate further quantitative analyses within the chapter and the following chapters of the thesis. Subsequently, quantitative experiments will be presented comparing the models based on predictive performance. A particular focus will also be given to the question about the language scope of the two model classes. While a rule-based system might excel at exploiting crisp patterns such as symmetry and equivalence (Meilicke et al., 2018), one question is if there is anything hidden from the rules that can be learned by a KGE model. For the purpose of investigating this questions, particular designed experiments will be presented in which the rule-based approach proposes initial ranking candidates that are rescored by KGE models.

Contribution

Parts of this chapter have been originally published in a study that marks the beginning of the research period of this thesis (Meilicke, Betz, and Stuckenschmidt, 2021). This chapter is composed of the analysis part of this work with extended ablations regarding the rule-based approach. The contributions will be highlighted in the following. In particular, in this chapter:

- (i) We explore and compare KGE models and the rule-based approach qualitatively by analysing selected examples.
- (ii) We perform an initial quantitative comparison between KGE models and rule-based approaches in regard to defined confidence aggregation methods of chapter 2.
- (iii) We perform further ablations regarding the two model classes and experiments to investigate if patterns are hidden from the rule-based approach that can be learned by the KGE models.

The next section highlights related work in the context of rule-based and latent-based knowledge graph completion. In Section 3.2, we first introduce a metric to identify interesting cases in which model behaviour differs structurally and subsequently present and discuss selected examples. Section 3.3 discusses the quantitative experiments and Section 4.4 provides the experimental results. Finally, Section 4.5 concludes.

3.1 Related Work

The introduction of the seminal KGE models RESCAL in 2011 (Nickel, Tresp, and Kriegel, 2011) and TransE in 2013 (Bordes et al., 2013b) had a large influence on the research communities around KGs. Over the next years, many variants and new models were introduced. For instance, TransH (Wang et al., 2014), TransR (Lin et al., 2015), and TransF (Feng et al., 2016) are proposed augmentations of TransE. As mentioned in Section 2.7, DistMult (Yang et al., 2015) is a simplification of RESCAL and is itself extended by ComplEx (Trouillon et al., 2016) with vector operations in the complex space.

The increased attention for KGE models also motivated the work of Meilicke et al. (2018) who performed a fine-grained analysis based on different subsets of the data and proposed the symbolic baseline RuleN. It achieved surprisingly strong results on the datasets WN18 and FB15K (Bordes et al., 2013b) when compared to the evaluated KGE models.

Regarding the performance of KGE models, the work of Ruffinelli, Broscheit, and Gemulla (2020) showed that the predictive performance of various KGE models is relatively similar, when trained under a combined codebase while using the same protocol of hyperparameter optimization. The training code for the models was also published via the libKGE library (Broscheit et al., 2020) which allows easy access to the trained models based on the study of Ruffinelli, Broscheit, and Gemulla (2020). Meanwhile, the datasets FB51k and WN18 had been replaced with the more challenging versions Fb15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018). On the rule learning side, AnyBURL (Meilicke et al., 2019) was introduced as an augmentation of RuleN and its competitiveness was also confirmed shortly after in a study conducted by Rossi et al. (2021).

However the comparisons of KGE models and rule-based approaches in earlier works are potentially outdated. For example, while Meilicke et al. (2018) reported hits@10 scores for the FB15k-237 dataset of around 0.4 to 0.43 for the KGE models, the more recent studies report values of around 0.5 to 0.55 (Ruffinelli, Broscheit, and Gemulla, 2020; Rossi et al., 2021). The newest and reproducible results of the libKGE library and the strong performance of AnyBURL require a more recent comparison that will be provided in this chapter.

3.2 Qualitative Comparison

To obtain an initial understanding about the behaviour and differences of KGE models and rule-based approaches, this section will focus on a qualitative model comparison based on the benchmark Codex-M (Safavi and Koutra, 2020). In particular, we will discuss specific model predictions and rankings and aim to identify differences or similarities. First, we will introduce a difference metric that allows to find interesting cases.

3.2.1 Challenges of Model Comparison

Performing a meaningful comparison of individual candidate predictions between a KGE model and a rule-based approach is not straightforward. A direct score comparison between the same candidate predictions provides little insights due to the different score distributions: The aggregation scores of a rule-based approach are typically given by the confidences whereas a KGE model calculates real-valued scores. Using score normalization techniques does not make the scores immediately comparable. For example, KGE models are in general not calibrated (Tabacof and Costabello, 2020). Alternatively, ranking differences between the same candidates can be analysed as they directly determine the evaluated predictive performance of the models. However, this also has merits as the raw difference in ranks can bias the perceived model difference. We show a hypothetical example in Table 3.1.

	Max+ aggregation		KGE	
Rank	Candidate	Score	Candidate	Score
#1	english	0.80	italian	2.5
#2	italian	0.10	english	2.3
#3	french	0.10	french	-0.6

TABLE 3.1: Hypothetical rankings for the query *speaks(lisa, ?)* with correct answer *english*.

The table shows hypothetical head rankings for the query *speaks(lisa, ?)* with the correct answer *english*. The difference in ranks is $2-1=1$. It is the lowest possible difference in ranks except of the case where both models assign the same ranks. However, the candidate predictions of both models are structurally different. The KGE model ranks the correct answer at position two with a similar score as its top answer. The rule-based approach, on the other hand, finds only little evidence for all other candidates except of the true answer. The example might hint towards a structural difference that could not be found when only considering the difference in ranks. If, for instance, the confidence of the rule-based approach for *italian* would be close to that of *english*, the model behaviour would be more similar.

3.2.2 Metric Computation

We will define a metric that helps to identify interesting cases of model similarities and differences. It takes into account ranking differences of both models but it only compares aggregation scores from the rule-based approach. The discussion is based on a head query and the treatment for tail queries is equivalent.

Let $p(?, o)$ be a head query with correct candidate s and let $\phi_M(s)$ be its Max-aggregation score (compare Definitions 3 and 16). We aim to analyse the model differences with respect to their ranking position of s . We are given the head ranking of the rule-based approach and the Max-aggregation scores (ties are handled with Max+) and the KGE model ranking. First, we collect the Max-aggregation score $\phi_M(s)$. Subsequently, we collect the rank of s for the KGE model, i.e., the position of s in the KGE ranking. Then, we take the entity at this position in the rule-based ranking and collect its Max-aggregation score $\phi_M(s')$. The final metric is given by the ratio of scores,

$$\Psi(s) = \frac{\phi_M(s)}{\phi_M(s')}. \quad (3.1)$$

It holds that $\Psi \geq 0$ and we do not allow for values of zero in the denominator. The possible values of the metric will be described briefly in the following.

$\Psi(s) > 1$. The rule-based approach assigns a lower (better) rank to s but the magnitude of Ψ depends on the score distribution of the Max-aggregation scores. Therefore, if Ψ is large, from the rule-based perspective, the candidate rank of the KGE model is too high (its score is too low).

$\Psi(s) < 1$. The KGE model assigns a lower (better) rank to s . As above, the magnitude of Ψ identifies if there is a structural difference from the rule-based perspective.

$\Psi(s) \approx 1$. The behaviour of the two model classes is similar. If both models rank s on the same position, then $s = s'$ and $\Psi(s) = 1$. However the metric can also be close to one in cases where s is ranked on different positions.

In the example made in Table 3.1, the value of $\Psi(\text{english})$ would be large despite a small difference in the ranks. In the following, we will discuss selected cases from the benchmark Codex-M and compare the KGE models ComplEx (Trouillon et al., 2016) and RESCAL (Nickel, Tresp, and Kriegel, 2011) with the rule-based approach.

3.2.3 Selected Cases

In the following, four individual cases will be discussed. While rule predictions can be explained by the human-readable rules, findings about KGE model behaviour need to be considered with more caution. The examples are found by searching for significant values of the Ψ metric as defined in the previous section. In particular, the first two examples are based on large values of Ψ , the third example is based on values $\Psi < 0.45$ and for the last example the values are close to one.

Where is the City Darwin?

The city Darwin is a city in Australia (in the test set). We are concerned with the head query $\text{contains}(?, \text{Darwin})$ with correct answer *Australia*. Darwin is contained in the

Northern Territory (in the training set). The training set contains an additional fact that states that Darwin is also the capital of the Northern Territory. Finally, a fact in the training set states that Australia contains the Northern Territory.

Max+ aggregation			Complex		RESCAL	
Rank	Candidate	Score	Candidate	Score	Candidate	Score
#1	Australia	0.659	South Australia	11.780	New South Wales	1.683
#2	USA	0.032	Queensland	11.226	Australia	1.385
#3	Canberra	0.026	New South Wales	10.323	New Zealand	1.221
#4	South Australia	0.017	Western Australia	9.515	South Australia	0.276
#5	New South Wales	0.017	Tasmania	8.676	Queensland	0.249
#6	Western Australia	0.017	Victoria	8.539	United Kingdom	0.074
#7	Queensland	0.017	Australia	8.338	England	-0.064

TABLE 3.2: Rankings for the completion task *contains(?, Darwin)* based on Codex-M.

Table 3.2 shows the respective head rankings with Max-aggregation scores and Max+ tie handling for the rule-based approach. In the rule-based ranking, Australia is the first candidate with all other alternatives having low scores, while Complex ranks each Australian territory first before Australia appears at #7. RESCAL puts it on #2, however, its scores are similar to the scores of other alternatives that are clearly wrong. The two rules with the highest confidences that predict Australia with are shown below:

$$[0.659] \quad \text{contains}(X, Y) \leftarrow \text{ap}(A, X), \text{adjoins}(A, B), \text{capital}(B, Y) \quad (3.2)$$

$$[0.273] \quad \text{contains}(X, Y) \leftarrow \text{contains}(X, A), \text{contains}(A, Y) \quad (3.3)$$

The numbers in brackets are the standard rule confidences and *ap* is short for *administrative_parent*. Rule (3.2) is a complex rule with three body atoms whereas rule (3.3) expresses the transitivity of the *contains* relation. This rule would be sufficient already to rank Australia at the first position given the low scores of all other candidates.

Considering the score distribution of Complex and RESCAL, it appears that both regularities do not have a significant impact on the KGE rankings. They might rather be affected by the similarity that other Australian territories share with the Northern Territory, which is known to contain Darwin (all territories are within Australia, some share borders, etc.). The KGE rankings could approximately be replicated by substituting Northern Territory in the known fact *contains(Northern Territory, Darwin)* with entities that are similar, such as other Australian territories.

Who produced Tomb Raider?

We are now concerned with the test query *producedBy(TombRaiderI,?)*, where *TombRaiderI/II* refers to the first and second part of the movie series. The correct answer is Lawrence Gordon, who produced both movies. The actor Angelina Jolie starred in both movies and she also produced several other movies (but none of the Tomb Raider series) in the training set. The correct answer is ranked on top in the Max+ ranking (compare Table 3.3). The reason is a rule with confidence of 0.445 and two atoms in the body that says that a producer of a movie also produced the prequel of the movie. On the other hand, in the ranking of Complex, Angelina Jolie is placed at the first position. Although there is no strong rule supporting this answer, it

is somewhat plausible as she starred in the movie and also is known to be a producer in the training set for other movies.

Max+ aggregation			ComplEx		RESCAL	
Rank	Candidate	Score	Candidate	Score	Candidate	Score
#1	Lawrence Gordon	0.445	Angelina Jolie	10.064	Avi Arad	4.025
#2	Michael G. Wilson	0.222	Steven Spielberg	9.134	Kathleen Kennedy	3.833
#3	Steven Spielberg	0.200	Lawrence Gordon	9.112	Lawrence Gordon	3.077

TABLE 3.3: Ranking results for query *producedBy(TombRaiderI,?)* on Codex-M.

Who influenced Robert Schumann?

Given the test query *influencedBy(schumann,?)*, one correct answer is given by the composer Felix Mendelssohn. In the Max+ ranking the correct answer has a relatively high rank of #28 whereas the KGE models assign a rank of #9 and #4 (compare Table 3.4). Interestingly, ComplEx ranks Schumann on the first position predicting that he influenced himself. There exists several facts in the training set which describe persons that influenced Schumann and many of these are similar in their properties to Schumann itself. While it is possible within the language bias of, for example, AnyBURL, to make reflexive predictions, we found a stronger tendency of KGE models to make these predictions.

Max+ aggregation			ComplEx		RESCAL	
Rank	Candidate	Score	Candidate	Score	Candidate	Score
#1	Arthur Schopenhauer	0.36	Robert Schumann	9.062	Arnold Schoenberg	2.571
#2	Victor Hugo	0.25	Bach	8.232	Thomas Mann	2.514
#4	Spinoza	0.231	Schopenhauer	8.116	Mendelssohn	2.061
#9	Jean-Jacques Rousseau	0.153	Mendelssohn	7.517	Sigmund Freud	1.684
#28	Mendelssohn	0.07	Aleksandr Pushkin	5.853	Friedrich Hayek	0.427

TABLE 3.4: Ranking results for the query *influencedBy(schumann,?)* on Codex-M.

Where was Metropolis presented?

The test query *festivals(Metropolis,?)* asks at which festival the movie Metropolis has been shown. The correct answer is the 39th Berlin International Film Festival. In the Max+ ranking, the answer is ranked at position #18 with a low score of 0.016 and the position in the ComplEx ranking is #25.

In this case, the candidates that can be found before the correct answer are of higher interest. The first 23 AnyBURL ranks contain film festivals, followed by a mixed list that contains both festivals and cities (the relation *festivals* has sometimes been used to express that a movie has been shown in a certain city). In the ComplEx ranking, starting from position #23, nonsensical candidates appear: USA (#23), natural death cause (#24), Chris Parnell (#31), electric guitar (#33). Interestingly, these candidates are ranked above more meaningful candidates without any semantic reason. A possible explanation could be given by a small amount of training existing triples describing the meaningful candidates in conjunction with the random initialization of the embeddings before the training phase.

3.2.4 Influential Facts

For the Darwin example in the previous section, we showed two long rules that made the correct prediction Australia but seemingly did not affect the KGE ranking. In this section, on the other hand, we discuss with an example that there can indeed exist rules that are also relevant for the KGE model behaviour.

Michael Fisher works for King’s College

Consider the query $employer(fisher,?)$ that asks for one employer of Michael Fisher, a well-known mathematician and physicist, who studied at the King’s College (training set). The correct answer from the test set on which we focus here is likewise King’s College as fisher both worked and studied there. Moreover, the training set states that Fisher also worked for the Leiden University. In the Max+ ranking, the correct answer King’s College is ranked at position 10 only. On the other hand, ComplEx and most of the other KGE models achieve a better result and assign a very low rank of around one but always less than six.

From the set of learned rules, there exist two short rules that predict the correct answer:

$$employer(X, Y) \leftarrow studiedAt(X, Y) \quad f_1 : studiedAt(fisher, kingscoll) \quad (3.4)$$

$$employer(X, kingscoll) \leftarrow employer(X, leiden) \quad f_2 : employer(fisher, leiden) \quad (3.5)$$

On the right-hand side, the body groundings (f_1, f_2) of the rules are shown. Throughout this thesis, these facts will also be termed the influential facts or the explanations since they lead to the prediction of the correct answer.

Aggregating Different Signals

Both rules have a confidence of around 0.08. All remaining rules that also predict King’s College have a confidence smaller than 0.02. Aggregating both rules with the Noisy-or aggregation technique (compare Definition 6.1) would lead to a better ranking position of the correct answer in this case. This might hint towards a weakness of the Max-aggregation strategy in which only one rule can contribute to the final aggregation score. However, Noisy-or comes with other difficulties and we will discuss different aggregation strategies in Chapters 5 and 6 in more detail.

Even with Max-aggregation, both facts f_1, f_2 are important for the ranking. If only one fact would be removed from the KG, the ranking position of King’s College would be unchanged as both rules have a similar confidence. If both facts are removed, on the other hand, then King’s College falls out of the first 50 ranking positions regarding the Max+ ranking.

Influence on KGE Models

There are two considerations with respect to the comparison of rules and KGE models. (1) Do the influential facts f_1 and f_2 also affect the KGE models? (2) If so, do both facts contribute to the final score of the KGE models for King’s College?

For investigating these questions, we re-trained the KGE models ComplEx and Hitter* (Chen et al., 2021) on the training sets with their original hyperparameters using three different settings. In the first (second) setting, we remove f_1 (f_2) from the training set. In the last setting, both facts are removed. After training, we calculate the KGE model rankings for the test query $employer(fisher,?)$ with correct answer King’s

College as above. For each setting, the averages over six independent training runs are reported. Results are shown in Table 3.5.

	Complex		HittER*	
	Avg. Rank (Std)	Avg. Score	Avg. Rank (Std)	Avg. Score
All triples	1.5 (0.83)	5.53	1.66 (0.81)	5.16
Without f_1	2.83 (1.32)	4.95	10.8 (19.85)	4.39
Without f_2	31 (24.43)	3.19	44.16 (25.37)	1.30
Without f_1 and f_2	68 (53.28)	2.28	97 (74.63)	-0.17

TABLE 3.5: The impact of removing f_1 and f_2 on the scores and ranking position of King’s College.

In general, both influential facts also affect the behaviour of both KGE models significantly. Contrary to Max-aggregation, both models seem to aggregate the evidence given by each fact into a higher score. However, they seem to react less sensitive to the removal of f_1 where a higher effect can be observed for HittER*. If both facts are removed, for both models, the ranks drop below #50. This is similar to the observation made with respect to the Max+ ranking discussed above. Therefore, for this example we showed that there exist facts that are equally important for both model classes, while they differ in how the signals are aggregated into a final score.

The concept of using rules to identify influential facts that affect KGE model scores inspired our study about explaining KGE models with rules (Betz, Meilicke, and Stuckenschmidt, 2022a) and will therefore be revisited in the next Chapter.

3.2.5 Summary

We discovered differences and similarities between the two model classes based on individual examples. The core observations are highlighted in the following.

1. Candidates might appear in a KGE ranking due to a small amount of training triples for particular entities and the random initialisation of embeddings (e.g., Metropolis example, Section 3.2.3) or because of similarity considerations that are not supported by a strong regularity in form of a rule (e.g., Darwin and Tomb Raider examples, Section 3.2.3).
2. There are cases where the same facts from the training set are influential for the same candidate predictions made by both approaches (e.g., Fisher example, Section 3.2.4). However, this rather appears in cases where short rules are involved opposed to longer complicated rules as in the Darwin example.
3. There might be cases where Max-aggregation is inferior because it does not allow to combine meaningful evidence into a higher score (e.g., Fisher example, Section 3.2.4) in contrast to the KGE models. Also the Max+ ranking strategy can only incorporate more than one rule, if a tie was discovered between two candidates.

3.3 Quantitative Analysis

Next, we will focus on a quantitative comparison of the two model classes. We will do so by comparing the rule-based approach using the aggregation functions presented

in Chapter 2 with the KGE models trained and tuned under a unified protocol for hyperparameter search (Ruffinelli, Broscheit, and Gemulla, 2020).

Apart from the raw comparison of the predictive performance, based on the findings of the previous section, we are additionally interested in the question of the language scope of the model classes. Specifically, we seek to investigate if the KGE models learn patterns from the data that cannot be expressed with the rules. First, we will describe the practical difference that arise when calculating candidate rankings with the two approaches.

3.3.1 Differences in Ranking Calculation

We examined different candidate rankings of KGE models and the rule-based approach in the previous section. However, the approaches vary in how they calculate the rankings. The rule-based approach indeed proposes candidate predictions and scores them according to the defined aggregation function. On the other hand, KGE models do not directly predict candidates. Instead, they calculate scores for all possible facts that complete the query.

Suppose that $p(s, ?)$ is a query, and the task is to calculate a ranking of candidates from the set of entities \mathcal{E} . Moreover, let \mathcal{G} be the training KG. As shown by Definition 3, a score of some candidate o' is calculated as the score of the fact $p(s, o')$. Thus, a candidate ranking is a ranking of scores concerning facts that complete the query. A KGE model calculates all possible candidate scores. It calculates all scores for all possible facts $p(s, o')$ with $o' \in \mathcal{E}$.

The rule-based approach does not compute scores for all possible facts. It only calculates the facts that complete the query and are predicted by at least one rule. That is, it proposes the candidates from the set $\{o' \mid o' \in \mathcal{E} \text{ and } \mathcal{R} \cup \mathcal{G} \models_1 p(s, o')\}$. The scores are then calculated according to the defined confidence aggregation functions based on the predicting rules of each fact as shown in Algorithm 1.

If a pattern exists in the KG that causes a fact to be true, and this pattern is not expressible by the rules, then the corresponding fact will never be predicted and the candidate will not appear in the ranking. This observation enables us to conduct a simple experiment to investigate whether there are patterns used by the KGE models that cannot be expressed with the rules.

3.3.2 KGE-Rescoring

We propose a simple experiment to investigate if the KGE models exploit patterns that are hidden from the rule-based approach. In particular, for each test query, we let the rule-based approach calculate a ranking of maximum 100 candidates according to the Max+ aggregation strategy. Subsequently, we assign new scores to the candidates calculated with the KGE model. We can calculate the evaluation metrics based on this new ranking and compare them with the original results of the approaches.

The initial decision about the candidates in the ranking is based on the rule-based approach. Therefore, if the rules fail to predict correct candidates that would have obtained high scores by the KGE models, we should observe a drop in the predictive performance when comparing the final evaluation metric with the original KGE evaluation. On the other hand, the procedure could be beneficial for the predictive performance since only candidates will be proposed that are supported by at least one rule. The approach will be termed *KGE-rescoring* in the experimental section. The benchmarks that are used within this chapter range from 14k to 40k entities. For instance, proposing 100 entities randomly and scoring them with a KGE model would

lead to poor evaluation results. Moreover, quite often, when calculating the top-100 candidates, the actual set of proposals made by the rule-based approach contains fewer than 100 candidates.

3.4 Experiments

In the following sections, various experiments will be presented concerned with the comparison of KGE models and the rule-based approach. Additionally, different ablations will be performed based on the gained insights. First, the experimental setup will be introduced in the next section.

3.4.1 Experimental Setup

Benchmarks and Evaluation

We evaluate the approaches on Fb15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018). We also include the two larger graphs of the Codex benchmark (Safavi and Koutra, 2020). All evaluation results are based on top-100 rankings when evaluating the rule-based approach as described in Remark 2. They are based on the full rankings when reporting the plain KGE results. All reported metrics are based on the evaluation facts of the testing sets of the benchmarks and they are filtered as described in Section 2.2.

KGE Models and Rules

In regard to the KGE models, we use the libKGE library (Broscheit et al., 2020) which focuses on reproducibility and has shown to produce state-of-the-art results. We exclusively use this library for the experiments in this chapter to have trained embeddings under a joint codebase and experimental protocol.

We include TransE (Bordes et al., 2013b), RESCAL (Nickel, Tresp, and Kriegel, 2011), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), and TuckER (Balazevic, Allen, and Hospedales, 2019). For all these models, we use the pre-trained embeddings provided by libKGE which are based on the work from Ruffinelli, Broscheit, and Gemulla (2020). Finally, we use the no-context version of the KGE model Hitter (Chen et al., 2021) which is also available in libKGE. It is based on a transformer architecture (Vaswani et al., 2017) and provides strong predictive performance results. The reported evaluation performance matches the results of the original publication.

Benchmark	Number of Facts			Num Rules
	Train	Valid	Test	
Fb15k-237	272 115	17 535	20 466	5 084 903
WN18RR	86 835	3 034	3 134	97 329
Codex-M	185 584	10 310	10 311	7 409 385
Codex-L	551 193	30 622	30 622	4 160 476

TABLE 3.6: Number of facts and learned rules on each benchmark.

For the rule-based approach, we learn rules with AnyBURL for 3600 seconds for each benchmark on the training KGs. We use the default learning parameters provided on the AnyBURL homepage. This involves a maximal rule length for **B**-rules of five on WN18RR and a maximal length of three for the remaining benchmarks. For all other

rule types, the maximal rule length is one. We show the number of learned rules and the number of facts in Table 3.6. The number of learned rules is related to the size of the training KGs. However, a larger number of training facts not necessarily leads to a larger number of rules. Nevertheless, the number of rules is generally large; for each dataset, it exceeds the number of facts from the training KG. This characteristic will be discussed again several times throughout the remainder of this thesis. Finally, for the confidence aggregation functions, we use the rule confidence as specified in Definition 13 with a smoothing parameter of 5 as described in Remark 7.

3.4.2 Summarized Comparison

We will first discuss summarized results in which we compare the best KGE models with the rule-based approach. In Figure 3.1, the MRR results on the test sets of the respective benchmarks are shown. For the KGE models, we select the one with the highest MRR on each dataset. For WN18RR, the selected model is ComplEx. For the remaining datasets, the selected model is HittER*. For the rule-based approach, we use the Max+ aggregation ranking strategy (compare Definition 17). The comparison provided in the figure also denotes the starting point that was given at the beginning of the research period of this thesis.

The predictive performances of the two model classes are relatively similar. Nevertheless, the absolute magnitudes of the differences are often perceived of being significant in the KGC community. Additionally, there is little variance when training KGE models and the results of Max+ aggregation are deterministic.

The most relevant benchmarks within this thesis will be Fb15k-237, WN18RR, and Codex-M. For these datasets, the rule-based approach is inferior on Fb15k-237 and Codex-M and it achieves a higher result when compared against the best performing KGE model (ComplEx) on WN18RR.

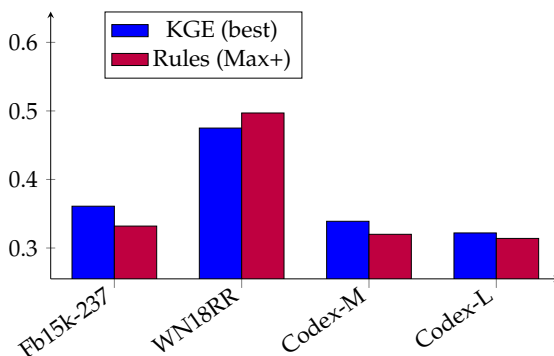


FIGURE 3.1: The MRRs of the best KGE model on each dataset compared with Max+ aggregation.

For WN18RR, the rule-based approach benefits from rules up to length five. This aligns with the discussion in Section 3.2. We hypothesized that KGE models are less affected by longer regularities opposed to shorter ones. For the datasets Fb15k-237 and Codex-M, on the other hand, rules with more than three body atoms do not provide any further benefits to the evaluation results of the Max+ ranking. To summarize, the rule-based approach is superior to all KGE models on one dataset where longer rules are important and slightly inferior on the remaining datasets when compared to the best performing model.

3.4.3 Detailed Comparison

Detailed results showing the predictive performance on the respective test sets are provided in Tables 3.7 and 3.8.

The H@1 results for most approaches are in the range from 0.22 to 0.27 on all benchmarks except for WN18RR. That means that the correct candidate is ranked at the first position in 22-27 percent of the queries. Likewise, in roughly 40-56 percent of the queries, the correct candidate is ranked within the first 10 positions, as shown by the H@10 results. Interestingly, for WN18RR, the H@1 values are commonly higher compared to the other benchmarks although the H@10 values are relatively similar. In the following, we will compare the approaches based on the MRR which leads to the same qualitative conclusions.

Approach	Fb15k-237			WN18RR		
	H@1	H@10	MRR	H@1	H@10	MRR
ComplEx	0.253	0.536	0.347	0.438	0.547	0.475
ConvE	0.248	0.521	0.338	0.411	0.505	0.442
HittER*	0.268	0.549	0.361	0.437	0.531	0.469
RESCAL	0.263	0.541	0.355	0.439	0.517	0.457
TransE	0.221	0.497	0.312	0.053	0.520	0.228
DistMult	0.249	0.531	0.342	0.414	0.531	0.452
Max+	0.246	0.506	0.331	0.457	0.574	0.497
Noisy-or	0.251	0.499	0.333	0.391	0.560	0.446

TABLE 3.7: Detailed results for the KGC task on Fb15k-237 and WN18RR.

Over all benchmarks, the best-performing KGE model is HittER*, followed by ComplEx and RESCAL. On WN18RR, ComplEx performs slightly better than HittER* and clearly better than the remaining KGE models.

On the Fb15k-237 dataset, all plain KGE models achieve higher performance compared to Max+ aggregation, except for TransE. For instance, on Fb15k-237, HittER* achieves an MRR of 0.361, whereas Max+ aggregation achieves 0.331. On WN18RR, on the other hand, Max+ aggregation achieves an MRR of 0.497, while the best KGE model, ComplEx, achieves 0.475. On this benchmark, as previously mentioned, B-rules are of higher importance for the rule-based approach. We will also demonstrate this with an ablation study in Section 3.4.6.

The results for Codex-M are most similar to Fb15k-237 where Max+ aggregation is slightly inferior to some of the KGE models, while being on par with RESCAL and ConvE and being better than TransE. For Codex-L, only HittER is superior to Max+ aggregation. The remaining models are slightly inferior, for instance, ComplEx achieves an MRR of 0.294 while Max+ aggregation achieves 0.320.

When comparing the Noisy-or aggregation strategy with Max+, it is clearly outperformed on all benchmarks except for Fb15k-237. Here, Max+ achieves an MRR of 0.331 and Noisy-or reports 0.333. On WN18RR it only reports 0.446 which is similar to the average performance of the KGE models. In Chapter 5, we will provide a detailed discussion about these differences and also develop a formal argument why Noisy-or is inferior in our setting.

Approach	Codex-M			Codex-L		
	H@1	H@10	MRR	H@1	H@10	MRR
ComplEx	0.262	0.476	0.337	0.237	0.400	0.294
ConvE	0.239	0.464	0.318	0.240	0.420	0.300
HittER*	0.262	0.486	0.339	0.257	0.447	0.322
RESCAL	0.244	0.456	0.317	0.242	0.419	0.304
TransE	0.223	0.454	0.303	0.116	0.317	0.187
TuckER	0.259	0.458	0.328	0.244	0.430	0.309
Max+	0.249	0.456	0.320	0.256	0.427	0.314
Noisy-or	0.219	0.427	0.290	0.246	0.430	0.308

TABLE 3.8: Detailed results for the KGC task on Codex-M and Codex-L.

3.4.4 KGE-Rescoring Results

Next, we will introduce the results for the KGE-rescoring experiment as described in Section 3.3. Figure 3.2 depicts the results for the datasets Fb15k-237 and WN18RR. In the figure, we mark the Max+ aggregation result (dashed line), the original KGE model results shown in the previous section (bottom symbol), and the result of the KGE-rescoring experiment (top symbol).

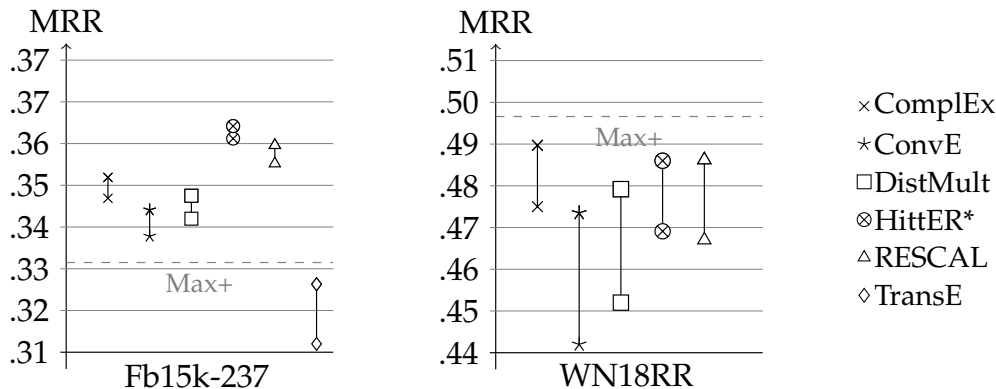


FIGURE 3.2: MRRs of Max+, original KGE model (bottom), and KGE-Rescoring (top)

The improvements over the original KGE evaluation are generally small. The differences are larger for WN18RR although they do not result in a higher MRR than the Max+ aggregation result. However, it is striking that there does not exist a single experiment in which the original predictive performance of the KGE model decreases. Therefore, calculating the initial candidate proposals with the rule-based approach does not miss important candidates associated with patterns that cannot be expressed with the rules.

Table 3.9 shows detailed results for the KGE-rescoring experiments on all datasets. In the second column, the improvement over the original KGE evaluation results from Tables 3.7 and 3.8 is shown. Also for the remaining datasets, the final evaluation result of the rescoring approach does never lead to a decrease of the evaluation results. We observe the strongest improvement for TransE on WN18RR with 19.4 percentage

Approach	KGE-Rescoring			Improvements			
	H@1	H@10	MRR	H@1	H@10	MRR	
Fb15k-237	ComplEx	0.259	0.541	0.352	+0.006	+0.005	+0.005
	ConvE	0.254	0.528	0.344	+0.006	+0.008	+0.006
	DistMult	0.255	0.536	0.348	+0.006	+0.004	+0.005
	HittER*	0.273	0.552	0.364	+0.005	+0.002	+0.003
	RESCAL	0.268	0.546	0.360	+0.005	+0.006	+0.004
	TransE	0.237	0.509	0.326	+0.016	+0.012	+0.014
WN18RR	ComplEx	0.448	0.575	0.490	+0.010	+0.027	+0.015
	ConvE	0.434	0.551	0.474	+0.023	+0.047	+0.032
	DistMult	0.437	0.563	0.479	+0.023	+0.033	+0.027
	HittER*	0.447	0.566	0.486	+0.010	+0.035	+0.017
	RESCAL	0.449	0.557	0.486	+0.010	+0.04	+0.019
	TransE	0.337	0.570	0.422	+0.284	+0.049	+0.194
Codex-M	ComplEx	0.266	0.487	0.341	+0.003	+0.011	+0.004
	ConvE	0.248	0.475	0.325	+0.009	+0.011	+0.007
	HittER*	0.267	0.487	0.342	+0.006	+0.001	+0.003
	RESCAL	0.250	0.469	0.323	+0.006	+0.014	+0.006
	TransE	0.238	0.461	0.312	+0.015	+0.008	+0.009
	TuckER	0.262	0.471	0.333	+0.003	+0.013	+0.005
Codex-L	ComplEx	0.247	0.430	0.309	+0.010	+0.030	+0.015
	ConvE	0.247	0.439	0.312	+0.007	+0.019	+0.012
	HittER*	0.262	0.453	0.327	+0.006	+0.007	+0.005
	RESCAL	0.252	0.436	0.314	+0.010	+0.017	+0.010
	TransE	0.172	0.363	0.236	+0.056	+0.046	+0.049
	TuckER	0.254	0.442	0.318	+0.010	+0.012	+0.009

TABLE 3.9: Detailed results for the rescoring experiments. The first column section shows the results when proposing candidates with the rules and rescoring them with the respective KGE model. The second column section shows the improvement over the respective original KGE results provided in Section 3.4.3.

	Fb15k-237	WN18RR	Codex-M	Codex-L
Percent	0.253	0.298	0.347	0.413

TABLE 3.10: The proportion of correct candidates missing from the top-100 answers for a Max+ ranking on the different benchmarks.

points. Nevertheless, the resulting MRR of 0.422 is still lower compared with the best KGE models or Max+ aggregation.

The results show that the rule-based approach is able to propose all the candidate answers needed to reach the performance level of the KGE models. One could argue that the KGE-rescoring experiment should be more restrictive since passing the top-100 candidates to the KGE models should always allow the model to assign a high score to the correct answer. However, in practice, the correct candidate often does not appear within the top-100 candidates. If the correct candidate is missing, the evaluation metrics adds a contribution of zero to the overall metric for the current query. For example, Table 3.10 reports for each benchmark the proportion of correct answers not ranked within the top-100 by Max+. On Codex-L, 41.3 percent of queries have the correct answer missing from the candidates passed to the KGE model (this value is simply $1-H@100$). Still, the KGE models show similar performance gains on this benchmark. This suggests that, even in the original KGE rankings, these candidates would have received relatively low scores already; otherwise, we would see a drop in the MRR.

In Chapter 4, we will further examine to which extent both models use the same facts from the training KG when making predictions at test time.

3.4.5 Averaging KGE Models

We discussed in Section 3.2 that nonsensical candidates can appear within a KGE ranking above the correct answer. One potential reason is that these candidates appear randomly due to not enough training signal and the random initialization of the embeddings. Proposing the initial candidates with the rule-based approach, on the other hand, can prevent them from appearing in the final ranking and might be an explanation for the improvements reported with the rescoring experiment in the previous section.

If the nonsensical candidates are indeed random, however, an alternative is to simply train a KGE model multiple times and to average the results. In the best case, it allows good candidates to maintain their position as they are proposed due to non-random patterns. By averaging the same model and initializing embeddings differently in every training run, the random candidates, however, might fall out from the top ranking position.

Table 3.11 shows results for ComplEx and DistMult on WN18RR. We report the lowest and highest results and the averaged models out of five runs. Interestingly, while only observing a low variance between the individual runs, the averaged results are higher. For instance, the difference for ComplEx of the best and the worst run is only 0.02 while the averaged model achieves an MRR that is one percentage point higher than the best individual run. For DistMult, the improvement in regard to the original result is 1.7 percentage points. The improvements when calculating the model averages are nevertheless lower when compared to the KGE-rescoring experiment.

	Worst ↔ Best MRR	Averaged MRR	Original MRR	KGE-Rescoring MRR
ComplEx	0.474 ↔ 0.476	0.486	0.475	0.490
DistMult	0.451 ↔ 0.454	0.459	0.442	0.479

TABLE 3.11: Example results for WN18RR when training models five times. Best and worst results are shown as well as an average ensemble of all runs (joint).

3.4.6 Rule-Based Ablations

When introducing the different rule types in Section 2.3.2, it was already highlighted that **B**-rules and U_c -rules are most important. We will investigate and discuss the different effects of different subsets of rule types in the following. Table 3.12 shows results for three datasets for different specifications of rules. For each section of the table, the first row contains the original results using all the rules. Each subsequent row shows results when removing all rules of the respective type. The last row depicts results when using **B**- and U_c -rules exclusively.

		Fb15k-237			WN18RR			Codex-M		
Rules		H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
Max+	All (\mathcal{R})	0.246	0.506	0.331	0.457	0.574	0.497	0.249	0.456	0.320
	$\mathcal{R} \setminus U_z$	0.246	0.505	0.331	0.455	0.572	0.494	0.248	0.455	0.318
	$\mathcal{R} \setminus U_d$	0.246	0.505	0.332	0.457	0.571	0.496	0.248	0.455	0.318
	$\mathcal{R} \setminus U_c$	0.203	0.431	0.277	0.444	0.559	0.483	0.183	0.372	0.244
	$\mathcal{R} \setminus \mathbf{B}$	0.242	0.488	0.325	0.220	0.287	0.244	0.236	0.446	0.308
	B + U_c	0.246	0.505	0.331	0.455	0.567	0.492	0.248	0.455	0.318
Noisy-or	All (\mathcal{R})	0.251	0.499	0.333	0.391	0.560	0.446	0.219	0.427	0.290
	$\mathcal{R} \setminus U_z$	0.251	0.499	0.333	0.391	0.559	0.444	0.219	0.427	0.290
	$\mathcal{R} \setminus U_d$	0.251	0.499	0.333	0.391	0.560	0.446	0.219	0.427	0.290
	$\mathcal{R} \setminus U_c$	0.210	0.443	0.288	0.373	0.550	0.431	0.183	0.392	0.252
	$\mathcal{R} \setminus \mathbf{B}$	0.248	0.498	0.332	0.206	0.288	0.235	0.205	0.428	0.281
	B + U_c	0.251	0.498	0.331	0.391	0.559	0.444	0.219	0.427	0.290

TABLE 3.12: Ablation results for selected datasets. Performance results when using different subsets of rules are shown.

The types **B** and U_c determine the final predictive performance to the most extent. In fact, using only these rule types leads to a decrease in the performance that is in the range of only 0.2-0.5 percentage points. Likewise, there is only a small or no influence when removing any of the other rule types individually. The results also confirm our findings from earlier sections in regard to WN18RR. On this dataset, longer **B**-rules have the largest influence. For instance, when removing **B**-rules the MRR drops to 0.244 under Max+ whereas it only drops to 0.483 when removing U_c -rules. For Fb15k-237, on the other hand, **B**-rules have less influence. Here, the MRR drops to 0.325 when removing **B**-rules and it drops to 0.277 when removing U_c -rules. Nevertheless, both rule types are required on all datasets to achieve state-of-the-art results.

3.5 Conclusion

In this chapter we, compared the rule-based approach with KGE models both qualitatively and quantitatively. Our initial quantitative comparison showed that the predictive performance of the model classes is not that different in general. However, on 3 of 4 datasets, the best KGE model achieved a higher MRR. The benchmark on which the rules are superior is WN18RR, a KG where longer rules are important. This comparison is based on selecting the best KGE model on each dataset and the results are obtained from the libKGE library to have a unified codebase.

We discussed individual example predictions with the goal of understanding model behaviour and identifying advantages and disadvantages of both model classes. We could observe cases in which the KGE models were seemingly affected by

the same facts that are also relevant for rule predictions. The relevant rules in these cases were short rules opposed to long rules. This observation is also supported in a more recent work by Liu et al. (2023).

When proposing candidates with the rule-based approach and rescored them with KGE models, their evaluation results always improves. This suggests that the rule-based approach remains relatively within the language scope of the KGE models. We could not find any conclusive evidence that KGE models exploit patterns that cannot be expressed with rules. For instance, if a KGE model predicts a candidate because of a similarity consideration, this can potentially be represented by multiple U_c -rules. However, a more detailed investigation is required which will be the topic of Chapter 4.

On the other hand, we identified one possible disadvantage in the fact that Max-aggregation cannot increase a candidate score when multiple rules predict the same candidate. However, taking into account all the rules with the Noisy-or product does not solve the problem and leads to a worse predictive performance on average. Therefore, a thorough analysis of the confidence aggregation problem is required to find a better suited strategy which will be presented in Chapter 5.

Chapter 4

Explaining KGE Models with Rules

The black-box nature of neural and latent-based models is a rising concern in the machine learning community (Burkart and Huber, 2021). When inference mechanisms lack transparency due to large parameter spaces and complex scoring mechanisms, potential users do not trust the models. Therefore, a substantial amount of research aims to either find causes for model predictions or adapt models for more transparency (Ribeiro, Singh, and Guestrin, 2016; Lundberg and Lee, 2017; Lundberg et al., 2020; Bechler-Speicher, Globerson, and Gilad-Bachrach, 2024). The fact scores of KGE models are not directly interpretable due to the latent learning of the embeddings. Therefore, efforts to explain predictions are also made in the context of KGE models (Rim et al., 2021). Rule-based approaches, on the other hand, are inherently interpretable, and in this chapter, we develop and investigate approaches to utilize them as an interface for explaining KGE predictions.

Context and Goals

In the qualitative analyses in the previous chapter, we discussed individual cases in which the same facts seemingly influenced predictions of both the rule-based approach and the KGE models. For instance, in the example presented in Section 3.2.4, the task was to identify King’s College as one employer of Michael Fisher. We identified the facts that Fisher studied at the King’s College and that he was employed at the Leiden University as important facts for the prediction. Furthermore, we were able to show that these facts influenced both the ranking of the KGE model and that of the rule-based approach. Additionally, the KGE-rescoring experiment demonstrated that the rule-based approach is able to propose all necessary candidates required to obtain the predictive performance of a KGE model.

These findings indicate that KGE models and the rule-based approach might, to some extent, rely on similar underlying patterns. However, the insights are obtained by individual examples or by observing aggregated performance metrics and are not yet conclusive. Therefore, in this chapter, we put these observation to the test through a systematic analysis. In particular, we aim to quantify the degree to what extent both model classes are influenced by the same facts from the training KG when making predictions at inference time.

To this end, we propose a method that allows to calculate explanation for KGE models with the rule-based approach. An explanation is a fact from the training KG that is influential for the score of a target fact at inference time. If KGE models are influenced by the rule patterns, then the rules should be able to reliably identify these influential facts. Therefore, we evaluate our method against other approaches. In particular, the evaluation is performed in the context of adversarial attacks against KGE models (Bhardwaj et al., 2021). In this setup, an attacker has access to the trained models and can corrupt or perturb the training data. We will argue in the following

sections, however, that the evaluation likewise can measure the degree to which extent an approach can explain the predictions of a KGE model.

Contribution

The contents of this chapter were originally published in the work from Betz, Meilicke, and Stuckenschmidt (2022a). The study was directly influenced by the experiments and analyses presented in Chapter 3. In this chapter, we added extended experiments and ablations. The key contributions can be summarized as follows.

- (i) We propose a method for performing adversarial attacks against KGE models that is inspired by abductive reasoning. Although our approach does not rely on access to the training data or knowledge about the attacked models, it performs on-par with other approaches that rely on these elements.
- (ii) We propose a revised evaluation protocol after identifying interpretation issues in regard to attack quality in the most recent protocol (Bhardwaj et al., 2021).
- (iii) We demonstrate that the method additionally provides a fully interpretable explanation of the underlying statistical regularity that leads to a prediction.

In Section 4.1, the problem statement will be introduced and Section 4.2 discusses related work. The proposed method will be introduced in Section 4.3. Experimental results we be discussed in Section 4.4 and Section 4.5 concludes.

4.1 Problem Setting

4.1.1 Attacks and Explanations

The problem setting of this chapter is based on adversarial attacks against KGE models. The concept of an attack involves a minimal modification of the training KG such that the score ϕ_{KGE} of a correct fact decreases after re-training the model, leading to a decline in the predictive performance. Typically, adversarial attacks are studied with the motivation of investigating model security and robustness (Chakraborty et al., 2021). However, an attacking method that can effectively determine a modification to the training data that causes the most damage during inference, also provides an explanation for the model prediction under the original training data. In other words, increased attack efficacy results in superior explanation quality in regard to the original model.

4.1.2 Attack Protocol

In our definition of an adversarial attack, we follow the framework proposed by Bhardwaj et al. (2021). As mentioned above, the goal is to perturb the training KG such that the predictive performance of a model declines after re-training. To that end, let \mathcal{G} be the training KG and let \mathcal{E} denote entities and \mathcal{P} the relations. Furthermore, let us assume a KGE model was trained on \mathcal{G} . Let $f = p(s, o)$ be a fact which is known to be correct but does not exist in \mathcal{G} , e.g., a fact from the test set. Within this chapter, f is termed the *target* of an attack. We define two distinct attacking settings in the following:

Adversarial Deletion. For each target $p(s, o) \notin \mathcal{G}$, we are allowed to delete one fact from \mathcal{G} that shares at least one common entity with the target. That is, we have to

select and delete a $p'(s', o) \in \mathcal{G}$ or $p'(s, o') \in \mathcal{G}$ with $o, s, o', s' \in \mathcal{E}$ and $s' \neq s$ and $o' \neq o$ and $p, p' \in \mathcal{P}$. It is allowed that $p' = p$.

Adversarial Addition. Instead of deleting a fact, we add one fact to \mathcal{G} for each target. The restrictions of the fact to be added are the same as in the deletion setting, i.e., it has to share one entity with the target fact.

The fact that is deleted (added) based on the adversarial deletion (addition) is termed the attacking fact. Throughout the remaining sections, the two specifications are also termed the *Add* setting and the *Del* setting.

In the procedure, we first train a KGE model on the original training graph \mathcal{G} . Subsequently, target facts which achieved a strong initial performance in terms of MRR are selected from the test set. This will be explained in more detail in the experimental sections. Finally, the attacking facts are selected by the respective attack strategy and deleted or added to the training KG. After re-training the models, the performance metrics are compared. We show pseudocode for the procedure in Algorithm 2 based on one single target fact.

Algorithm 2 Adversarial Attacks for KGC

Input: Training KG \mathcal{G} , KGE model ϕ_{KGE}

- 1: **procedure** ATTACK PROTOCOL
- 2: $\phi_{KGE} \leftarrow \text{trainKGE}(\mathcal{G})$
- 3: $f \leftarrow \text{select target fact } p(s, o)$
- 4: $mrr \leftarrow \text{calculate MRR of } f \text{ based on scores of } \phi_{KGE}$
- 5: $\mathcal{G}' \leftarrow \text{performAttack}(f, \mathcal{G})$ *// delete or add one fact from/to \mathcal{G}*
- 6: $\phi'_{KGE} \leftarrow \text{trainKGE}(\mathcal{G}')$
- 7: $mrr' \leftarrow \text{calculate MRR of } f \text{ based on scores of } \phi'_{KGE}$
- 8: $diff = mrr - mrr'$ *// calculate attack strength*

The most accurate procedure is to re-train the KGE model individually for each selected target fact as shown in the pseudocode. This would mitigate any side effects that attacking facts might have on other target facts. However, this is computationally not feasible. We therefore opt to keep the sets of target facts small and repeat the whole procedure for multiple times with different target facts instead.

4.2 Related Work

Studying the concept of adversarial attacks is closely related to searching for influential data points. For example, Hanawa et al. (2021) investigate relevance metrics for instance-based explanations, i.e., explaining model predictions by similar training instances, and Charpiat et al. (2019) aim to express similarities from the neural network perspective. Koh and Liang (2017) apply influence functions in the context of image classification to trace back model predictions towards individual training images. Lawrence, Szttyler, and Niepert (2021) estimate individual influences of facts by first tracking for every training instance the gradient-based updates induced to its parameters. The influence of a fact f' on a target fact f is then estimated by the difference of the original score of f and the score calculated in regard to parameters where the accumulated updates induced by f' are subtracted (rolled back).

In the context of adversarial attacks, influential data points are identified with the motivation to cause the most harmful effect on a particular model prediction by

perturbing the data. Pezeshkpour, Tian, and Singh (2019) study the robustness of KGE models by defining adversarial modifications in the context of KGC queries. The change in the fact score of an attack is estimated by using a Taylor approximation and the attacking facts are selected by a parameterized decoder that maps the maximal change in score vectors back to entities and relations in the embedding space. Zhang et al. (2019) investigate data poisonous attacks against KGE models by defining *Direct* attacks. An embedding shifting vector of a target fact is defined as the negative gradient of the scoring function and the attacking facts are selected by calculating a perturbation benefit score based on this vector for every candidate fact. The methods proposed by Bhardwaj et al. (2021) represented state-of-the-art during developing our methods and are used as the main comparison in our experimental section. The attack setting is based on instance attribution methods and the main specifications are based on selecting the attacking facts by computing similarities to the target facts similar to instance-based explanations (Hanawa et al., 2021). These similarities are either based on the embeddings or the gradients of the loss functions with respect to the candidate facts.

4.3 Method

In an attack scenario, one can distinguish between white-box and black-box methods, with recent literature focusing on white-box approaches (Pezeshkpour, Tian, and Singh, 2019; Bhardwaj et al., 2021; Lawrence, Szttyler, and Niepert, 2021). These methods have full access to the trained model and further specifications. In particular they have access to (i) the embeddings that have been learned and (ii) the method that was used to learn these embeddings (e.g., the scoring and loss functions).

A black-box method, on the other hand, does not have access to (i) or (ii). It is solely driven by the underlying assumptions that it identifies reasons (in form of facts) for predictions that are generically important for the attacked methods. Motivated by the findings of Chapter 3, we base the attacking method on the rule-based approach. The comparison to state-of-the-art white-box attack methods allows to obtain further insights about the differences and similarities of KGE models and rule-based approaches.

4.3.1 Abductive Reasoning

The proposed approach is inspired by the concept of abductive reasoning (Mayer and Pirri, 1993). The goal is to find an explanation E for an observation f given a theory Φ such that $\Phi \cup E \models f$ with $\Phi \not\models f$ and $\Phi \cup E$ is consistent. An explanation E is minimal if for each $E' \subset E$ we have that $\Phi \cup E' \not\models f$. We refer to minimal explanations when mentioning explanations in the following paragraph.

Apart from minimality, explanations are usually described according to some desired criteria such as their simplicity. For the given setting in this thesis, however, we propose a definition for a *best* explanation which takes into account the rule confidences. For our context, we set the theory to be an initial set of rules, i.e., we set $\Phi = \mathcal{R}$ and an explanation E is a set of facts. Finally let $\min_{conf}(\mathcal{R}')$ denote the smallest confidence of all rules contained in a set of rules \mathcal{R}' .

Definition 19 (Best explanation). *Let f be a target fact, E be an explanation, and \mathcal{R} a set of rules. We say that E is the best explanation with respect to \models iff there exists a $\mathcal{R}' \subseteq \mathcal{R}$ with $\mathcal{R}' \cup E \models f$ such that for any other $\mathcal{R}'' \subseteq \mathcal{R}$ and E'' with $\mathcal{R}'' \cup E'' \models f$, we have that $\min_{conf}(\mathcal{R}') \geq \min_{conf}(\mathcal{R}'')$.*

In other words, an explanation is *best*, if it entails the target together with the subset of rules which has the highest minimal confidence under all other possibilities of explanations and rule subsets. In general, the definition does not require a *best* explanation to be unique.

4.3.2 Calculating Explanations

Abductive reasoning is intractable in general (Bylander et al., 1991). Nevertheless, the main inference mechanism considered in this thesis is the one-time application of rules defined by one-step entailment \models_1 (Definition 10) which drastically simplifies the process of calculating explanations. For instance, one-step entailment can conceptually be calculated by chaining database joins which has polynomial time complexity.

Let \mathcal{R} be the set of rules learned on the training KG \mathcal{G} . For a target fact f , we collect all the rules from \mathcal{R} that predict f , denoted by \mathcal{R}_f , with respect to \mathcal{G} . We only regard explanations which are contained in \mathcal{G} . Therefore, we identify the rule with the highest confidence from \mathcal{R}_f , denoted by r^* . From r^* , we collect the body grounding that leads to the predictions which is a set containing one or more facts from \mathcal{G} (Definition 9). If there are multiple body groundings, we randomly select one. Finally, we set the explanation E to the selected body grounding. It defines the facts that ground the rule with the highest confidence such that the target f is predicted.

By following this procedure, the explanation E is constructed such that it holds $\{r^*\} \cup E \models_1 f$ and due to $r^* \in \mathcal{R}$ we have that $\mathcal{R} \cup E \models_1 f$ according to the definition of one-step entailment. The rule r^* is the rule with the highest confidence that predicts f . Therefore, for every other $r' \neq r^* \in \mathcal{R}$ with alternative explanation E' and $\{r'\} \cup E' \models_1 f$, we have that $\min_{conf}(\{r^*\}) \geq \min_{conf}(\{r'\})$. It follows that E is a *best* explanation with respect to \models_1 from all possible explanations in \mathcal{G} .

The explanation E is also an explanation with respect to \models as general entailment is implied by one-step entailment (compare Section 2.4). Nevertheless, it is not guaranteed that it is also a *best* explanation with respect to \models . For example, f could be derived by two reasoning steps including two rules that both do not predict f individually but have a larger confidence than r^* .

4.3.3 Defining the Attack

We first learn a set of rules \mathcal{R} on the training KG \mathcal{G} which has to be done only once for any attack related to the same data set. Recall the Darwin and Fisher examples from Sections 3.2.3 and 3.2.4. For the Fisher example, we observed that the grounding of short rules affected both the KGE and the rule-based ranking. On the other hand, for the Darwin example, a relatively complicated rule with three body atoms was the reason for the correct rule-based prediction. However the rule seemingly did not influence the KGE models. Therefore, we choose to only learn short rules. In particular, we learn **B**-rules of length one and two and **U_c**-rules of length one. We have also experimented with using different sets of rules, in particular, using a specification with all rule types and longer **B**-rules. This could not improve the efficacy of the approach on any of the benchmarks. An ablation study will be presented in the evaluation section. The rules that will be considered are shown in the following:

$$p_1(X, Y) \leftarrow p_2(X, Y) \quad (4.1)$$

$$p_1(X, Y) \leftarrow p_2(X, Z) \wedge p_3(Z, Y) \quad (4.2)$$

$$p_2(X, e) \leftarrow p_1(X, e') \quad (4.3)$$

The symbols p_1, p_2, p_3 denote arbitrary relations, e, e' are arbitrary entities and X, Y, Z denote variables. Note that only for the second rule type, an explanation will contain more than one fact.

We will continue with precisely discussing our approach after the rules are learned. To that end, let $f = p(s, o)$ denote the target fact. In both of the two settings, we calculate its *best* explanation E with respect to \models_1 based on the learned set of rules \mathcal{R} and \mathcal{G} as described in the previous section.

Deletion Attack

The explanation E denotes the facts that are the body grounding of the rule with the highest confidence that predicts the target fact. In the deletion setting, we delete E if it contains one fact and if it contains two facts, we delete one randomly. In the most cases, the explanation consists of only one facts as the largest proportion of rules have length one. As E is the body grounding of one of the rule types above, it will share one entity with the target fact $p(s, o)$ by construction (via the substitute of X or Y). Therefore the attack strategy adheres with the defined problem statement in Section 4.1. We show pseudocode for the whole procedure in Algorithm 3.

Algorithm 3 Rule-Based Deletion Attack

Input: Training KG \mathcal{G} , target fact $f = p(s, o)$, entities \mathcal{E}

Output: Perturbed training KG \mathcal{G}'

```

1: procedure DELETION ATTACK
2:    $\mathcal{R} \leftarrow \text{learnRules}(\mathcal{G})$ 
3:    $r^* = \arg \max_r \{ \text{conf}(r) \mid r \in \mathcal{R} \text{ and } r \cup \mathcal{G} \models_1 f \}$ 
4:   // selecting explanation (see Section 4.3.2)
5:    $E \leftarrow \text{select body grounding of } r^* \text{ that leads to predicting } f$ 
6:   if  $|E| == 1$  then
7:      $\mathcal{G}' = \mathcal{G} \setminus E$ 
8:   else
9:      $p'(s', o') \leftarrow \text{sample one fact from } E$ 
10:     $\mathcal{G}' = \mathcal{G} \setminus \{p'(s', o')\}$ 
11:  return  $\mathcal{G}'$ 

```

Addition Attack

As above, we select one fact from E randomly if it contains more than one facts or we select E if it contains only one fact.

Based on target fact $p(s, o)$, let $p'(s', o')$ denote the fact that is selected from E . By construction, we have that either $o' = o$ or $s' = s$. In the addition setting, we add a perturbation of $p'(s', o')$ to the training set. We first select an entity randomly from the training KG \mathcal{G} . We do this by sampling a fact from \mathcal{G} and subsequently randomly selecting the head or tail entity, denoted by e . If $o' = o$ then we add $p'(e, o')$ to the training KG and if $s' = s$ then we add $p'(s', e)$. By this procedure, the added fact

will always share an entity with the attacked target fact $p(s, o)$ as required by the problem statement. We show pseudocode for the addition attack in Algorithm 4.

Algorithm 4 Rule-Based Addition Attack

Input: Training KG \mathcal{G} , target fact $f = p(s, o)$, entities \mathcal{E}
Output: Perturbed training KG \mathcal{G}'

- 1: **procedure** ADDITION ATTACK
- 2: $\mathcal{R} \leftarrow \text{learnRules}(\mathcal{G})$
- 3: $r^* = \arg \max_r \{ \text{conf}(r) \mid r \in \mathcal{R} \text{ and } r \cup \mathcal{G} \models_1 f \}$
- 4: *// selecting explanation (see Section 4.3.2)*
- 5: $E \leftarrow$ select body grounding of r^* that leads to predicting f
- 6: $p'(s', o') \leftarrow$ sample one fact from E
- 7: $e \leftarrow$ sample one entity from \mathcal{E}
- 8: **if** $o' \in \{o, s\}$ **then**
- 9: $\mathcal{G}' = \mathcal{G} \cup p'(e, o')$
- 10: **else if** $s' \in \{o, s\}$ **then**
- 11: $\mathcal{G}' = \mathcal{G} \cup p'(s', e)$
- 12: **return** \mathcal{G}'

4.3.4 Discussion

The proposed strategies are independent of the KGE models. The attacks are solely based on the assumption that KGE models and rule-based approaches to some extent rely on the same influential facts as discussed in Chapter 3. Since the KGE models do not use the training KG explicitly at test time, some of the statistical regularities present in the KGs must be encoded in the embeddings of the KGE models.

In the context of explaining KGE models with the rule-based approach, we are more interested in the deletion attack compared to the addition setting. Both presented procedures rely on selecting a fact from the body grounding (which we term explanation E) of the rule that makes the target prediction. However, for the addition setting, this fact is maintained in the training data. While the selection might also be important in this setting, in the deletion setting, its importance for the KGE model is explicitly estimated by its deletion from the training data.

Furthermore, the deletion fact is selected based on the rule with the highest confidence that predicts the target fact. Therefore, this fact is also the most important fact for the rule-based approach when considering the KGC task. By definition, it is the fact whose removal from the training set (on which the rules are applied) has the strongest negative effect on the predictive performance of the rule-based approach.

To summarize, our deletion method actually deletes the most important fact for the rule-based approach and assumes it is also an important fact for the KGE model. Therefore, evaluating the efficacy of our approach and comparing it to alternative methods allows to investigate the extent to which both model classes rely on the same patterns.

4.4 Experiments

We present experimental details and results in the following sections. Moreover, we argue that the previous evaluation protocol to measure the attack strength should be revisited by showing that the largest part of the MRR degradation is caused by

the protocol instead of the attack. Therefore, we propose a new protocol, re-evaluate the best performing specifications of the related work (Bhardwaj et al., 2021), and compare them to our approach.

4.4.1 Experimental Settings

The general structure of our experiments follows the procedure of the study from Bhardwaj et al. (2021). A KGE model is trained on the training KG and a subset of target facts from the test set for which the model achieved a high filtered MRR is obtained. The attack is performed by deleting or adding the attacking facts calculated by the method described in the previous sections. In the last step, the model is retrained on the perturbed data and the evaluation based on the target facts is repeated.

To have a fair comparison to existing literature, we base the experiments and the comparison to related work on the public implementation of Bhardwaj et al. (2021) and we inherit the respective settings. This also holds for the methods proposed by Zhang et al. (2019). In particular, 100 target facts which had an MRR of 1 in both directions are randomly selected from the test set. We run the experiments five times with newly sampled target facts each time and report average results.

We use the KGE models ComplEx, DistMult, and ConvE and the same datasets as Bhardwaj et al. (2021). That is, we use WN18RR and Fb15k-237. We compare against the best methods studied by Bhardwaj et al. (2021). That is, we include feature-based and gradient-based models (Bhardwaj et al., 2021) and *Direct* attack (Zhang et al., 2019).

We have set the time available for AnyBURL to learn the rule set to 100 seconds. This only has to be done once for each dataset. We use the standard rule confidence (Definition 13) with a smoothing parameter of 5 for all rules. After that the computation of a single deletion and addition attack requires around 0.05 seconds. This is slightly slower than the feature-based methods and faster than the gradient-based methods proposed by Bhardwaj et al. (2021).

4.4.2 Evaluation Protocol

First we will demonstrate in the following, why the existing evaluation protocol (Bhardwaj et al., 2021) leads to misleading results and propose a revised version. Let T be the set of selected target facts. Furthermore, let A be the set of attacking facts, i.e., the selected facts that should be deleted or added to the training graph.

Existing Protocol

The existing protocol selects T by searching for a subset of facts from the test set which achieve a high filtered MRR in both directions. At this point, the full original train, validation, and testing splits are used for filtering which is the standard procedure and was already described in Section 2.2.1. The important aspect is the definition of the filter set after the attack; we focus on the deletion setting in the following: After selecting A , (1) A is *removed* from the filter set and (2) likewise all the test facts from the original test split which are not contained in T are removed from the filter set. The impact of the attack is then measured via the MRR calculated with the adjusted filter set.

The problem with (1) is that it allows for a trivial baseline that simply searches in the training set for a matching fact that has a higher score than a target f and adds

it to A . As it is removed from the filter set, by construction, it will be ranked higher than f eventually degrading the MRR.

The problem with (2) is that it decreases the MRR by simply having a smaller filter set after the attack. These two effects lead to a degradation of the original MRR without a connection to any attacking scheme.

We demonstrate the problems in Table 4.1, where in the middle column we apply the described protocol *without* actually performing an attack, i.e., the model is not retrained on the perturbed data after selecting A . Most of the degradation effect is caused by the protocol and not by the attack.

	Original	No Attack	Attack
MRR	1.0	0.756	0.619
H@1	1.0	0.605	0.475

TABLE 4.1: The degradation in MRR for ComplEx on Fb15k-237 in the *Del* setting when only following the existing protocol without actually performing an attack and without retraining (middle column).

New Protocol

In the new protocol, we maintain filtering in general as it prevents model punishing when true candidates are ranked better than the current query candidates. Therefore, in the deletion setting we do not modify the filter set after training the original model which keeps the MRR on its original value when no attack is performed in contrast to Table 4.1. In the addition setting, we follow the same procedure but we additionally augment the filter set with the facts from A . This is important as otherwise models would be rewarded with overfitting the data in A , ranking these facts higher than the facts in T which would lead to a misleading MRR degradation during test time.

4.4.3 Results

Tables 4.2 and 4.3 show the results for WN18RR and Fb15k-237, respectively. The best (second best) results are marked in bold (underlined). The first (second) part of the tables refers to the gradient (feature)-based similarity metrics proposed by Bhardwaj et al. (2021). The third part contains *Direct* attack (Zhang et al., 2019) and two baselines. In particular, we include a random baseline (Random) that randomly removes a fact from the neighbourhood of the target in the *Del* setting and randomly adds one fact containing one entity of the target in the *Add* setting. Finally, we include a baseline that retrains the models but does not alter the training KG at all (Rerun).

None of the previous state-of-the-art attacks is clearly dominant. When results are averaged over 5 runs and the revised protocol is used, the differences between approaches are smaller than reported in previous work. Our proposed method generates in 17 from 24 settings at least the second best result without having access to the respective model architecture, embeddings, or the loss functions. We achieve the strongest result for ConvE on WN18RR where the MRR degradation is 7.3 (4.9) percentage points higher for Hits@1 (MRR) compared to the second strongest method in the *Add* setting. On the other hand, we achieve the weakest results for ConvE on Fb15k-237 where the reported MRR degradation is 2.9 percentage points lower

Approach	COMPLEX				DISTMULT				CONVE			
	Del		Add		Del		Add		Del		Add	
	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR
GC (cos)	0.193	<u>0.273</u>	0.789	<u>0.887</u>	0.152	0.237	0.794	0.894	0.122	0.174	0.854	<u>0.927</u>
GD (dot)	0.222	0.296	0.794	0.892	0.182	0.259	0.797	0.896	0.261	0.302	0.885	0.942
GL (l_2)	0.208	0.284	0.803	0.894	0.175	0.250	0.796	0.895	0.129	<u>0.175</u>	<u>0.831</u>	0.931
Cos Metric	0.201	0.282	0.813	0.893	0.165	0.246	0.793	0.894	0.981	0.981	0.993	0.997
Dot Metric	0.915	0.928	0.927	0.958	0.904	0.917	0.928	0.963	0.934	0.937	0.966	0.983
l_2 Metric	0.199	0.272	<u>0.788</u>	0.890	<u>0.162</u>	<u>0.243</u>	<u>0.785</u>	<u>0.890</u>	0.937	0.960	0.996	0.998
Direct	0.949	0.958	0.964	0.974	0.965	0.974	0.984	0.989	0.786	0.789	1.0	1.0
Random	0.880	0.890	0.971	0.982	0.906	0.916	0.994	0.995	0.895	0.901	0.999	0.999
Rerun	0.972	0.981	0.976	0.983	0.988	0.992	0.991	0.993	0.995	0.998	0.998	0.999
Ours	<u>0.197</u>	0.278	0.760	0.874	0.167	0.244	0.757	0.876	<u>0.123</u>	<u>0.175</u>	0.758	0.878

TABLE 4.2: Results for WNRR. All results are averages over five runs. Lower is better. The original MRR and H@1 values are 1.0 in all specifications.

than the best performing method. Overall, our results are competitive to recent state-of-the-art.

Please also note the magnitude of the attack strengths in both tables and in particular the comparisons to the *Rerun* baseline. Although this baseline achieves a weaker degradation than the attacks in all cases, the differences are marginal in some settings on the Fb15k-237 dataset (Table 4.3). Contrary to previous work, we therefore conclude that the efficacy of the attack schemes is to a substantial part overshadowed by the effect of re-training the models for this dataset. Remarkably, this changes when looking at the WN18RR results (Table 4.2). For instance, in the *Del* setting, the *Rerun* baseline achieves no degradation whereas the best attack schemes lead to MRR values of around 0.12-0.2. This means that attack efficacy and KGE robustness is highly dataset specific. We will use our approach in the next section to provide an explanation why the attacks have more influence on the WN18RR dataset.

Approach	COMPLEX				DISTMULT				CONVE			
	Del		Add		Del		Add		Del		Add	
	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR
GC (cos)	0.734	0.814	<u>0.776</u>	0.853	0.74	0.823	0.781	0.856	0.720	0.784	0.716	0.805
GD (dot)	0.739	0.818	<u>0.776</u>	<u>0.849</u>	0.74	0.817	0.776	0.848	0.750	0.802	0.703	0.793
GL (l_2)	0.758	0.835	0.785	0.854	0.759	0.835	0.773	0.849	0.743	0.786	0.715	0.806
Cos Metric	0.747	0.829	0.789	0.859	0.730	0.828	0.764	0.845	0.768	0.789	0.714	0.804
Dot Metric	0.806	0.871	0.787	0.864	0.778	0.856	0.813	0.874	0.745	0.825	0.723	0.812
l_2 Metric	0.739	0.825	0.772	0.845	0.743	0.827	0.738	0.828	0.682	<u>0.777</u>	0.719	0.808
Direct	0.738	0.822	0.793	0.859	0.754	0.833	0.782	0.853	0.753	0.806	0.679	0.779
Random	0.810	0.873	0.806	0.863	0.796	0.869	0.800	0.870	0.755	0.818	<u>0.695</u>	<u>0.788</u>
Rerun	0.773	0.853	0.800	0.865	0.795	0.865	0.795	0.870	0.762	0.821	0.739	0.820
Ours	<u>0.735</u>	<u>0.817</u>	0.781	0.856	<u>0.734</u>	<u>0.822</u>	<u>0.763</u>	<u>0.843</u>	<u>0.684</u>	0.775	0.708	0.799

TABLE 4.3: Results for Fb15k-237. All results are averages over five runs. Lower is better. The original MRR and H@1 values are 1.0 in all specifications.

4.4.4 Rule Ablations

In the following, we present an ablation study based on WN18RR and ComplEx. In particular, the attacking methods are unchanged but we use different sets of input rules to calculate the deletions and additions. Table 4.4 shows the results.

In the first row, we allow for the same rules that are also used in the previous chapter when comparing KGE models with the rule-based approach. That is, in contrast to the specifications used for the main results in the previous section, we also allow for \mathbf{U}_d -rules of length one and longer \mathbf{B} -rules. The results are slightly worse in regard to using the rule specifications from the main results. Therefore, the most effective rule types are given by shorter \mathbf{B} -rules and \mathbf{U}_c -rules of length one.

Rules	Del		Add	
	H@1	MRR	H@1	MRR
All	0.210	0.292	0.782	0.889
\mathbf{B} -rules	0.232	0.322	0.785	0.886
\mathbf{B} -rules (11)	0.225	0.289	0.792	0.894
\mathbf{B} -rules (12)	0.970	0.980	0.947	0.968
\mathbf{B} -rules ($1 \geq 3$)	0.968	0.974	0.972	0.983
\mathbf{U}_d -rules (11)	0.803	0.814	0.954	0.960
\mathbf{U}_c -rules (11)	0.512	0.570	0.853	0.923

TABLE 4.4: Ablation results for ComplEx on WN18RR. Lower is better. The original MRR and H@1 values are 1.0 in all specifications. In the parentheses, e.g., *11* refers to a rule length of one.

A relevant question is how the lengths of \mathbf{B} -rules impact KGE models. In the previous chapter, we hypothesized that KGE models cannot express the more complex patterns reflected by longer \mathbf{B} -rules. Therefore, the second row is based on all \mathbf{B} -rules whereas the next three rows split them based on their lengths. There is a strong difference when considering the different subsets. Almost all the effect is based on \mathbf{B} -rules of length one. There is almost no decrease in the MRR and H@1 for the longer types. On the other hand, when considering the predictive performance of the rule-based approach on this benchmark, also longer \mathbf{B} -rules are required to achieve the best results.

4.4.5 Understanding the Explanation

The methods against we compared in the previous section define an attacking fact without providing any data-based explanation of why this fact should be influential for the target fact. In contrast, our method provides the attacking fact, which we term the explanation, and a human-readable rule that links it to the target fact. Additionally, the respective rule confidence can provide further insights into the pattern strength. To demonstrate this, in the following, one representative example based on WN18RR and two examples based on the Fb15k-237 benchmark will be provided.

Example 12. From WN18RR, we select a target fact about the word form *breakable*, its selected explanation E and the underlying rule r with confidence as above:

$$\begin{aligned}
 f: & \text{ relatedForm}(\text{breakable}, \text{break}) \\
 E: & \text{ relatedForm}(\text{break}, \text{breakable}) \\
 r: & \text{ relatedForm}(X, Y) \leftarrow \text{ relatedForm}(Y, X) [0.92]
 \end{aligned}$$

This example uncovers the symmetric behaviour of the relationship *derivationally related form*. A correct prediction can be assigned with a high score by simply predicting the inverse of the explanation fact. Our attacking strategy helped to find

this phenomenon. The example also relates to the ablation study from the previous section where the most efficacy was observed for **B**-rules of length one.

Example 13. From Fb15k-237, we select a target fact f about Cheese, its selected explanation E from the training set, and the underlying rule r with confidence:

$$\begin{aligned} f: & \text{nutrient}(\text{Cheese}, \text{Carbohydrate}) \\ E: & \text{nutrient}(\text{Milk}, \text{Carbohydrate}) \\ r: & \text{nutrient}(\text{Cheese}, X) \leftarrow \text{nutrient}(\text{Milk}, X) [0.86] \end{aligned}$$

The relation $\text{nutrient}(X, Y)$ means that X contains the nutrient Y . The explanation together with the theory in form of a rule provides a complete understanding for predicting the target fact although we might have to apply some background knowledge regarding Milk and Cheese.

The next example also provides insight into why the attack strength is in general smaller on Fb15k-237.

Example 14. From Fb15k-237, we select a target fact f about Miyuki Sawashiro, its selected explanation E from the training set, and the underlying rule r with confidence:

$$\begin{aligned} f: & \text{nationality}(\text{Sawashiro}, \text{Japan}) \\ E: & \text{born}(\text{Sawashiro}, \text{Tokyo}), \text{located}(\text{Tokyo}, \text{Japan}) \\ r: & \text{nationality}(X, Y), \leftarrow \text{born}(X, Z) \wedge \text{located}(Z, Y) [0.76] \end{aligned}$$

Sawashiro is also described by facts in the training set to be a Japanese voice actress and to be living in Tokyo. Both of these facts could likewise be a suitable explanation since there exist rules that allow the prediction of the correct nationality with them. We found these alternative explanations by performing several deletion attacks in succession, while suppressing the deleted facts of previous attacks. For Fb15k-237, we could find many similar examples where the target facts were supported by multiple strong evidences that were equally important for making the target prediction. These observations align with the insights obtained already in Chapter 3 where we hypothesized that multiple rules and facts in the training KG are required to determine a high score for a correct candidate.

We will further investigate the observation discussed in the previous paragraph in the following. In particular, we will present an ablation in which we relax the protocol and allow for the deletion and addition of more than one fact.

4.4.6 Performing Multiple Attacks

Deleting or adding only a single fact is much less effective on Fb15k-237 compared to WN18RR. As demonstrated in the previous section, rules with high confidences can individually allow to rank the correct candidate at the first position on WN18RR. On Fb15k-237, on the other hand, multiple distinct patterns may be used by the KGE models for achieving the highest score for the correct candidate.

Therefore, in Figure 4.1, an ablation experiment is presented in which more than one fact is deleted or added based on the method introduced in Section 4.3. The figure reports the H@1 metric for ComplEx on Fb51k-237 and originates from a re-implementation with the PyClause library.

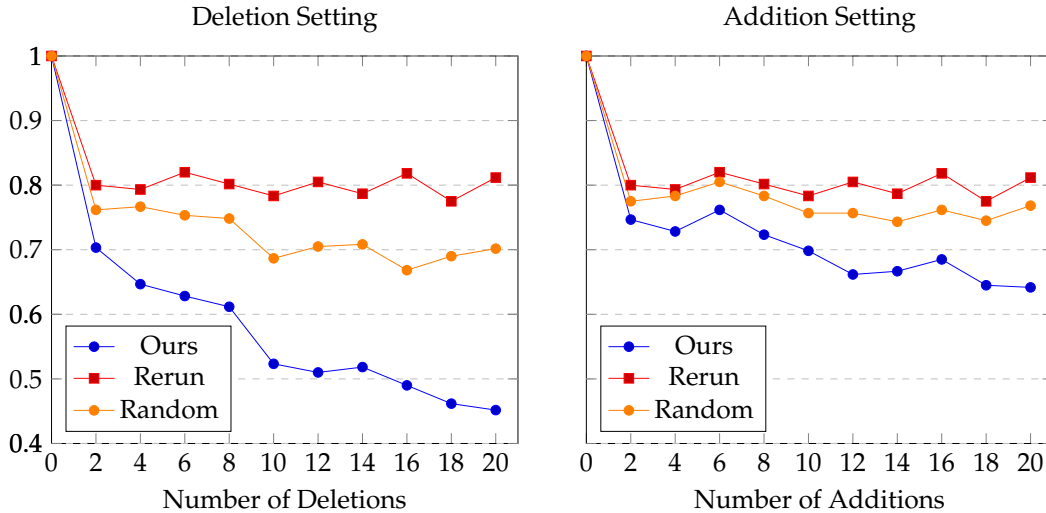


FIGURE 4.1: Results for the decline in H@1 for ComplEx on Fb15k-237 when deleting/adding multiple facts for each target fact.

The attacking procedure is the same as before. After the first fact is deleted or added, the method proceeds with the rule with the second highest confidence. Each mark in the plot is the reported average over three independent runs with new target facts sampled in each run.

We observe the expected decline in the evaluation metric when multiple facts are deleted or added with regard to each target fact. Similar to the previous results, our method has a larger effect in the deletion setting compared to the addition setting. Our method in the deletion setting, however, truly captures the ability of the rules to explain the KGE model predictions. Here, we observe a strong decline when deleting multiple facts compared to the random baseline. In general, the experiments further provide evidence for the efficacy of our method. Nevertheless, the effects are still weaker when compared to WN18RR even when deleting up to 20 facts from the training graph.

4.5 Conclusion

The predictions of a rule-based approach are fully explainable since we can inspect the rule that made a prediction and the facts that ground the body of the rule. If a potential user does not agree with the rule, it can be deleted such that the prediction is prevented. Within this chapter, we showed that the explanatory power of the rule-based approach can also be used to explain the predictions of KGE models.

We presented a rule-based approach for adversarial attacks against KGE models that is inspired by abductive reasoning. We defined the attacking facts based on the body groundings of the rules that predict a target fact. The approach is independent of the respective KGE models and therefore lacks the information that other white-box attacks can use. Nevertheless, it is competitive on the evaluated dataset with other attacking strategies.

On the one hand, the assumption that KGE models and rule-based approaches rely on the same facts for making predictions is overly optimistic. After all, the two approaches are based on fundamentally different paradigms. On the other hand, a model that learned something meaningful from the data, must have learned some traceable patterns. And when two model classes perform relatively well, it is

reasonable to assume that they learned similar patterns. For instance, if everyone in the data who lives in London speaks English, it is reasonable to assume that both the rule-based approach and the KGE models will use this pattern to make predictions about languages. Moreover, we restricted the language bias of the learned rules to only using short rules based on the insights obtained in Chapter 3 and we did not notice improvements when using more complex rule types.

On the WN18RR dataset, the proposed deletion attacks could significantly decrease the performance of the KGE models for most of the evaluated attacking strategies. Therefore, one or two single facts in the training set determine the score of a respective target fact to a large extent. And the rules reliably are able to identify these relevant facts for the KGE models. This observation aligns also with our findings regarding the general predictive performance of the rule-based approach on this dataset discussed in the previous chapter: On WN18RR, Max+ aggregation, which relies on one single rule, is superior to the KGE models under the standard ranking-based protocols.

Contrary to previous work (Bhardwaj et al., 2021), we found that the effects of the attacks are smaller on Fb15k-237. While the rule-based strategy is competitive with the other attacking schemes, the gap between the attacks and the random baseline is smaller compared to WN18RR. When analysing the results, we found that often multiple facts associated with different rules are important for the target predictions. We could confirm this by performing ablations on Fb15k-237 where more than one fact was deleted by the attacking method.

These insights also align with our findings from the previous chapter comparing the rule-based approach with KGE models. In particular, we hypothesized that KGE models might excel in aggregating different evidences into a higher prediction score when compared with the strict procedure of Max-aggregation. Therefore, we will have a closer look at the rule aggregation problem in the next chapter.

Chapter 5

Confidence Aggregation: A Closer Look

Performing logical inference under uncertainty has a rich tradition in the statistical learning literature (Muggleton, 1996; Kersting and De Raedt, 2001; Richardson and Domingos, 2006; De Raedt, Kimmig, and Toivonen, 2007). While the rule-based approach for KGC considered in this thesis falls into a similar scope, there are subtle but important differences when performing inference. For instance, the rule learner AnyBURL exclusively supports the Max+ ranking strategy based on the one-time application of rules for making predictions. However, the best KGE models have been shown to be slightly superior in earlier parts of this thesis when considering the predictive performance. Nevertheless, the previous findings also suggest that the expressiveness of the rule-based approach could be sufficient to achieve the same results. To truly uncover the full potential of the learned rules, we have to deeply explore the formal aspects of the problem beyond the basic definitions.

Context and Goals

On three out of four benchmarks, the respective best-performing KGE model reported a superior predictive performance in the experimental section of Chapter 3 compared with the rule-based approach. The findings of Chapters 3 and 4 further suggest that predictions of KGE models at inference time might be determined by multiple facts in the training KG associated with different regularities. For instance, in the example about Michael Fisher (Section 3.2.4), we found that multiple facts associated with different rules are important for making the correct prediction. In Section 4.4.5 and Example 14 about Miyuki Sawashiro, we found multiple explanations after performing multiple deletion attacks in succession. Additionally, we demonstrated in Section 4.4.6 that deleting multiple facts from the training graph leads to a stronger decline in the predictive performance of the KGE models after retraining.

A potential reason for the inferior predictive performance of the rule-based approach is given by the Max-aggregation function. In contrast to KGE models, it is not able to combine multiple evidences into a higher target score. However, in Chapter 3, the Noisy-or strategy, which takes into account all the predicting rules, was shown to perform inferior. To understand these results, we have to uncover and understand the formal aspects and assumptions of the confidence aggregation problem.

In this chapter we seek to compare different aggregation functions and inference methods, including multi-step reasoning, for the rules on a formal level. Hopefully the obtained insights can help to improve the rule-based approach for the KGC task.

To that end, we will perform a thorough theoretical analysis of the confidence aggregation problem. We will present a probabilistic model that generalizes different aggregation functions. Surprisingly, the probabilistic model allows to provide a full

characterisation of the Max-aggregation function and allows a formal comparison with the Noisy-or product. Furthermore, the derivations will inspire an overlooked baseline that perform slightly superior over the previous functions. The search for a better inference mechanism also opens up the possibility of using existing logical reasoning frameworks for our setting (De Raedt, Kimmig, and Toivonen, 2007; Richardson and Domingos, 2006). Although these approaches do not scale to the benchmarks considered in this thesis - which involve hundreds of thousands of facts and millions of learned rules - they potentially can be employed off-the-shelf. Therefore, we will additionally discuss other logical frameworks and the role of full reasoning in form of logical entailment.

Contribution

The contents of this chapter are based on a workshop paper that later has been published in the work from Betz et al. (2024c). While conducting a theoretical analysis of the confidence aggregation problem, we found that there is a well-defined formal description of the Max-aggregation strategy. In this chapter, we added additional theoretical discussions and experiments. The contributions of the chapter are highlighted in the following:

- (i) We demonstrate that confidence aggregation can be formulated as performing marginal inference in a probabilistic model where an implicit joint distribution over the rules is defined.
- (ii) We prove that under certain assumptions about the joint distribution over rules, probabilistic inference for a target fact boils down to performing Noisy-or aggregation and - more surprisingly - Max-aggregation.
- (iii) Our theoretical framework inspires a simple baseline, Noisy-or top-h, that is slightly superior over the existing methods. Additionally, we discuss alternative logical frameworks and the role of multi-step reasoning and demonstrate that it does not provide benefits for our setting.

In the next section, related work will be discussed. Section 5.2 introduces the probabilistic model and shows how it can be used to describe different confidence aggregation functions. In Section 5.3, we discuss different approaches to perform inference with the rules. Section 5.5 provides evaluation results for the discussed approaches and Section 5.6 concludes.

5.1 Related Work

There exist many branches and approaches to combine logic with uncertainty. For instance, Stochastic Logic Programs (Muggleton, 1996; Sato and Kameya, 1997) and Bayesian Logic Programs (BLP) (Kersting and De Raedt, 2001) augment inductive logic programming (Muggleton and De Raedt, 1994) with probability semantics. In BLPs a joint probability distribution is modelled over the least Herbrand base of the logic program where the ground atoms represent binary random variables. Rules are represented as conditional probabilities and the joint distribution factorizes according to the conditionals as in Bayesian Networks. Noteworthy here the aggregation problem becomes explicit, in particular, when multiple conditionals have the same effect variable they are collapsed into one by the use of a *combining rule*. A difficulty for BLPs is that the probability distribution is only well defined when the underlying

graph does not contain cycles which is quite unlikely in the context of KGC when millions of rules are learned.

Richardson and Domingos (2006) propose Markov Logic Networks (MLNs) to overcome the cycle problem as well as the requirement to define the *ad hoc* combining rule. MLNs define a template language that instantiates a markov network for a given dataset based on the logic program and they subsume many of the approaches from the statistical learning literature. Each possible ground atom is associated with a binary random variable and every possible grounding of every rule with a weight and a binary feature. While the aggregation needs to be stated directly in BLPs, it is only implicit for MLNs and cannot be modelled easily.

The aforementioned approaches are based on logical reasoning with entailment and they also include, e.g., weight learning. Given the more general focus, however, they are not specifically tailored towards the aggregation problem. In BLPs the confidence aggregation problem is revealed by the necessity of combining different conditional distributions and it is not explicitly expressed in MLNs. Moreover, relying on model theoretic entailment is infeasible when datasets are large. For instance, MLNs need to define $|\mathcal{E}|^2 \times |\mathcal{P}|$ random variables for a KG and a feature for every possible grounding of every rule. On FB15k-237, we have $|\mathcal{E}| = 15k$, $|\mathcal{P}| = 237$ and there are 5 million rules learned with AnyBURL.

The closest resemblance to the confidence aggregation problem can be found in Probabilistic Logic Programming (PLP) frameworks such as ProbLog (De Raedt, Kimmig, and Toivonen, 2007). ProbLog is based on distribution semantics (Taisuke, 1995) in which a probability distribution is modelled over logic programs and a target probability is calculated by marginalizing the joint distribution of the target and the logic programs (Riguzzi, 2016). While ProbLog is based on general logical entailment, a closely related branch discusses Lifted Probabilistic Logic Programs (LPLP) (Riguzzi et al., 2017; Nguembang Fadja and Riguzzi, 2019). Here, the clauses of the logic programs are constrained to consist of the same head predicate. Inferring logical entailment under this constraint (while excluding recursive rules) boils down to performing the one-step application of rules and is identical to only considering one-step-entailment.

However, these approaches exclusively treat individual probabilities as being independent. We will discuss in the following sections theoretically, why this is problematic in our setting. Closely related is the empirical result regarding the inferior predictive performance of Noisy-or aggregation presented in Chapter 3. As it will be shown in the following sections, the aggregation function is based on an independence assumptions and performs also inferior to Max-aggregation based on random tie handling.

5.2 Probabilistic Confidence Aggregation

We will start by revisiting the definition for the Noisy-or aggregation method (Definition 18). Let f be a target fact, \mathcal{R} a set of learned rules, and let \mathcal{R}_f denote the rules that predict f with respect to a KG \mathcal{G} . The Noisy-or score is calculated as:

$$\phi_{NO}(f) = 1 - \prod_{r \in \mathcal{R}_f} (1 - \text{conf}(r)) \quad (5.1)$$

The derivations of this section will be independent of what we use for the rule confidences. It was already discussed in Section 2.5.3 that the equation has a well-defined

probabilistic interpretation. If the confidences are substituted with probabilities of binary random variables, the equation calculates the probability that at least one of the variables is true, assuming mutual independence between the variables.

While the Max-aggregation function, on the other hand, does not have such an immediate probabilistic interpretation, we will show in this section that it calculates exactly the same probability as described above. The only difference is that different assumptions are made about the underlying joint distribution of the (rule) variables.

5.2.1 Formal Representation

Let \mathcal{R} be a learned set of rules and let $N = |\mathcal{R}|$. While occasionally we have used rule indices throughout the earlier chapters, we will define this formally for this section. That is, we first enumerate the rules in \mathcal{R} with an index set $I = \{1, \dots, N\} \subseteq \mathbb{N}$ such that $r_i \in \mathcal{R}$ for $i \in I$. Next, we want to represent the rules with binary random variables. Given that the rules in \mathcal{R} are logical objects, we introduce a new symbol for the random variables. Specifically, each rule $r_i \in \mathcal{R}$ is represented by a binary random variable R_i for $i \in I$ which can have values 0 or 1.

We let \mathbf{R} denote the random vector representing all N rules and likewise let $\mathbf{z} = (z_i)_{i \in I} \in \{0, 1\}^N$ denote the corresponding vector of values. For brevity, we write $P(\mathbf{z})$ for $P(\mathbf{R}=\mathbf{z})$, that is, the probability that \mathbf{R} takes value $\mathbf{z} \in \{0, 1\}^N$.

For a target fact f , the rules from \mathcal{R} that predict the target with respect to some KG \mathcal{G} are of particular interest (compare Section 2.5). As defined previously, let $\mathcal{R}_f \subseteq \mathcal{R}$ denote the rules that predict f where the reference to \mathcal{G} is dropped for brevity. Predictions are always made by grounding the rules on the KG on which they are initially learned. The predicting rules \mathcal{R}_f are likewise indexed with a set $I_f \subseteq I$. Therefore, for $j \in I$ and $j \in I_f$, rule r_j predicts target f . On the other hand, for $i \in I$ and $i \notin I_f$, rule r_i does not predict f . Similar as above, we let \mathbf{R}_f denote the vector of random variables representing the predicting rules \mathcal{R}_f . For target fact f , we will overload notation and simply write $P(f)$ for the probability that f is true. Finally, we will let π_i denote marginal probability that R_i takes value one given the KG, i.e., $\pi_i = P(R_i = 1 | \mathcal{G})$ for $i \in I$.

5.2.2 Dealing with Uncertainty

The start of our modelling approach similar to approaches based on distribution semantics (e.g., Riguzzi, 2016; De Raedt, Kimmig, and Toivonen, 2007) where logical clauses are annotated with probabilities. There are two differences: First, we only consider one-step-entailment as inference mechanism (compare Definition 10) opposed to general entailment. Second, we deviate from the common notation by explicitly representing rules with binary random variables. Their Bernoulli success probabilities are the same as the annotated probabilities in distribution semantics. However, the explicit representation with random variables uncovers that a modelling decision has to be made about the implicit joint distribution over the rules and it allows to describe the connection to the confidence aggregation problem.

Let f be a target fact and let \mathcal{R} be some set of rules represented with the random vector \mathbf{R} and indexed by index set I . The set of rules is learned on some KG \mathcal{G} . Finally, let L_I be a mapping that collects all symbolic rules r_i whose corresponding random variables have value one,

$$L_I : \mathbf{R} \mapsto L_I(\mathbf{R}) = \{r_i \in \mathcal{R} : [\mathbf{R}]_i = 1 \text{ and } i \in I\}, \quad (5.2)$$

where $[\mathbf{R}]_i$ refers to accessing the vector at index i which corresponds to random variable R_i . In other words, L collects every symbolic rule that is labelled as correct by the current realisation of its random variable ($R_i=1$). It is important to note that the collected set of rules is independent of the rules that predict target fact f . The connection of both will be made in the next equation in which we define the conditional probability of target f given a KG and the rules:

$$p(f|\mathbf{R},\mathcal{G}) = \begin{cases} 1, & \text{if } L(\mathbf{R}) \cup \mathcal{G} \models_1 t \\ 0, & \text{else} \end{cases} \quad (5.3)$$

A target fact is true (its probability to be true is one), if there exists at least one correct rule that directly predicts the target. Here, *correct* is meant in a distributional sense as above, e.g., $R_i = 1$ for some i , and not in a semantic sense.

If there would be no uncertainty associated with the learned rules \mathcal{R} , we could easily perform rule application with equation (5.3). The mapping $L(\mathbf{R})$ would return the whole set of rules and we only have to evaluate one-step entailment according to its definition.

5.2.3 Inference for Target Facts

Our goal is to calculate the probability that an unknown target fact f is true, given the observed data, i.e., given a KG \mathcal{G} . That is, we seek a probabilistic formulation for the rule-based fact scoring function $\phi(f)$. This target probability is written as $P(f|\mathcal{G})$. Although we have the set of rules \mathcal{R} learned on \mathcal{G} at hand, we cannot directly observe the correctness of the rules. Some of them might describe spurious regularities and there might be interdependencies between them. We therefore treat the rules as latent variables and proceed with a standard approach in which the rules are marginalized out in the formulation of the target probability:

$$P(f|\mathcal{G}) = \sum_{z \in \{0,1\}^N} P(f, \mathbf{R}=z|\mathcal{G}) \quad (5.4)$$

$$= \sum_{z \in \{0,1\}^N} P(f|\mathbf{R}=z, \mathcal{G})P(\mathbf{R}=z|\mathcal{G}) \quad (5.5)$$

In equation (5.5), $P(f|\mathbf{R}, \mathcal{G})$ is set to equation (5.3). This part performs inference with the rules and can simply be calculated for each different realisation z of the random vector. The distribution $P(\mathbf{R}=z|\mathcal{G})$, on the other hand, is more problematic. It defines the joint distribution over all N rules, given the data \mathcal{G} . First, we cannot directly observe outcomes of this distribution in \mathcal{G} as only the correctness of the facts can be observed. Second, it is a multivariate Bernoulli distribution which has 2^N parameters and is therefore not tractable in general. Finally, confidence aggregation is defined exclusively with regard to the rules that predict the target, \mathcal{R}_f , instead of all rules \mathcal{R} .

The last consideration will be mitigated by the following proposition.

Proposition 1. *Let k be the number of predicting rules for target f , i.e., $k = |\mathcal{R}_f|$. Let \mathbf{R}_f be the random vector representing the predicting rules. Under one-step entailment, the target probability can be calculated with respect to only \mathbf{R}_f , i.e.,*

$$P(f|\mathcal{G}) = \sum_{z_f \in \{0,1\}^k} P(f|\mathbf{R}_f=z_f, \mathcal{G})P(\mathbf{R}_f=z_f|\mathcal{G}). \quad (5.6)$$

As before, for the conditional $P(f|\mathbf{R}_f=z_f, \mathcal{G})$, the inference mechanism from equation (5.3) is taken (while only regarding the predicting rules). We prove the equality of the right-hand sides of equations (5.5) and (5.6) in the appendix. The proposition allows to directly focus exclusively on the predicting rules with the marginalized $P(\mathbf{R}_f=z_f|\mathcal{G})$ instead of using the global joint distribution over all rules. The vector of realisations $z_f \in \{0, 1\}^k$ contains only k entries.

While marginal inference can be equally complex, the proposition allows the link to confidence aggregation which only takes into account the k predicting rules. An important observation is that Proposition 1 would not hold if the chosen inference mechanism defined in equation (5.3) would be based on general logical entailment.

Finally, from equation (5.3) and the definition of one-step entailment, we obtain the following proposition that simplifies the target probability even further.

Proposition 2. *Let R_j be the random variable for rule r_j and $I_f \subseteq I$ the indices of the rules that predict target f . The probability $P(\sum_{j \in I_f} R_j \geq 1)$ is the probability that at least one of the predicting rules is true. Under one-step-entailment, it is equal to the target probability $P(f|\mathcal{G})$, i.e., we have that*

$$P(f|\mathcal{G}) = P\left(\sum_{j \in I_f} R_j \geq 1 \mid \mathcal{G}\right). \quad (5.7)$$

Proof. We plug in equation (5.3) for $P(f|\mathbf{R}_f=z_f, \mathcal{G})$ in the sum of equation (5.6). As only the predicting rules are considered, only one term in the sum will be rendered to be zero by $P(f|\mathbf{R}_f=z_f, \mathcal{G})$. In particular, the term in which all realisations are zero. Therefore, we obtain,

$$P(f|\mathcal{G}) = \sum_{\substack{z_f \in \{0,1\}^k \\ z_f \neq \mathbf{0}}} P(\mathbf{R}_f=z_f|\mathcal{G}),$$

where $\mathbf{0}$ denotes the vector containing k zeros. The previous expression sums up all probabilities of all configurations where one or more variable realisations (vector entries of z_f) are one. This is equivalent to the probability that at least one variables is true. \square

The result is independent of any assumption made about the joint distribution over rules. We will continue with an example that summarizes the derivations in the previous sections.

Example 15. Let us assume that there exist four rules learned from a KG \mathcal{G} and that Lisa is predicted by two of them, r_2 and r_3 , to work for Google. We seek to calculate the probability $P(\text{worksFor}(\text{Lisa}, \text{Google})|\mathcal{G})$. Let us further assume that the joint distribution over the rules $P(\mathbf{R}|\mathcal{G}) = P(R_1, R_2, R_3, R_4|\mathcal{G})$ is known. Based on the derivations above, the target probability can be calculated by querying the joint distribution for the probability that at least one of r_2 and r_3 are correct, i.e., $P(\text{worksFor}(\text{Lisa}, \text{Google})|\mathcal{G}) = P(R_2=1, R_3=1|\mathcal{G}) + P(R_2=1, R_3=0|\mathcal{G}) + P(R_2=0, R_3=1|\mathcal{G})$.

In practical terms, unfortunately, the joint distribution is not known. Nevertheless, the example and the previous derivations reveal that we have made a step closer to the connection between confidence aggregation and the probabilistic formulation.

In the introduction of this section, we discussed that Noisy-or aggregation can be interpreted from a probabilistic viewpoint calculating the probability that at least one of the variables (rules) is true. This is identical to the probability that we derived with Proposition 2 and which is shown in Example 15. There are only two

differences. First, Noisy-or is already based on a particular distributional assumption made about $P(\mathbf{R}|\mathcal{G})$ whereas the derivations in this section are general for all possible distributions. Second, the formulation in the Noisy-or product calculates one minus the counter-probability. For instance, based on Example 15, we have that $P(R_2=1, R_3=1|\mathcal{G}) + P(R_2=1, R_3=0|\mathcal{G}) + P(R_2=0, R_3=1|\mathcal{G}) = 1 - P(R_2=0, R_3=0|\mathcal{G})$.

In the next sections, we will introduce concrete assumptions about $P(\mathbf{R}|\mathcal{G})$ and show how these indeed lead to the confidence aggregation functions.

5.2.4 Recovering Aggregation Functions

Probabilistic Max-aggregation

At this point, it is still unclear why Max-aggregation would have a probabilistic interpretation as it calculates the aggregation score by simply taking the maximum rule confidence of the predicting rules. It was introduced in the context of KGC as a computational heuristic (Galárraga et al., 2015) and later it was described as accounting for strong rule dependencies without providing a detailed treatment (Meilicke et al., 2019). Finally it was even described with assuming fact independence (Svatoš et al., 2020). We will now introduce the Fréchet-Hoeffding bound which will help us to achieve a formal derivation. It defines the maximal achievable correlation of two random variables (Joe, 1997).

Let π_i and π_j be the marginal probabilities for two Bernoulli variables, i.e., the probabilities for each variable to be one. It holds for the correlation ρ_{ij} of the variables that $\rho_{ij} \leq U(i, j)$, where $U(i, j)$, is defined as

$$U(i, j) = \min \left\{ \left(\frac{\pi_i(1-\pi_j)}{\pi_j(1-\pi_i)} \right)^{1/2}, \left(\frac{\pi_j(1-\pi_i)}{\pi_i(1-\pi_j)} \right)^{1/2} \right\}. \quad (5.8)$$

Example 16. Let $\pi_1 = 0.64$, $\pi_2 = 0.44$, and $\pi_3 = 0.41$. The maximal achievable correlation for variable 1 and 2 is $U(1, 2) \approx 0.66$. Whereas for 2 and 3, we have that $U(2, 3) \approx 0.94$. Intuitively, the maximal achievable correlation will be higher when the values of the marginals are closer. It will be one if the marginal probabilities are the same.

While the configuration of the marginals in Example 16 allows for non trivial dependencies in regard to the joint distribution, they are not compatible with complete dependence as this would require unit correlation (Lancaster, 1963).

In our viewpoint, the marginals π_i correspond to the individual rule probabilities $P(R_i=1|\mathcal{G})$, given the data. This uncovers a difference to the treatment in distributional semantics, where clauses are assigned with independent probabilities without explicitly modelling a joint distribution while making an implicit mutual independence assumption.

Interestingly, equation (5.8) suffices to specify a joint distribution $P(\mathbf{R}|\mathcal{G})$ such that the inference model for $P(f|\mathcal{G})$ from Section 5.2.3 boils down to taking the max operator over the marginals of the rules that predict the target. This will be described formally with the next result, where all the definitions made previously apply.

Theorem 1. *Let f be a target fact, \mathcal{G} a KG, and let $\pi_j = P(R_j=1|\mathcal{G})$ be the rule marginals for the predicting rules indexed by I_f . Additionally, let $\mathbf{W} \in [-1, 1]^{(N \times N)}$ denote the correlation matrix for the random variables representing all the learned rules. If, for every entry ρ_{ij} of \mathbf{W} , it holds that $\rho_{ij} = U(i, j)$, then a unique distribution for $P(\mathbf{R}|\mathcal{G})$ is induced, such that the*

target probability $P(f|\mathcal{G})$ is equal to the maximum marginal of the predicting rules, i.e.,

$$P(f|\mathcal{G}) = \max\{\pi_j : \{r_j\} \cup \mathcal{G} \models_1 f \text{ and } j \in I_f\} \quad (5.9)$$

In other words, if we assume that the correlation of every pair of rule variables is maximal, then probabilistic inference is the same as taking the maximum marginal of the predicting rules. Equation (5.9) is identical to our definition of Max-aggregation (Definition 16) with the difference that it includes rule marginals π_j instead of the confidences.

In the following, we will briefly show the proof for the case where $k=2$ rules predict the candidate and the general case can be found in the appendix of this thesis. Let $\pi_{\bar{i}} = 1 - \pi_i$ and let, e.g., $\pi_{\bar{i}j} = \pi(R_i=0, R_j=1|KG)$ and likewise for the remaining configurations of the random variables. Additionally, we will use the definition for the correlation of binary random variables:

$$\rho_{ij} = \frac{\pi_{ij} - \pi_i\pi_j}{(\pi_i(1 - \pi_i)\pi_j(1 - \pi_j))^{1/2}} \quad (5.10)$$

Proof (k=2). Let us assume that of all the rules only two rules r_i and r_j predict the target fact f . Following Propositions 1 and 2, $P(f|\mathcal{G})$ is equivalent to querying the joint distribution for the probability that at least one of the rules is true. We can write this probability as $\pi_{\bar{i}j} + \pi_{i\bar{j}} + \pi_{ij}$. We assume the global distribution $p(\mathbf{R}|\mathcal{G})$ exists and is unique. It therefore suffices to show that

$$\max\{\pi_i, \pi_j\} = \pi_{\bar{i}j} + \pi_{i\bar{j}} + \pi_{ij}. \quad (5.11)$$

Assume w.l.o.g. that $\pi_i \geq \pi_j$. Recall that $U(i, j) = \rho_{ij}$ is required by the theorem. Therefore, we set equation (5.8) equal to the left-hand side of equation (5.10) and solve the expression for π_{ij} . We obtain $\pi_{ij} = \pi_j$, however, by the definition of a marginal it holds that $\pi_j = \pi_{ij} + \pi_{\bar{i}j}$. Hence, it must hold that $\pi_{\bar{i}j} = 0$. Rearranging the right-hand side of equation (5.11) leads to

$$\begin{aligned} \max\{\pi_i, \pi_j\} &= \max\{\pi_{\bar{i}j} + \pi_{ij}, \pi_{\bar{i}j} + \pi_{ij}\} \\ &= \max\{\pi_{\bar{i}j} + \pi_{ij}, \pi_{ij}\} \\ &= \pi_{\bar{i}j} + \pi_{ij} \\ &= \pi_{\bar{i}j} + \pi_{i\bar{j}} + \pi_{ij}. \end{aligned} \quad \square$$

Example 17. We use $\pi_1 = 0.64$ and $\pi_2 = 0.44$ from above. We obtain $\pi_{12} = 0.44$, $\pi_{\bar{1}2} = 0$, and $\pi_{1\bar{2}} = \pi_1 - \pi_2 = 0.2$. This leads to $P(f|\mathcal{G}) = 0 \cdot \pi_{\bar{1}2} + 1 \cdot \pi_{12} + 1 \cdot \pi_{1\bar{2}} + 1 \cdot \pi_{\bar{1}2} = 0.64 = \max\{0.64, 0.44\}$.

The theorem allows to specify a unique multivariate Bernoulli distribution $p(\mathbf{R}|\mathcal{G})$ by only defining a correlation matrix. This only holds under the assumption of maximal correlation. In general, specifying the N^2 values of a correlation matrix is not sufficient for defining a distribution that has 2^N parameters and not every correlation matrix is admissible in the first place (e.g., Huber and Marić, 2019). We will show in the appendix of the thesis an example where the full distribution is specified, i.e., we will show how each of the 2^N individual probabilities can be calculated.

With the previous results, a complete explanation for using Max-aggregation in the context of KGC can be provided.

Corollary 1. *Using the Max-aggregation function ϕ_M (Definition 16) is identical to performing probabilistic inference under one-step entailment where (1) rule marginals are approximated with rule confidences, i.e., $\pi_j = \text{conf}(r_j)$, and (2) maximal correlation is assumed between all learned rules.*

The corollary directly follows from Theorem 1 and defines the main result of this section. There exist two observations regarding the derivations. First, the result is independent of the type of confidence that is used for the rule marginals. Second, the result has a purely descriptive nature, that is, it does not provide insights into what a good approximation for the rule marginals might be.

Noisy-or Aggregation

We already discussed the probabilistic nature of Noisy-or aggregation in the introduction of this section. In fact, similar to the treatment of Max-aggregation, it can be derived from the probabilistic inference model $P(f|\mathcal{G})$ presented in Section 5.2.3. The required assumption made about the joint distribution, however, is the strongest assumption that can be made regarding the independence of the variables. In fact, we need to assume mutual independence between all rule variables which even goes beyond pairwise interactions.

Proposition 3. *Let f be a target fact, \mathcal{G} a KG, and let $\pi_j = P(R_j=1|\mathcal{G})$ be the rule marginals. Finally, let I_f be the indexes of the rules that predict the target. If the N rules in $p(\mathbf{R}|\mathcal{G})$ are mutually independent, then the target probability is equal to*

$$p(f|\mathcal{G}) = 1 - \prod_{j \in I_f} (1 - \pi_j). \quad (5.12)$$

It is straightforward to derive equation (5.12) from the inference model. Mutual independence allows a full factorisation of the joint distribution. The equation then follows when writing the target probability as one minus the counter probability as described in Example 15. For completeness, the proof is shown in the appendix of this thesis. The next corollary provides the full explanation for Noisy-or aggregation.

Corollary 2. *Using the Noisy-or aggregation function ϕ_{NO} (Definition 18) is identical to performing probabilistic inference under one-step entailment where (1) rule marginals are approximated with rule confidences, i.e., $\pi_j = \text{conf}(r_j)$, and (2) mutual independence is assumed between all learned rules.*

Similar as above, the result is purely descriptive and independent of the definition of the rule confidence.

5.2.5 Discussion

We have discovered that both aggregation functions make strong assumptions in regard to the dependence structure of the rules. While Noisy-or assumes mutual independence, Max-aggregation assumes maximal correlation. One could describe these as assumptions placed on opposite ends of a dependency spectrum. Naturally, the assumptions can lead to an over- or underestimation of the calculated probability. This can also be observed empirically in the context of KGC. Max-aggregation often assigns low scores to target predictions such as 0.02. Noisy-or scores, on the other hand, are frequently returned as 1.0 due to floating point problems. The independence assumption of Noisy-or, however, is most problematic in settings where large sets of

rules are learned. The noisy learning of rules will automatically create many rules that make predictions for similar reasons. Using Max-aggregation is more robust against these redundancies.

5.2.6 Mixing Assumptions

When both aggregation functions rely on assumptions that are likely to be violated, an intuitive approach is to combine both approaches. A simple compromise between both aggregation functions is obtained when taking the average of the Noisy-or and the Max-aggregation scores. A weighted mixture between two distributions is a well-defined distribution.

Definition 20. (*Average Baseline*) Let ϕ_M be the Max-aggregation scoring function and let ϕ_{NO} denote the Noisy-or scoring function. The average baseline calculates scores according to:

$$\phi_{AVG}(f) = \frac{\phi_M(f) + \phi_{NO}(f)}{2} \quad (5.13)$$

Naturally, the scores will be distributed in between the scores of ϕ_M and ϕ_{NO} . However, as it will be shown in the experimental section, the baseline does not lead to improvement in regard to the predictive performance. We will now present a simple heuristic which was overlooked in the literature and also operates in between Noisy-or and Max-aggregation.

Definition 21. (*Noisy-or top-h*) Let \mathcal{G} be a KG and \mathcal{R} be a learned set of rules. Let $\mathcal{R}_f^{(h)} \subseteq \mathcal{R}$ denote the h rules with the highest confidences of the rules that predict a target f with respect to \mathcal{G} . The Noisy-or top- h aggregation strategy calculates the final score according to

$$\phi_{NO_h}(f) = 1 - \prod_{r_j \in \mathcal{R}_f^{(h)}} (1 - \text{conf}(r_j)). \quad (5.14)$$

Instead of using all predicting rules, only the h rules with the highest confidences are included in the product. It is an open question what the exact description of a global joint distribution and its corresponding dependency assumption is. However, similar to the average baseline, we can make a statement about the scoring behaviour in correspondence to Max and Noisy-or aggregation.

Proposition 4. *The score calculated with Noisy-or top- h is between between Max and Noisy-or aggregation for every possible confidence definition, i.e., $\phi_M(f) \leq \phi_{NO_h}(f) \leq \phi_{NO}(f)$. The equality holds for Max-aggregation for $h = 1$ and it holds for Noisy-or aggregation when $h = k$, where k is the number of rules that predicted target fact f .*

The proposition immediately follows from the definitions of the scoring functions. It underlies that Noisy-or top- h operates in between Max and Noisy-or aggregation. This also leads to more realistic probability computation given that Max-aggregation can lead to an underestimation and Noisy-or aggregation can lead to an overestimation. Moreover, conceptually, Noisy-or top- h it is more efficient to compute compared to Max+ aggregation and Noisy-or. The restriction of only using h rules with the highest confidences allows for a stopping criterion during rule application for every target fact. Finally, instead of setting one value for h we can exploit the mixture property more fine-grained and set the value independently for relations and query-directions.

5.3 ProbLog and Confidence Aggregation

We mentioned already the similarities between reasoning under uncertainty with ProbLog (De Raedt, Kimmig, and Toivonen, 2007) and the confidence aggregation problem. Although it is not feasible from a computational perspective to use ProbLog for our problem setting, conceptually it could be used off-the-shelf to calculate the fact probabilities. As we will see in the following sections, ProbLog is based on a mutual independence assumption and therefore it is closely related to Noisy-or aggregation.

5.3.1 ProbLog and Noisy-or Aggregation

The mutual independence assumption made in Proposition 3 in the previous section reveals the explicit connection of Noisy-or aggregation and ProbLog. It is based on distributional semantics similar to the derivations in the previous sections. By using logical entailment, the target probability is calculated as the probability that the target has a proof (De Raedt, Kimmig, and Toivonen, 2007).

Under ProbLog, the probability for a logic program \mathcal{L} given a ProbLog program \mathcal{T} is defined as

$$P(\mathcal{L}|\mathcal{T}) = \prod_{x_i \in \mathcal{L}} P(x_i) \prod_{x_j \notin \mathcal{L}} (1 - P(x_j)), \quad (5.15)$$

where the ProbLog program \mathcal{T} is a collection of definite clauses and each clause (ground facts included) is assigned with a probability $P(x_i)$. The final probability $P(f|\mathcal{T})$ for a target f is calculated by summing up the probabilities $P(\mathcal{L}'|\mathcal{T})$ of every logic program \mathcal{L}' that entails the target. An example will be shown in the next section.

In our context, the set of learned rules \mathcal{R} together with the KG \mathcal{G} form the logic program \mathcal{L} . Facts from \mathcal{G} have a probability of one and therefore do not alter the probability of the program. The individual clause probabilities in \mathcal{T} are set to the individual rule probabilities π_i from above. Then, the factorization of the product in equation (5.15) is identical to the factorization that we obtain from $P(\mathcal{R}|\mathcal{G})$ after making the mutual independence assumption. However, the logical inference part in ProbLog assumes general entailment opposed to one-step entailment. The explicit relationship is presented in the following result.

Proposition 5. *Let f be a target fact and let all the previously made definitions apply. If clause probabilities are estimated with rule confidences, then the target probability $P(f|\mathcal{T})$ calculated by ProbLog is larger or equal than Noisy-or aggregation, i.e., $P(f|\mathcal{T}) \geq \phi_{NO}(f)$.*

More technical details and the proof are given in the appendix of this thesis. Intuitively, the proof follows from the following observation. For the final target probability, ProbLog sums the probabilities of all programs that entail the target. This includes:

1. The programs that one-step entail \models_1 and entail \models the target.
2. The programs that do not one-step entail but entail the target.

Noisy-or, on the other hand, only sums the probabilities of programs that one-step entail and entail the target which is clearly smaller given that probabilities are non-negative. We will explain the result of Proposition 5 in the next section with an example.

5.3.2 Aggregating Logic Programs

We will demonstrate the differences of Noisy-or aggregation and ProbLog with three rules and a KG with only one fact. In the following paragraphs, it will be described why the target probability calculated by ProbLog cannot be smaller than the Noisy-or aggregation scores.

Example 18. Let us assume the following set of learned rules \mathcal{R} :

$$[0.82] r_1: \text{speaks}(X, \text{english}) \leftarrow \text{lives}(X, \text{london})$$

$$[0.81] r_2: \text{speaks}(X, \text{english}) \leftarrow \text{lives}(X, \text{UK})$$

$$[0.78] r_3: \text{lives}(X, \text{UK}) \leftarrow \text{lives}(X, \text{london})$$

As before, numbers in brackets denote the rule confidences. Note that due to data incompleteness, it is quite common that rules such as the third rule, which describe a regularity that holds true by construction, do not have a confidence of one. Additionally, we are given the following KG \mathcal{G} , consisting of only one fact.

$$\mathcal{G} = \{\text{lives}(\text{marta}, \text{london})\}$$

The goal is to calculate the probability of the target fact $\text{speaks}(\text{marta}, \text{english})$. In Table 5.1, we show all possible programs (subsets of rules) and their corresponding probability according to the factorization made in equation (5.15) when the rule probabilities are set to the confidences. Facts are excluded since their probability is one. For each program, we show if it entails and one-step entails the target.

Program \mathcal{L}	$\mathcal{L} \models f$	$\mathcal{L} \models_1 f$	Probability
$\{\}$	No	No	-
$\{r_1\}$	Yes	Yes	$0.82 \cdot (1 - 0.81) \cdot (1 - 0.79) = 0.032718$
$\{r_2\}$	No	No	-
$\{r_1, r_2\}$	Yes	Yes	$0.82 \cdot 0.81 \cdot (1 - 0.79) = 0.139482$
$\{r_3\}$	No	No	-
$\{r_1, r_3\}$	Yes	Yes	$0.82 \cdot (1 - 0.81) \cdot 0.79 = 0.123082$
$\{r_2, r_3\}$	Yes	No	$(1 - 0.82) \cdot 0.81 \cdot 0.79 = 0.115182$
$\{r_1, r_2, r_3\}$	Yes	Yes	$0.82 \cdot 0.81 \cdot 0.79 = 0.524718$
			$\Sigma \quad 0.935182$

TABLE 5.1: Fine-grained probability calculation with ProbLog for target fact $\text{speaks}(\text{marta}, \text{english})$.

As mentioned, for ProbLog, the target probability is the sum over all programs where the target is entailed. The sum is depicted in the last row of the table. We obtain the exact same result when using the ProbLog online editor to calculate the target probability directly.

The Noisy-or score, on the other hand, is smaller. As only one rule predicts the target it is 0.82. We can obtain this value from the table by summing the probabilities of all programs that one-step entail the target. This procedure sums the probabilities of all configurations where the predicting rule is true (contained in the logic program). As already mentioned, it is the same as one minus the probability that all predicting rules are false. We obtain $0.524718 + 0.123082 + 0.139482 + 0.032718 = 0.82 = \phi_{\text{NO}}(\text{speaks}(\text{marta}, \text{english}))$.

While the previous example shows in a detailed comparison why the ProbLog probability is larger or equal to the Noisy-or score, in the next section, two examples will be described that showcase identical behaviour.

5.3.3 Additional Examples

We modify Example 18 and add the fact *lives(marta, UK)* to the KG. The respective example is shown in Figure 5.1, which depicts the original ProbLog program with the respective notation. The first three rows show the rules and row five to six show the KG. In this case, both the first and second rule directly predict target *speaks(marta, english)*.

```

1 0.82::speaks(X,english) :- lives(X,london).
2 0.81::speaks(X,english) :- lives(X,uk).
3 0.79::lives(X,uk) :- lives(X,london).
4
5 lives(marta,london).
6 lives(marta,uk).
7
8 query(speaks(marta,english)).

```

FIGURE 5.1: ProbLog program based on a modification of Example 18.

The Noisy-or score for the target fact is relatively high with $1 - (1 - 0.82)(1 - 0.81) = 0.9658$. Despite the fact that both rules are closely related, their confidences are aggregated into a higher score. On the other hand, the Max-aggregation score is 0.82. For both aggregation functions, the third rule does not play any role as it does not predict the target.

One might suspect at this point that performing full reasoning can exploit the given relationship between living in London and living in the UK more properly and will result in a more accurate probability. However, the resulting target probability calculated with the ProbLog online editor is exactly the same as calculated by Noisy-or aggregation (0.9658). The third rule does not modify the query probability as *lives(marta, UK)* is already contained in the evidence.

An alternative scenario is given when two rules make the same prediction although one rule is a more general form of the other rule, say, *likes(X,Y) ← friends(X,Y)* and *likes(X,lisa) ← friends(X,lisa)*. If, for example, both rules predict Bernd and Lisa to be friends, there is no mechanism to prevent Noisy-or from aggregating both rule confidences into a higher score. This clearly overestimates the final target probability. We show a corresponding example of a ProbLog program in Figure 5.2.

```

1 0.57::friends(X,Y) :- likes(X,Y).
2 0.78::friends(X,lisa) :- likes(X,lisa).
3
4 likes(bernd,lisa).
5
6 query(friends(bernd,lisa)).

```

FIGURE 5.2: ProbLog program where one rule is the more general form of another rule.

Noisy-or aggregation results in $1 - (1 - 0.57)(1 - 0.78) = 0.9054$. Max-aggregation, on the other hand, results in 0.78. Executing the program with the ProbLog online editor, again, results in exactly the same value that Noisy-or calculated (0.9054).

For the last example, one might argue that the underlying problem is caused by the set of rules and can be mitigated by removing some rules in a preprocessing step. However, we cannot remove either of the two rules from the example. The more general rule (first rule in Figure 5.2) cannot be removed because then predictions for entity pairs not involving Lisa would be lost. Also the more specific rule cannot be removed as it has a higher confidence than the second rule. Removing it would lead to a less precise score/probability estimate for cases in which Lisa is involved.

5.3.4 Discussion

Performing multi-step reasoning by using general entailment can be beneficial in many cases and it is easy to modify the examples given above such that one-step entailment will not suffice to predict a correct answer. However, this type of reasoning excels for smaller-scaled problems with less and possibly handcrafted rules. In the context of the problems in this thesis, the KGs can be large with millions of learned rules which was already shown in Chapter 3. Naturally, a large number of rules leads to rule redundancies and the fact that most existing regularities within the language scope can be found. In many cases it is also possible to express multi-step reasoning with one-step entailment when using longer rules.

The main problem in the context of rule-based KGC is not to make the target prediction in the first place. Instead, the crucial difficulty is given by estimating a reasonable score given all the rules that immediately predict the target. We have discussed already that in these cases, the Noisy-or scoring mechanism will lead to an overestimation of the final score. Empirically, it performs poor on some benchmarks because many similar rules are learned that make predictions based on similar reasons; this is also reflected by the examples made in the previous sections. In many cases, it can even lead to floating point problems where all scores are rendered to be 1.0 (compare Section 5.2.6). Proposition 5 states that the estimated probability by ProbLog will be larger or equal than the Noisy-or scores. Clearly, this aspect has the potential to intensify the issues further. We will nevertheless also show an ablation in the evaluation section in which we use the PyClause library to perform an experiment that is similar to multi-step reasoning.

5.4 Markov Logic and Confidence Aggregation

Similar to ProbLog, we can wrap the rules together with their confidences into a Markov Logic Network (MLN). MLNs (Richardson and Domingos, 2006) use the KG as background knowledge, and can calculate fact probabilities off-the-shelf. If certain rules make the same prediction and the MLN can calculate the fact probabilities, clearly some form of computation that aggregates the individual confidences must be performed. In particular, we will focus in the next sections on the question how an MLN would perform the aggregation in a setting where no multi-step reasoning is involved.

5.4.1 Representation

When representing our setup with MLNs, each rule $r_i \in \mathcal{R}$ is associated with a real-valued weight $\omega_i \in \mathbb{R}$ with $i \in I$. An MLN is then defined as collection of tuples

(r_i, ω_i) for $i \in I$. The MLN forms a *ground* MLN by a) assigning a binary node to every possible (ground) fact that can be formed with the set of entities \mathcal{E} and the relations \mathcal{P} and b) by adding a binary feature for every possible grounding of every rule. In this case, the grounding refers to the ground instance of a rule, including the implication symbol. The feature evaluates to one if a respective grounding is true given the assignments of the fact variables and it evaluates to zero otherwise.

The ground MLN defines a probability distribution over worlds η . A world is an assignment of truth values to all binary variables of the ground MLN. That is, in our case, it is an assignment of truth values to all possible facts with size $|\mathcal{P} \times \mathcal{E} \times \mathcal{E}|$. The probability of a world is given by an exponential formulation:

$$P(\eta; \Omega) = \frac{1}{Z} \exp \left\{ \sum_i \omega_i n_i(\eta) \right\}. \quad (5.16)$$

In the equation, $n_i(\omega)$ is the number of true ground instantiations of r_i in η , i.e., it sums the feature values of all features belonging to the same rule. Furthermore, Z normalizes the distribution, and Ω denotes all rule weights, i.e., $\Omega = \{\omega_i : i \in I\}$.

5.4.2 The Probability of a World

We will continue with an example that shows how to calculate the probability of a world based on equation (5.16). For simplicity, we will use unary predicates for this example.

Example 19. Let $\mathcal{P} = \{\text{smokes}, \text{cancer}, \text{ill}\}$ denote unary predicates for this example. Additionally, let $\mathcal{E} = \{\text{karl}\}$, i.e., there only exists one constant (entity). We are given the two rules where variables are universally quantified.

$$\text{conf}_1: \quad \text{cancer}(X) \leftarrow \text{smokes}(X) \quad (5.17)$$

$$\text{conf}_2: \quad \text{cancer}(X) \leftarrow \text{ill}(X) \quad (5.18)$$

The rules are assigned with confidences and to obtain real-valued weights for the rules, we use the log-odds of the confidences, i.e., $\omega_i = \frac{\text{conf}_i}{1-\text{conf}_i}$ for $i \in \{1, 2\}$. A possible world consists of three elements, in particular, it consists of truth values assigned to the three ground atoms $\text{smokes}(\text{karl})$, $\text{cancer}(\text{karl})$, and $\text{ill}(\text{karl})$.

In the following paragraphs and in the next section, we will show different calculations based on Example 19. Let us assume that we seek to calculate the probability of the world η_1 in which Karl smokes, is ill, and has cancer as depicted in Table 5.2. Each of the three possible facts of a world is represented with one binary variable \mathcal{Y}_j .

Ground Atom	Truth Value	Variable in MLN
$\text{cancer}(\text{karl})$	True	\mathcal{Y}_1
$\text{smokes}(\text{karl})$	True	\mathcal{Y}_2
$\text{ill}(\text{karl})$	True	\mathcal{Y}_3

TABLE 5.2: The possible world η_1 .

We have two rules and each rule has only one ground instantiation. Therefore, the ground MLN consists of only two binary features. One feature regarding the instantiation of the first rule, $\text{cancer}(\text{karl}) \leftarrow \text{smokes}(\text{karl})$. This feature evaluates to one as Karl smokes and has cancer in η_1 . The second feature is regarding to the

instantiation $cancer(karl) \leftarrow ill(karl)$ and it also evaluates to one in η_1 . Therefore, we have $n_1(r_1) = 1$ and $n_2(r_2) = 1$. We can now plug in the values in equation (5.16):

$$P(\eta_1; \{\omega_1, \omega_2\}) = \frac{1}{Z} \exp \{\omega_1 + \omega_2\}. \quad (5.19)$$

The normalization term Z sums the exponentiated values of each of the 2^3 individual worlds. While we can potentially calculate this term, we will see in the next section that it cancels out when we adapt the calculated probability to a setting that matches confidence aggregation.

5.4.3 Incorporating Evidence and KGs

In the previous section, we have shown how to calculate the unconditional probability that Karl smokes, has cancer, and is ill. We have also seen how the rule weights are aggregated by summing them up. However, the normalization constant was still present and the probability calculation does not exactly match our setting in which the KG provides evidence in form of the facts.

What we rather seek to calculate is the probability that, for instance, Karl has cancer given the observation that he smokes and is ill. This resembles the setting of the confidence aggregation problem where both rules from above predict that Karl has cancer. MLNs can easily be used to calculate these conditional probabilities by following simple probability calculus. We will use the variables for the three ground atoms as depicted in Table 5.2. We seek to calculate the conditional probability that Karl has cancer given that he is ill and smokes:

$$P(\mathcal{Y}_1=1 \mid \mathcal{Y}_2=1, \mathcal{Y}_3=1) = \frac{P(\mathcal{Y}_1=1, \mathcal{Y}_2=1, \mathcal{Y}_3=1)}{P(\mathcal{Y}_2=1, \mathcal{Y}_3=1)} \quad (5.20)$$

$$= \frac{P(\mathcal{Y}_1=1, \mathcal{Y}_2=1, \mathcal{Y}_3=1)}{P(\mathcal{Y}_1=1, \mathcal{Y}_2=1, \mathcal{Y}_3=1) + P(\mathcal{Y}_1=0, \mathcal{Y}_2=1, \mathcal{Y}_3=1)} \quad (5.21)$$

Until here, we have only used simple probability rules. Practically, the last expression contains only unconditional probabilities for worlds that we can calculate with the base equation of the MLN. In fact, we have already computed the term that appears in the numerator and also in the denominator with equation (5.19). It is the probability of the world in which Karl smokes, is ill, and has cancer. The remaining term in the denominator is the probability of the world in which Karl does not have cancer but smokes and is ill. In this world, both feature values are zero since the respective ground instantiations of the rules are not satisfied (Note that this does not render the probability of this world to be zero). Therefore, we can plug in the world probabilities into the last expression and obtain:

$$P(\mathcal{Y}_1=1 \mid \mathcal{Y}_2=1, \mathcal{Y}_3=1) = \frac{\exp(\omega_1 + \omega_2)Z^{-1}}{\exp(\omega_1 + \omega_2)Z^{-1} + \exp(0)Z^{-1}} \quad (5.22)$$

$$= \frac{\exp(\omega_1 + \omega_2)}{\exp(\omega_1 + \omega_2) + 1} \quad (5.23)$$

$$= \text{Sigmoid}(\omega_1 + \omega_2) \quad (5.24)$$

The last expression provides an easy understanding of how the MLN performs the aggregation in a setting where both rules from Example 19 predict that Karl has cancer.

In fact, it simply sums up the rule weights, which we defined to be the log-odds of the rule confidences, and subsequently applies the Sigmoid function.

Example 20 (continued). Let us assume that the confidences of the two rules from Example 19 are $conf_1 = 0.9$ and $conf_2 = 0.1$, respectively. Then, we have for the log-odds used as rule weights $\omega_1 = 1.386$ and $\omega_2 = -2.197$. When we plug in the weights of the rules into equation (5.24), we obtain 0.3076 for the final probability that Karl has cancer. This is the exact same result that we obtain when running the example with the Rockit online solver under exact marginal inference (Noessner, Niepert, and Stuckenschmidt, 2013).

Interestingly, the small confidence of 0.1 of the rule $cancer(X) \leftarrow ill(X)$ leads to a decrease in the final probability that Karl has cancer. When considering the behaviour of the Sigmoid function, each rule that has a confidence lower than 0.5 will decrease the predicted probability. One might argue at this point that using log-odds for the rule weights is problematic. However, when there would exist only one rule, using the log-odds for the rule weights leads to the final probability being equal to the rule confidence. This holds because the Sigmoid function is the inverse of the log-odds.

Finally, we will make an example in which we use a given KG and a set of predicting rules.

Example 21. Consider the following rules.

$$conf_1: married(X, Y) \leftarrow engaged(X, Y)$$

$$conf_2: married(X, Y) \leftarrow commonChild(X, Y)$$

$$conf_3: married(X, Y) \leftarrow inLove(X, Y)$$

Further assume the KG $\mathcal{G} = \{inLove(a, b), commonChild(a, b), engaged(a, b)\}$ where a, b are entities. Finally, assume the hypothetical confidences $conf_1 = 0.8$, $conf_2 = 0.7$, and $conf_3 = 0.5$.

The example is typical for the problems that we have seen in regard to confidence aggregation. Based on the gained understanding in this section, we compare the MLN behaviour with the confidence aggregation functions. The target is the probability for the fact $married(a, b)$ and all the three rules predict the fact. If we use ProbLog, we obtain 0.97 which is the same as Noisy-or aggregation as no multi-step reasoning is involved. Using Max-aggregation leads to a final probability of 0.8. When using the log-odds for the confidences, we can again use equation 5.24 or calculate the probability with Rockit. The final target probability is 0.903 for the MLN.

5.4.4 Discussion

In the previous examples, the only type of involved reasoning in regard to inferring the target facts is given by one-step entailment. We established that in these cases, the MLN sums all rule weights and applies the Sigmoid function. Additionally, when a single rule has multiple groundings that lead to the same prediction, each grounding would be individually added in the sum. A rule with a lower confidence than 0.5, which results in log-odds smaller than 0, will decrease the predicted probability. However, this can become problematic when using the log-odds of the rule confidences as rule weights. Suppose that 10 distinct rules with a confidence of 0.4 predict the target fact. In terms of the KGC task, this can be seen as substantial evidence that the target fact is true. However, the MLN would output a probability of ≈ 0.02 .

It has to be mentioned at this point that MLNs also provide a rule parameter learning setting in which potentially better suited weights could be learned. Nevertheless, we can conclude that MLNs are not well suited for the problems of KGC when their rule weights have to be based on the given rule confidences. We will also confirm this empirically by including two baseline experiments. First, we will base the aggregation function on the sum of the log-odds of the rule confidences. Additionally, we evaluate the aggregation function that calculates the sum of the original standard confidences. Further details will be discussed in the next section.

Finally, the discussion about the potential benefits of multi-step reasoning is the same as already given in regard to ProbLog in Section 5.3.4. The difficulty that we face when concerned with the confidence aggregation problem is in most cases not that the fact prediction cannot be made in the first place. Instead, the difficulty is given by the question of how to perform a reasonable aggregation of the confidences of all the rules that immediately predict the target fact.

5.5 Experiments

The goal of the experiments is to provide an empirical analysis of the approaches inspired by the theoretical discussions in the previous sections. We will also include one KGE model as a reference; the comprehensive comparison to KGE models will be continued in the next chapter. As in Chapter 3, we use Fb15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018), and Codex-M (Safavi and Koutra, 2020). We additionally add the KG Yago3-10 (Dettmers et al., 2018) that contains more than a million of facts in the training set.

5.5.1 Experimental Setup

We previously discussed the difference between a Max+ ranking and Max-aggregation (compare Definitions 16 and 17). Max-aggregation resolves ties randomly and Max+ is an extension applied when the Max-aggregation score is the same which orders candidates lexicographically according to the rule confidences. We will evaluate both strategies in the experiments below where we include the Max+ and Noisy-or (NO) results from Chapter 3.

Furthermore, we evaluate the Noisy-or top-h mixture approach (NO top-h) as described in Section 5.2.6. In particular, we investigate a global setting in which we set $h = 5$. In an alternative setting, the best value for h is searched according to the performance of the validation set for relations and directions independently (denoted by NO top- h^*). We search over the values $h \in \{1, 4 \dots 10\}$. Additionally, we include the baselines discussed in Section 5.4.4 that calculate the sum of the confidences and the sum of their log-odds to imitate the behaviour of MLNs. Finally, we include one of the best performing KGE models used in Section 3.4.1. We choose ComplEx since an a trained model is available from the libKGE library for the dataset Yago3-10.

The used sets of rules and settings are equal to Section 3.4.1. For Yago3-10, rules are learned for 3600 seconds with the default AnyBURL parameters. In Table 3.6, the overall number of learned rules learned were shown for the respective benchmarks. The smallest number is observed for WN18RR with roughly 100k rules while on Codex-M, more than 7 millions are learned. General summary statistics for the datasets have been shown in Table 2.3.

5.5.2 Results

Table 5.3 shows the overall results. The first section of the table contains the discussed baselines and ComplEx. The second and third sections show the confidence aggregation functions and the discussed mixtures of Max-aggregation and Noisy-or, respectively.

Baselines

The baselines are motivated by the reasoning discussion regarding MLNs. However, summing the log-odds of the confidences does not perform well. A closer inspection of the rankings reveals that in many cases, a high number of predicting rules with small confidences will lead to a small target score even if some rules with higher confidences predicted the respective target. This effect was discussed in Section 5.4.4 and has a significant effect on the overall predictive performance. Similar behaviour can be expected when using the average of the confidences as an aggregation function.

In contrast, summing the confidence values avoids this problem which is confirmed by the superior predictive performance over all benchmarks. However, it is inferior to all remaining strategies including the Noisy-or results. Summing the confidences will not result in the same candidate rankings compared to those calculated with the Noisy-or product. For example, the Noisy-or product $1-(1-0.1)(1-0.8)$ is larger than $1-(1-0.5)(1-0.5)$ but the sum of $0.1+0.8$ is smaller than $0.5+0.5$. Nevertheless, we will show in the next chapter, how a linear formulation can be derived from the Noisy-or product. However, it is slightly more complicated than calculating the sum of the confidences.

Approach	Fb15k-237			WN18RR			Codex-M			Yago3-10		
	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
ComplEx	0.253	0.536	0.347	0.438	0.547	0.475	0.262	0.476	0.337	0.476	0.682	0.551
Sum logits	0.053	0.107	0.075	0.098	0.156	0.121	0.004	0.012	0.008	0.041	0.082	0.063
Sum confs	0.242	0.496	0.327	0.300	0.523	0.371	0.203	0.420	0.277	0.327	0.612	0.425
Max	0.236	0.496	0.321	0.442	0.561	0.482	0.240	0.443	0.309	0.394	0.640	0.477
Max+	0.246	0.506	0.331	0.457	0.574	0.497	0.249	0.456	0.320	0.498	0.691	0.566
NO	0.251	0.499	0.333	0.391	0.560	0.446	0.219	0.427	0.290	0.367	0.628	0.456
Avg(NO, Max)	0.249	0.511	0.335	0.453	0.570	0.493	0.249	0.464	0.322	0.487	0.690	0.558
NO top-5	0.260	0.524	0.347	0.458	0.578	0.499	0.244	0.468	0.320	0.486	0.697	0.560
NO top- h^*	0.263	0.524	0.349	0.459	0.578	0.499	0.253	0.464	0.326	0.498	0.698	0.568

TABLE 5.3: Results for MRR, H@1, and H@10.

Confidence Aggregation Functions

It was already discovered in Chapter 3 that Max+ aggregation performs superior to Noisy-or. Even when using random tie handling with Max-aggregation, the results are superior. The derivations in the previous sections now allow for a formal explanation: When rules are learned automatically and the resulting sets of rules are large, the maximal correlation assumption is superior to the mutual independence assumption. In general, Max+ aggregation is competitive with ComplEx while being better on WN18RR and worse on Fb15k-237 and Codex-M. Also for the newly included dataset, Yago3-10, the MRR is 1.5 percentage points higher than the result achieved by ComplEx.

Mixtures of Max and Noisy-or

The mixture model that calculates the average between Noisy-or and the Max-aggregation scores does not improve in regard to the evaluation metrics on average. The only exception is Codex-M where it achieves a competitive result.

Noisy-or top-5, on the other hand, performs surprisingly well and it is on average superior to Max+ aggregation. It only falls short on Yago3-10. On Fb15k-237, it is on par with ComplEx achieving 1.6 percentage points improvement over Max+ when considering the MRR. While the average improvement over Max+ is small, Noisy-or top-5 outputs well defined scores that lead to the respective ranking. It can therefore also be used for other tasks such as fact classification whereas Max+ is specifically tailored towards ranking-based tasks. Although the parameter $h = 5$ is set heuristically, we propose to change the default aggregation function for rule-based KGC to Noisy-or top-5 instead of Max+ aggregation.

Not surprisingly, tuning the parameter h on the validation set for relations and directions independently achieves additional improvements. Noisy-or top- h^* outperforms all other simple strategies on average but performs relatively similar to each best performing method, respectively. In regard to ComplEx, it performs significantly better for Yago3-10 and WN18RR, equal on Fb15k-237, and it is inferior on Codex-M.

Summary

In general, it can be observed that the predictive performance of the aggregation methods depends on the respective dataset. Approaches that prioritize the predicting rule with the highest confidence work well on WN18RR and Yago3-10. Aggregating multiple rule confidences into a higher score, on the other hand, is more beneficial for Fb15k-237 and Codex-M. While taking the average of Noisy-or and the Max-aggregation scores does not perform well, Noisy-or top-5 is a compromise between the dependency assumptions that performs competitive over all datasets.

5.5.3 Multi-Step Reasoning

We also tried to run ProbLog on the datasets by transforming the rules and KGs into a ProbLog program as described in Section 5.3. Even on the smallest benchmark, WN18RR, which consists of 86k facts and 6k test queries, ProbLog did not terminate after two hours. After restricting the number of rules from more than 100k to only 400 **B**-rules of length two and reducing the number of test queries from 6k to 20, ProbLog still did not terminate after two hours.

We present the following alternative experiment, based on the PyClause library, that involves multi-step derivations. Let \mathcal{R} be the set of rules and \mathcal{G} the training KG, that is, the KG on which the rules are applied to calculate predictions for the test queries. We select a fraction of rules for each relation based on the highest confidences to obtain a smaller set of rules, termed \mathcal{R}' . Subsequently, we calculate all predicted facts that can be made when applying \mathcal{R}' to \mathcal{G} , i.e., we materialize the rules. Finally, we obtain a larger KG by taking the union of \mathcal{G} and the newly predicted facts. Based on this larger KG and the original set of rules \mathcal{R} , we can perform standard rule application with the aggregation functions for the test queries as in the previous section.¹ By doing so, we perform two reasoning steps instead of only one in the

¹A small intricacy is the filtering procedure in ranking-based evaluation: We ensure that filtering is only performed with the original train, validation and testing sets as described in Section 2.2.1.

standard procedure. If we repeat the procedure of augmenting the KG for multiple steps based on \mathcal{R}' , we obtain a process similar to m-step reasoning.

Rule fraction	2 Steps				3 Steps			
	H@1	H@10	MRR	Num. Facts	H@1	H@10	MRR	Num. Facts
0.1	0.222	0.456	0.300	454k	0.211	0.425	0.283	3.16m
0.2	0.206	0.427	0.280	961k	0.189	0.391	0.256	12.0m
0.3	0.186	0.408	0.260	2.49m	0.175	0.367	0.238	46.1m

TABLE 5.4: Ablation results for multi-step reasoning on Fb15k-237 and Max+.

Table 5.4 shows the results for Fb15k-237 using the Max+ strategy after augmenting the KG. The first column shows the fraction of the original set of rules that is used to augment the KG. For instance, 0.1 means that, for each relation, we use 10 percent of the rules with the highest confidences. We also present the number of facts obtained after augmenting the KG. The values illustrate the complexity of using multi-step reasoning. The original size of Fb15k-237 is 270k. For instance, moving from two steps to three steps and using only 10 percent of the original rules, increases the number of facts in the KG from roughly 450k to more than 3 million. When using 30 percent of the rules and three steps, the final KG contains more than 46 millions facts. Unfortunately, however, the performance of the rule-based approach does not improve.

5.6 Conclusion

In this chapter, we took a close look at different confidence aggregation methods and general approaches to perform inference with rules. We presented a theoretical model for calculating the probability for a target fact to be true. We showed that Max-aggregation and Noisy-or aggregation can be recovered from the model when certain assumptions are made about an underlying joint distribution over the rules. This allows a precise comparison between Noisy-or aggregation and Max-aggregation within the same formal framework: While Max-aggregation assumes maximal correlation, Noisy-or requires a mutual independence assumption. It follows that each of the approaches either under- or overestimates the calculated target probability. Therefore, we presented another baseline aggregation strategy, Noisy-or top-h, which outputs scores that are in between Noisy-or and Max-aggregation. We also discussed other reasoning systems that potentially can be used to calculate fact probabilities with the rules although they are computationally not feasible in our settings. In particular, we contrasted ProbLog (De Raedt, Kimmig, and Toivonen, 2007) with Noisy-or aggregation and we showed how MLNs (Richardson and Domingos, 2006) would perform the confidence aggregation when only one reasoning step is needed. Although these systems can perform multi-step reasoning, we showed that they do not allow for more precise fact probability calculations.

Throughout this and earlier chapters, the rule confidences are pre-computed. Although the theoretical results are independent of the confidence definition, they also hint towards potential deficiencies of the standard confidence computation (Definition 13). The computation for one rule does not take into account the remaining rules. This can lead to a distortion of the confidences. Moreover, the proposed baseline Noisy-or top-h does not fully close the gap between the rule-based approach and the KGE models on the datasets Codex-M and Fb15k-237. Therefore, in the next chapter, we will propose to learn the aggregation functions directly from the data.

Chapter 6

Supervised Confidence Aggregation

Supervised machine learning has made remarkable advancements in the last decade across various fields and problem settings in terms of model fidelity (e.g., Krizhevsky, Sutskever, and Hinton (2012) and Devlin et al. (2019)). The KGE models discussed in this thesis use supervision from the KG to learn suitable embedding representations. During the learning stage, they are trained to calculate high scores for correct facts and low scores for (constructed) negatives (Ruffinelli, Broscheit, and Gemulla, 2020). This approach is also employed for alternative model classes such as GNNs (Zhu et al., 2021). The rule-based approach, on the other hand, applies a different type of learning. Rules are learned from the KG by sampling them first and subsequently estimating their confidences which are used in the pre-defined aggregation functions as discussed in the previous chapters. A natural question is whether this procedure can benefit from the addition of a learning step that is similar to the training of KGE models. Therefore, in this chapter we will explore how supervised learning can be employed to improve the predictive performance of the rule-based approach.

Context and Goals

We established in the previous chapter that Noisy-or aggregation is based on a mutual independence assumption whereas Max-aggregation assumes maximal correlation between rules. These findings are based on the specification of a joint probability distribution over all rules. However, the assumptions are unrealistic and the distribution has $2^{|\mathcal{R}|}$ parameters which cannot be learned from the data due to the latent truth values of the rules. Moreover, the aggregation functions take as input the pre-computed rule confidences which neither are based on the theoretical assumptions nor do they take into account that individual predictions can be made by multiple rules. Although we proposed an aggregation function that constitutes a compromise between the assumptions, the reported improvements are limited.

Therefore, to improve the predictive performance of the rule-based approach, we need to employ alternative techniques that do not rely on the pre-computed rule confidences. The study of Ruffinelli, Broscheit, and Gemulla (2020) showed that the predictive performance of different KGE models is similar and mostly caused by the training objective when used within the same codebase. Thus, we hypothesize that the training objective might also be beneficial for the rule-based approach. Along these lines, in this chapter, we aim to improve the rule-based approach further by borrowing from the training procedures that are used to train KGE models. Our goal is to identify the models and approaches that work best for our setting, while maintaining the interpretability of the rules.

To that end, we propose to cast inference with the rules into a supervised learning problem, that is, to learn the aggregation functions directly from the data. In particular, we will assign learnable rule confidences or latent parameters to the rules and learn

them on the training KG using objectives that are based on fact scores. We will start with demonstrating that the rule confidences can directly be learned from the Noisy-or product. Furthermore, we propose a simple linear discriminative model which generalizes the Noisy-or scoring. Here, we argue that it might be beneficial to - instead of making assumptions that are not realistic - not make assumptions about the rule dependencies in the first place. Furthermore, we will discuss more complex model formulations that allow to incorporate the maximal correlation assumption and are based on latent rule representations and we also explore a transformer-based model (Devlin et al., 2019) as a possible scoring formulation. In the experiment section of this chapter, we also open the scope of alternative methods for KGC and compare our models against a wide variety of models, including GNNs.

Contribution

The contents of this chapter have originally be published in two works. In the first work, we initially proposed to cast confidence aggregation into a supervised learning problem and presented a scoring function inspired by Max-aggregation (Betz, Meilicke, and Stuckenschmidt, 2022b). The dataset construction and the preprocessing codebase was also used in the second work which is a collaboration with my colleague Simon Ott and explores more canonical representations derived from the Noisy-or product (Ott et al., 2023). We had initially performed experiments that were based on learning confidences from the Noisy-or model while he explored a more complicated aggregation method and could find good hyperparameter configurations for a simpler linear scoring function. The contributions of this chapter are:

- (i) We cast the confidence aggregation problem into a supervised learning framework in which rule confidences are learned directly from a labeled dataset.
- (ii) We explore various model formulations for supervised confidence aggregation that maintain the interpretability of the rule-based approach and we explore different training strategies.
- (iii) We improve the predictive performance of the rule-based approach and show that the gap to KGE models is closed when the confidence aggregation is learned.

We will discuss related work in the next section. Subsequently, in Section 6.2, we explore how to learn rule confidences starting from the Noisy-or product. In Section 6.3, a model is proposed that represents rules with latent features and we discuss a transformer-based architecture in Section 6.4. The following sections present the experiments and the last section concludes.

6.1 Related Work

The goal of this chapter is to relax the treatment of having pre-computed rule confidences and to learn the aggregation from the data instead. However, there also exists research that focuses on calculating modifications of the standard confidence formulation. Commonly, the calculated metrics are also referred to as rule measures and these efforts are made in both the context of association rules (Megiddo and Srikant, 1998; Azevedo and Jorge, 2007) and in the relational setting (Galárraga et al., 2013; Toutanova and Chen, 2015; Ho et al., 2018; Zupanc and Davis, 2018).

With respect to the relational setting, the PCA confidence (Galárraga et al., 2013) was already defined in Section 2.3 (Definition 2.5). It restricts the counting of the negative examples in the denominator of the confidence definition and it was further extended in later studies. For example, Toutanova and Chen (2015) propose a modification of the PCA confidence that is based on each head relations cardinality with respect to the complete KG. Here, complete KG refers to the completed KG in which all correct facts exist. This is therefore not applicable to KGC where the core problem is to find these missing facts in the first place.

Zupanc and Davis (2018) propose an adaption based on estimating the number of unknown but correctly predicted facts of a rule. However, these approaches do not consider the relationship between rules and calculate the rule measure of one rule independent of the remaining rules. This is less of a concern in settings where only few rules are learned. For instance, Zupanc and Davis (2018) report on the Yago2 KG (Hoffart et al., 2013), which contains more than 800k facts, that they learned 137 rules. A set of learned rule with that size can never be competitive to other methods like KGE models. To be competitive, large numbers of rules are required which was also demonstrated with Table 3.6. Then, the relationship between the rules and the rule redundancies become a major concern.

On the other hand, there also exist branches that focus on parameter learning in the context of logic programming or in the general relational domain. It was already mentioned in Chapter 5 that MLNs additionally provide a framework for the learning of clause weights. Moreover, structure learning algorithms have been proposed in the context of PLP (Bellodi and Riguzzi, 2015) and LPLP (Bellodi et al., 2014; Riguzzi et al., 2017; Nguembang Fadja and Riguzzi, 2019). For example, LIFTCOVER (Nguembang Fadja and Riguzzi, 2019) learns rule parameters within an expectation maximization framework. We will, however, show that standard gradient-based algorithms are most effective when augmented with certain implementation-based modifications.

In the context of KGC, Alberto Garcia-Duran (2018) proposed KBLRN which is a mixture of experts model. The experts are based on numerical, latent, and relational features. The latent features are taken from the DistMult model and, interestingly, the relational features are based on rules learned from AMIE+. We will compare against the respective result in the experimental section of this chapter.

6.2 Learning from Noisy-or

The core idea of this chapter can be best represented with the Noisy-or scoring function. Instead of using it together with the pre-computed rule confidences, we seek to relearn them based on a training objective.

6.2.1 Motivation

In the previous chapter, we established that the independence assumption in Noisy-or aggregation can cause the final target probability to be overestimated. Here, we will show how this problem arises in conjunction with using pre-computed rule confidences.

Example 22. Consider the following two rules, previously introduced:

$$\begin{aligned} r_1 [0.50]: & \text{ employerOf}(X, Y) \leftarrow \text{industrySector}(X, A), \text{ expertIn}(Y, A) \\ r_2 [0.50]: & \text{ employerOf}(X, Y) \leftarrow \text{industrySector}(X, A), \text{ graduatedIn}(Y, A) \end{aligned}$$

These two rules are quite similar, since a person who graduated in a field is likely also an expert. For this example, additionally suppose that in the training KG, everyone who is an expert in a field also graduated in that field and vice versa. This means that the relations *expertIn* and *graduatedIn* contain exactly the same facts. As a result, both rules make the same predictions when applied on the training KG and have the same confidence which we set for this example to 0.5.

However, when computing the confidences, the redundancy between the rules is ignored. If both rules make a correct prediction, it is counted both for the confidence of the first rule and for the confidence of the second rule. When using the Noisy-or product over both rules, the score becomes $1 - (1 - 0.5)(1 - 0.5) = 0.75$, which is too high since both rules are identical from a data perspective. Further below, we will show that using an objective function can mitigate this problem. In particular, we will see that by setting the rule confidences such that the loss is minimized, we can make the Noisy-or product across both rules equal 0.5 essentially matching the data. Although this is an artificial example, the problem exists also in practice where rules are partly redundant instead of being fully redundant.

6.2.2 Differentiability

Let us revisit the Noisy-or scoring formulation (compare Definition 18) where f is a target fact and \mathcal{R}_f are the rules that predict f with respect to some KG:

$$\phi_{NO}(f) = 1 - \prod_{r \in \mathcal{R}_f} (1 - \text{conf}(r)).$$

The scoring formulation defines a smooth and continuous function with respect to the rule confidences and therefore it can easily be differentiated. If we are able to obtain a dataset of labeled facts with their respective predicting rules, we can cast confidence aggregation into a supervised learning problem.

Therefore, we can define a learnable parameter for each rule and learn them directly from the training KG. The assumptions made about the rules depend on the scoring formulation - in this case, mutual independence (compare Section 5.2.4). However, we can, hopefully, make them align with the given data as close as possible by learning the rule confidences directly.

6.2.3 Base Model

As before, the set of learned rules \mathcal{R} is indexed by an index set $I \subset \mathbb{N}$ with $|I| = |\mathcal{R}| = N$. Let $\omega_i \in \mathbb{R}$ be a learnable parameter for rule r_i with $i \in I$. Rule confidences are in the interval $[0, 1]$ and therefore we define the learnable confidence as $c_i = \sigma(\omega_i)$ where σ is the Sigmoid function. Let f be a fact. For better readability, we let x_i denote a binary feature that is one, if rule r_i predicts target f . That is, we define $x_i = 1$ if $r_i \in \mathcal{R}_f$ and zero otherwise. Then, the fact score is defined as

$$\phi_{NO}(f; \Omega) = 1 - \prod_{i=1}^N (1 - x_i c_i), \quad (6.1)$$

where Ω denotes the set of all learnable parameters. Next, we assume that we are given a label $y_f \in \{0, 1\}$ that describes the truth value of the fact. Every fact that exist in the training KG has label one. The negative facts are obtained using a similar

procedure as in KGE training, i.e., not-existing facts can be assigned with the label zero. An example will be shown below and the detailed approach for labelling the dataset is described in Section 5.

The scoring formulation outputs values between zero and one. Therefore, based on the fact label and the score, the individual fact loss l can be defined. A typical choice is the BCE loss:

$$l(f; \Omega) = -[y_f \log \phi(f; \Omega) + (1 - y_f) \log (1 - \phi(f; \Omega))] \quad (6.2)$$

The overall objective, i.e., the total loss, is the sum or average over all examples. The model can be learned with any gradient-based optimization framework by minimizing the objective with respect to the individual rule parameters. By following this procedure, the rule confidences are learnt under the Noisy-or assumption to fit the data with respect to the chosen loss function.

We will now return to the starting example from Section 6.2.1 and demonstrate, how the objective function can help with the redundancy problem.

Example 23 (continued). Recall the two rules from Example 22, each with confidence 0.5, under the assumption that *expertIn* and *graduatedIn* represent the same facts in the KG. Suppose that these confidences are estimated based only on two examples, i.e., both rules made one correct prediction which exists in the KG and one wrong prediction which does not exist. The task is to relearn the confidences based on these two training examples.

The dataset is shown below (left side of Figure 6.1). Both facts are predicted by both rules and therefore $x_1 = 1 = x_2$ for both rows of the table. One fact is present in the KG (label 1), and the other is absent and therefore assumed to be false (label 0). Note that this configuration of the dataset leads to a calculated rule confidence of 0.5 for both rules.

Dataset			
Facts	x_1	x_2	y_f
f_1	1	1	1
f_2	1	1	0

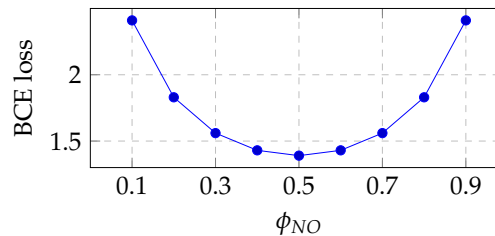


FIGURE 6.1: Left: Example dataset with two training facts and two rules. Right: The total loss as function of the Noisy-or product over both rules (right).

However, not much can be learned from this dataset. The examples are not linearly separable because both have same features but different labels. Intuitively, the best we can do from a data perspective is to predict a probability of 0.5 in a scenario where both rules jointly make a prediction. This intuition is confirmed by the BCE objective. As shown by the loss curve in Figure 6.1, the minimum loss occurs when the Noisy-or product over both rules is 0.5. This value is lower than using the Noisy-or product with the pre-computed confidences which results in 0.75 as discussed above.

Note that the loss does not have a unique minimizer with respect to the individual rule confidences; any combination that produces a Noisy-or score of 0.5 for both facts will be a minimizer. For example, one rule could have confidence 0 and the other 0.5, or both could have confidences of about 0.29.

To summarize, even though this example dataset does not allow us to learn much due to its simplicity, using an objective like the BCE nevertheless can lead to

a better representation of the data. The new confidences avoid overestimating the score, showing that the supervised approach can reduce some of the bias caused by redundant rules.

6.2.4 Generalisation

When inspecting the product in equation (6.1), the same difficulty can be encountered that was already discussed in Section 5.3: The product multiplies many (counter) probabilities and therefore can be computationally instable. If the product renders to zero due to floating point problems, this will translate to the gradient of the loss function and can result in poor optimization behaviour. While we will discuss in Section 6.5 how to mitigate this for directly learning the Noisy-or product, an alternative is to reformulate the scoring function. In fact, by using a monotonic transformation, we can preserve the ordering of the scores while achieving a more stable formulation.

Consider the function $g(z) = -\log(-z + 1)$. The gradient (first derivative) is $\frac{dg}{dz} = \frac{1}{z-1}$. Therefore it is strictly increasing for $z < 1$. That is, applying g to the scoring formulation $\phi(f; \Omega)$ in equation (6.1) performs a monotonic transformation. In particular, when plugging in equation (6.1) into function g , we obtain:

$$-\sum_{i=1}^N \log(1 - x_i c_i). \quad (6.3)$$

The obtained expression is a sum based on the learnable rule confidences that preserves the order of scores calculated by equation (6.1). Now, x_i is either zero or one and therefore the terms in the sum will be either $\log(1 - c_i)$ or $\log(1) = 0$. Therefore, a new parameter $\tilde{\omega}_i$ can be introduced with $\tilde{\omega}_i = -\log(1 - c_i)$. Then, from equation (6.3), we get

$$-\sum_{i=1}^N \log(1 - x_i c_i) = \sum_{i=1}^N \tilde{\omega}_i x_i \quad (6.4)$$

The last expression is the simple sum of the reparameterized learnable rule confidences. This means that for every learned configuration of rule parameters ω_i with $i \in I$, learned from equation (6.1), we can find a sum formulation of the form of equation (6.4) that will result in the same candidate ranking. We have evaluated in the previous chapter the simple sum of the confidences as an aggregation function which performed poor. The previous derivations show why this is not identical to the Noisy-or product. To obtain the behaviour of Noisy-or, we would need to sum $-\log(1 - \text{conf}(r))$ over the predicting rules with the pre-computed confidences.

The expression in equation (6.4) equivalent to the scoring formulation of a logistic regression model without an intercept term. Instead of learning the model given in equation (6.1), we can alternatively learn the weights $\tilde{\omega}$ of the linear model directly which will lead to a more stable learning problem. The final scoring function is defined as:

$$\phi_{LR}(f; \Omega) = \sigma\left(\sum_{i=1}^N \tilde{\omega}_i x_i + \omega_0\right) \quad (6.5)$$

Equation (6.5) is the canonical representation of a discriminative logistic regression model with an additional intercept term. A positive side effect is that this model does not make any assumptions about the dependencies between the features. All the possible assumptions about the relatedness of rules that were discussed so far are not realistic. Therefore, it might be beneficial to simply learn this discriminative mapping from input rules to target scores without considering the rule dependencies. In the following, the set of learnable parameters Ω will be explained precisely.

6.2.5 Disjunct Data

The sum in equation (6.5) ranges over all possible rules from the set of learned rules \mathcal{R} . In order to have an intuitive notation, the binary feature x_i was defined to select the rules that predict the target f . That is, it selects the rules from the set of predicting rules \mathcal{R}_f which was described in the previous chapters. The set of rules for which the feature will be zero, i.e., the not-predicting rules, $\mathcal{R} \setminus \mathcal{R}_f$, is composed of two types of rules. First, the rules that have the same head relation as the target fact f . Second, the rules that have a different relation in the head. In general, by construction, a rule can only predict facts composed of the relation that is also the head relation of the rule. There are two implications that follow from this observation, which will be discussed in the next paragraph.

First, when constructing a dataset with labeled facts, we can view this as constructing $|\mathcal{P}|$ disjunct datasets where each dataset is based on one particular relation. A parameter ω_i will exclusively appear in the dataset that belongs to the relation that is also the head relation of rule r_i for $i \in I$. In other words, the learnable rule parameters ω_i are not shared between different relations. The second implication is that a modelling decision in regard to the intercept term ω_0 has to be made. In particular, we introduce and learn a separate intercept term for each subset, i.e., the intercept terms are learned relation-wise. We have not written this explicitly in equations (6.5) to not clutter notation. To summarize, the set of learnable parameters Ω consists of a rule weight for each rule and an intercept term for each relation.

6.2.6 Expressiveness

For each configuration of parameters ω_i , learned from the Noisy-or model in equation (6.1), we can find a logistic regression formulation that will result in the same ordering of scores, i.e., in the same candidate ranking. To do so, we have to follow the steps of the derivations above and subsequently set all the intercept terms to zero. By construction, the transformed weights $\tilde{\omega}_i = -\log(1 - c_i)$ will be positive as $c_i = \sigma(\omega)$ is between zero and one where $\omega \in \mathbb{R}$ and σ is the Sigmoid function.

The reverse direction, however, is not possible. If we learn the $\tilde{\omega}$ and intercept terms directly from model (6.5), we are not guaranteed to find a formulation that can be expressed by the Noisy-or model in equation (6.1). One reason is the intercept term which cannot be expressed by the Noisy-or model. More importantly, nothing prevents the weights $\tilde{\omega}$ to be negative when learned from the data. In other words, the logistic regression model has a higher expressiveness than the Noisy-or model.

Positive weights resemble the semantics of rule confidences more closely. For example, as we will describe in detail in the experimental section, we observe positive effects when initializing the weights with the positive rule confidences. However, a higher model expressiveness can also be beneficial for supervised learning in general. We will therefore analyse different model configurations. First, we learn the vanilla logistic regression expression with no constraints on the weights (termed LR). Second,

we use a constrained version in which we enforce the learned weights to be positive (termed LR+). While there might be complicated methods to achieve this, the weights are simply squared in equation (6.5).

We have presented three different model configuration in the previous discussions: Learning the rule parameters ω_i in the Noisy-or model, learning the weights $\tilde{\omega}$ in a logistic regression with a positivity constraint, and finally learning the vanilla logistic regression model with no constraint. We also discussed that the supervised problem decomposes into relation-wise sub-problems. To investigate if the models learned different aspects or behave differently over different relations, we can exploit this property and select the best performing model for each relation based on results from the validation set. In the experimental section, this specification will be termed the ensemble model.

6.3 Incorporating the Max Assumption

In the previous section, we discussed how to cast confidence aggregation into a supervised learning setting by parameterizing the scoring function with learnable rule weights. The starting point for the derivations was the Noisy-or aggregation function. However, from Section 5.5 we know that dependency assumptions work well when having large sets of learned rules. Unfortunately, for Max-aggregation, we cannot follow the same procedure as in Section 6.2. There is not enough gradient signal to learn something meaningful when only the predictive rule with the highest confidence is taken into account.

In this section, we will present a modelling approach that has a strong inductive bias and is based on a Max operation over latent rule representations. The core idea is to represent each rule with more than only one parameter. The initial motivation for this approach was to perform Max-aggregation in parallel over different aspects of rule confidences. Therefore, for a given set of rules \mathcal{R} , we let $\omega_i \in \mathbb{R}^d$ denote the latent representation of rule r_i where d is the vector dimensionality. In the case of $d = 1$, we can retrieve the setting from the previous section.

6.3.1 Sparse Aggregation

Let \mathcal{R} be a set of N rules learned on the training KG \mathcal{G} . In the following, we will only consider the set of predicting rules. Therefore, let f be a target fact that is predicted by the rules $\mathcal{R}_f \subseteq \mathcal{R}$ with respect to \mathcal{G} . We set $k = |\mathcal{R}_f|$. The predicting rules are indexed by I_f and we use index j to describe a rule $r_j \in \mathcal{R}_f$.

The vector $\omega_j \in \mathbb{R}^d$ denotes the latent rule representation of predicting rule r_j . The latent rule representations define the learnable model parameters.

We first normalize the latent vectors ω_j with the SoftMax function such that each vector sums to one. Subsequently, we multiply each vector entry with the rule confidence,

$$\omega_j^* := \text{SoftMax}(\omega_j) \cdot \text{conf}(r_j), \quad (6.6)$$

with $\omega_j^* \in [0, 1]^d$. The outcome vector ω_j^* has the same dimensionality as the original latent vector. The idea is to distribute aspects of the rule confidence along the entries of the vector, weighted with its original entries. For example, when we sum up ω_j^* , we obtain the confidence again.

We will now turn to the aggregation part of the model. We stack the k vectors ω_j^* into a matrix $W^* \in [0, 1]^{d \times k}$. The l -th column of the matrix corresponds to ω_l^* . Subsequently, we apply the Max operator over the rules, i.e., each row of matrix W^* is collapsed into one value by taking the maximum element:

$$\mathbf{v} = \text{Max}(W^*), \quad (6.7)$$

The vector $\mathbf{v} \in [0, 1]^d$ is termed the value vector and has the same dimensionality as the latent rule vectors $\omega_j \in \mathbb{R}^d$. It results from the Max-aggregation over the rules. If we set $d = 1$, independent of what is learned for the ω_j , it will result in the standard Max-aggregation score. This will be discussed in more detail below.

Finally, we have to combine the value vector into a final score. While there are many possibilities, we opt for a way that maintains interpretability in the context of confidence aggregation and use the Noisy-or product. Therefore, the final fact score is calculated as,

$$\phi(f; \Omega) := 1 - \prod_{s=1}^d (1 - \mathbf{v}_s), \quad (6.8)$$

where we use \mathbf{v}_s to denote the s -th entry of the vector. The set of model parameters Ω consists of all latent rule representations. The scoring function is not differentiable as the Max operator is used within the calculations. However, in principle, modern automatic differentiation frameworks such as Pytorch (Lerer et al., 2019) or Tensorflow (Abadi et al., 2015) allow the seamless integration into the overall optimization pipeline by providing approximate gradients for the Max operation. We have nevertheless encountered difficulties when learning the model with a standard BCE objective, and therefore we use a custom loss function that will be introduced below.

6.3.2 Properties

Our goal was to obtain a model formulation that can be learned and also takes into account the dependency assumption of Max-aggregation. While intuitively, the use of the Max operator already suggests the connection, we will describe two properties of the scoring function in the following which make the connection explicit.

1) *At most d rules can contribute to the final score.* This follows from the fact that the Max function is taken over rules. Taking the Max of every row of W^* means that every entry of \mathbf{v} is determined by exactly one rule. As there are d rows of the matrix, at most d rules will determine the score. By controlling the vector dimensionality d , which is a hyperparameter of the model, we can make the model align stronger with the Max-aggregation score.

2) *When we set $d=1$, the score is identical to Max-aggregation.* To see this, recall that the SoftMax over only one value always returns one. Therefore, we have that $\omega_j^* = 1 \cdot \text{conf}(r_j)$ and then, $\mathbf{v} = \max\{\text{conf}(r_j) | j \in I_f\}$. Moreover, the Noisy-or product over just one value simply returns that value. Therefore, the final score is $\phi(f) = 1 - (1 - \mathbf{v}) = \mathbf{v} = \max\{\text{conf}(r_j) | j \in I_f\}$.

Like in Max-aggregation, not all the predicting rules contribute to the final score. The model will therefore be termed the *sparse-aggregator* in the evaluation section. The

two properties are independent of the learning of the rule parameters. Potentially, we can also express Noisy-or aggregation with the model, however, this requires to directly enforce a particular structure onto the parameters and the vector dimensionality d would need to be set to the overall number of rules.

6.3.3 Learning the Model

The idea of the model formulation is to express rule redundancies by assigning similar latent features to rules that are partly redundant. In general, the proposed model is not very flexible and more similar to a discrete formulation compared to standard supervised learning models. For instance, the SoftMax normalization restricts the updates of any learning algorithm to be a re-distribution of the rule confidences instead of a clear fitting of the data. However, searching the rule parameters with a discrete search is unfeasible. For instance, if we would restrict the latent rule parameters to have binary values per vector entry, there would be still $2^{|\mathcal{R}| \cdot d}$ possible configurations. Therefore, we need to employ a continuous search in the form of gradient-based optimization as in Section 6.2. However, we observed that, when using the BCE loss, the training resulted in a model performance that was worse compared to just using pre-computed confidences.

Mean-Rank Optimization

Fortunately, there exists recent work in machine learning that bridges the gap between continuous and discrete problems such as combinatorial optimization (Niepert, Minervini, and Franceschi, 2021; Pogančić et al., 2020). It has been demonstrated that these approaches can be applied to ranking-based metrics as these can be written as combinatorial optimization problems (Rolínek et al., 2020).

In Section 6.2.3, we defined the fact loss based on the BCE. Here, we will describe how to use the mean rank (MR), which was defined in Section 2.2.2 (Definition 5), as a loss function. When using the MR, there is no need to explicitly define negative and positive examples. Instead, we will define a fact loss for the head and tail directions separately. In particular, the fact loss is defined as the rank of the current true answer, given a head or a tail query formed from a target fact. The total loss, the MR, is then defined as the mean of all ranks. We will describe the tail direction in the following; the treatment is equivalent for both directions.

Let $f = p(s, o)$ be a fact from the training KG \mathcal{G} . The assumed scoring function $\phi(f; \Omega)$ is as described in equation (6.8). We define the fact loss as the rank of o , i.e., $l(f; \Omega) = rk_t(o|p, s)$ where we use the notation from Definition 5, i.e., t denotes the tail direction. To calculate the rank of the correct candidate (the fact loss), we first have to calculate the candidate scores. That is, we calculate scores with the sparse-aggregator for all $p(s, o')$ for all o' where the rules make the fact prediction. As described in Definition 3, we will refer to candidate scores instead of fact scores. From the sorted candidate scores, we can obtain a vector of ranks where the highest score is assigned with rank 1. Subsequently, to obtain the fact loss, we have to extract the rank of the current true candidate o .

Let ϕ be the vector of candidate scores and let rk be the vector of ranks. Furthermore, let $\text{rank}(\phi)$ be the operation that calculates the vector of ranks from the scores, i.e., $\text{rank}(\phi) = rk$. This operation is the crucial part as it is essentially an index sort which is not differentiable. To calculate the fact loss $l(f; \Omega)$, we simply have to look up the rank of o in rk .

To perform gradient-based optimization, we need to calculate the gradient of the fact loss with respect to the model parameters Ω , that is, we seek to calculate $\frac{dl}{d\Omega}$. From the chain rule, we obtain

$$\frac{dl}{d\Omega} = \frac{d\phi}{d\Omega} \frac{dl}{d\phi}. \quad (6.9)$$

The first term $\frac{d\phi}{d\Omega}$ is the gradient of the scoring function from equation (6.8) with respect to the model parameters, i.e., the latent rule representations. We have already discussed that this gradient can be computed with automatic differentiation frameworks.

The second term, $\frac{dl}{d\phi}$, is the problematic part. It is the gradient of the fact loss (the rank of o) with respect to the vector of scores ϕ . However, the fact loss is based on ranking the scores. That is, we have to push the gradient through the ranking operation $\text{rank}(\phi)$ which is non-differentiable.

Let λ be a hyperparameter. We follow the work of Rolínek et al. (2020), and calculate this gradient as:

$$\frac{dl}{d\phi} = -\frac{1}{\lambda} \left[\text{rank}(\phi) - \text{rank}\left(\phi + \lambda \cdot \frac{dl}{drk}\right) \right]. \quad (6.10)$$

We will explain the gradient calculation in detail in the following. First, it is important to be aware that the desired gradient, $\frac{dl}{d\phi}$, will result in a vector. This vector will contain as many entries as there are individual scores in the vector ϕ and correspondingly ranks in rk .

Now, let us consider the expression $\frac{dl}{drk}$. The fact loss $l(f; \Omega) = rk_t(o|p, s)$ obtained from the vector of all ranks, rk , is a simple lookup. It follows that $\frac{dl}{drk}$ is a vector that is one at the entry of the correct answer, in our case o , and zero otherwise. Therefore, the expression $\text{rank}\left(\phi + \lambda \cdot \frac{dl}{drk}\right)$ perturbs the scoring vector by increasing the score of the true candidate o , and subsequently calculates a new vector of ranks from the scores. The inner expression of equation (6.10) is then the difference in ranks before and after the score of the true answer has been increased.

In practice, in the forward pass, all scores are calculated and the corresponding ranks are extracted. In the backward pass, the score of the correct candidate is increased and the ranks are re-calculated. The gradient is returned as the negative difference of ranks, weighted with $\frac{1}{\lambda}$. Finally, to obtain the gradient of the MR, the mean of all gradients, based on the individual fact losses, is calculated. We continue with an example.

Example 24. Consider the gradient calculation in equation (6.10). Let us assume that the rank of the correct candidate is 50 before the perturbation and 1 after the perturbation. Then, the gradient entry for the correct candidate is $-\frac{49}{\lambda}$. For every candidate that was ranked lower than 50 before the perturbation, the gradient entry is $\frac{1}{\lambda}$. Every other candidate has a gradient entry of zero.

An iteration in gradient-based optimization makes a step towards the negative gradient direction. Thus, as desired, the gradient leads to a parameter update which increases the score of the correct candidate and decreases the scores of the candidates that are ranked lower than the correct candidate.

Gradient Scaling

In Example 24, the calculated gradient entry for the correct candidate is $-\frac{49}{\lambda}$ when it improves from position 50 to 1 after the score perturbation. However, if the correct candidate improves from rank 2 to rank 1, the respective gradient is $-\frac{1}{\lambda}$. While it is common that gradient steps can have different magnitudes, we found empirically that a lower variance was beneficial for the training behaviour. Therefore, we propose a gradient scaling method that is based on the rank of the true candidate and decreases the variance of the gradient magnitudes. Let o be the correct candidate as above. We abbreviate its rank with $rk^{(o)}$ in the following expression. Then, we calculate the scaled gradients as

$$\frac{dl}{d\phi} = -\frac{1}{\lambda(rk^{(o)} - 1)} \left[rk(\phi) - rk(\phi + \lambda \cdot \frac{dl}{drk}) \right]. \quad (6.11)$$

As before, we track the rank of the true candidate in the forward pass. In the backward pass, we scale the gradient with a proportional factor. This procedure ensures a constant signal strength independent of the original position. For instance, in Example 24, we obtain $-\frac{49}{\lambda} \frac{1}{50-1} = -\frac{1}{\lambda}$ for the gradient entry of the correct candidate decreasing its magnitude.

In general, using the MR as a loss function is not limited to the sparse-aggregator. Potentially, it can be used with the other models and it could even be used for training KGE models. We also performed experiments employing MR training for KGE models and for the models of Section 6.2. In these cases, we could not find any improvements regarding the predictive performance. However, the nature of the scoring function of the sparse-aggregator (it cannot fit the data tightly) seemingly works best with this loss formulation. We present an ablation in the appendix of this thesis, where we compare the scaled gradients with the vanilla gradients and we also demonstrate that the MR loss formulation is superior over using BCE for the sparse-aggregator.

6.4 Dense Aggregation

For each of the defined models, including the sparse-aggregator from the previous section, the score contribution of individual rules can be traced easily, which maintains interpretability. In this section, on the other hand, we investigate the trade-off between interpretability and predictive performance. Although the sparse aggregator from Section 6.3 is based on latent rule representations, it cannot fit the data tightly due to the use of rule confidences and the Max operator. The usage of latent rule representations, on the other hand, opens up the space of well-performing language model architectures (Devlin et al., 2019; Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017).

In particular, we define as the scoring function a self-attention architecture based on BERT (Devlin et al., 2019). The scoring function takes as input the latent representations of all rules that predicted a target fact and outputs a real-valued score.

We use the PyTorch implementation of the BERT encoder (Devlin et al., 2019). The latent inputs $\{\omega_1, \dots, \omega_k\}$ of the predicting rules are fed into the encoder which outputs hidden representations with the same dimensionality. For the aggregation, we use simple average pooling on the hidden representations and feed the resulting vector into a fully connected linear layer, which outputs one real-valued score.

For the model input, we sort the latent representations of the rules in descending order according to the rule confidences. We also found that using masked-attention was beneficial for the performance. To be precise, each latent representation ω_i can only attend backwards to the rules with higher confidences up from position $i - 1$ until the first position. Therefore, the rule with the highest confidence will not attend to any other rule. We refer to this model as the *dense*-aggregator because every rule in the input set contributes to the final score. Moreover, it is hardly possible to track individual rule contributions to the final score due to the attention mechanism.

The model is expensive to train and due to runtime considerations, we were unable to extensively explore the hyperparameter space. Although the model is not interpretable, an interesting question is if it learned regularities that cannot be expressed by the remaining models. To investigate this, we also evaluate a joint model of the sparse and dense aggregators. Similar to the ensemble strategy from Section 6.2, we pick the best performing model for each relation and direction based on the validation performance. This allows to find potential differences of the two models. The setting will be termed D+S in the experimental section.

6.5 Experimental Setup

We will evaluate the proposed models and compare them with KGE models, other related work, and the aggregation functions based on pre-computed confidences. First, in this section, we will demonstrate how a labeled dataset can be constructed from the rule predictions on the training KG. Subsequently, we will describe the experimental details and model specifications.

6.5.1 Dataset Construction

For learning the aggregation functions under the BCE loss (compare equation (6.2)), a labeled dataset has to be constructed that collects truth values of the training facts and their corresponding predicting rules. This also requires the construction of negative examples which relates to the assumptions made about the KG presented in Section 2.1.3. It was also discussed already in Section 2.7.2 how negative examples can be constructed for the training of KGE models. However, for the rule-based approach the examples need to be tied to their predicting rules.

The dataset is constructed in a pre-processing step in which the learned rules are applied with respect to the facts of the training KG. That is, the rules are applied on the same KG on which they are originally learnt. For labeling the facts, we use the PCA paradigm (compare Section 2.1.3) separately for the head and the tail direction. We continue with the tail direction; the treatment is equivalent for both directions.

Based on an existing training fact $p(s, o) \in \mathcal{G}$, we calculate all facts $p(s, o')$ that are predicted by the set of rules. For each fact prediction f' , we store the set of predicting rules $\mathcal{R}_{f'}(\mathcal{G})$. Subsequently, we evaluate if the fact is already contained in \mathcal{G} . If it is contained, we label it with one. If it is not contained in \mathcal{G} , according to the PCA, we label it with zero.

While the procedure aligns with the PCA for defining negative facts, it only considers the examples that are predicted by any rules. Therefore, the construction of negatives can also be viewed as the mining of hard examples. There might exist good reasons, given by the predicting rules, why they should be true but they might be false nevertheless. We show pseudocode for the detailed data construction in Algorithm 5 for the tail direction.

Algorithm 5 Dataset construction (tail direction)

Input: Training KG \mathcal{G} , rules \mathcal{R} , entities \mathcal{E} , relations \mathcal{P}
Output: labeled dataset \mathcal{D}

- 1: **procedure** CONSTRUCT DATASET
- 2: $\mathcal{D} = \emptyset$
- 3: **for** each fact $p(s, o) \in \mathcal{G}$ **do**
- 4: *// calculate fact predictions based on query $p(s, ?)$*
- 5: $F' \leftarrow$ calculate all facts $p(s, o')$ with $o' \in \mathcal{E}$
- 6: - such that $\mathcal{G} \cup \mathcal{R} \models_1 p(s, o')$
- 7: **for** each fact prediction $f' = p(s, o')$ in F' **do**
- 8: store the rules $\mathcal{R}_{f'}(\mathcal{G})$ that predict f'
- 9: **if** $f' \in \mathcal{G}$ **then**
- 10: Label $y_{f'} = 1$
- 11: **else**
- 12: Label $y_{f'} = 0$
- 13: Add $(y_{f'}, \mathcal{R}_{f'}(\mathcal{G}))$ to \mathcal{D}

The output of the joint procedure is a labeled dataset where each individual example describes a fact f' , its truth value $y_{f'}$, and its predicting rules $\mathcal{R}_{f'}$. In the model descriptions in Section 6.2, for instance in equation (6.1), we use the binary feature x_i for describing if a rule r_i predicts the respective fact. The feature value is directly determined by $\mathcal{R}_{f'}$, i.e., $x_i = 1$ if $r_i \in \mathcal{R}_{f'}$ and zero otherwise.

The code for constructing the dataset was originally developed for the work (Betz, Meilicke, and Stuckenschmidt, 2022b) and it was also used for the experiments in (Ott et al., 2023). The process, however, can be expensive due to the application of all learned rules with respect to all facts from the training KGs. With the development of the PyClause library, we could reduce the overall runtime significantly. For instance, on the dataset Fb15k-237, the execution time is reduced from more than three hours to less than three minutes (Betz et al., 2024b).

6.5.2 Experimental Setting

We evaluate the models on the datasets Fb15k-237, Codex-M, and WN18RR in detail. For Fb15k-237 and Codex-M, as shown in Section 5.5, the predictive performance of the rule-based approach using pre-computed confidences falls short when compared to the best KGE model and we seek to close this gap. On WN18RR, on the other hand, the rule-based approach is superior. We include this benchmark here as it will be important for the discussion in Chapter 7. Finally, for Codex-L, the rule-based approach was slightly superior to all but the best KGE model. We therefore run our best performing model on this benchmark and add it to the summarizing results.

In regard to the learned set of rules, for all the trained models, we only use **B**-rules and **U_c**-rules. They are learned with the default settings of AnyBURL on each dataset as in the previous chapters. For the latent-based models (Sections 6.3 and 6.4), rules are learned for 3600 seconds. For the Noisy-or and logistic regression specifications (Section 6.2), rules are learned for 1000 seconds. In general, in non-reported experiments, we found that the influence of the rule learning settings is marginal. The evaluation metrics are filtered and all the details are identical to the previous chapters of this thesis.

6.5.3 Model Specifications

In the following, the model specifications and hyperparameters will be introduced. We distinguish between two groups of models: (i) the Noisy-or and logistic regression variants defined in Section 6.2, and (ii) the sparse- and dense-aggregator proposed in Sections 6.3 and 6.4. The latter group will be referred to as latent-based models, as they use multidimensional vector representations for the rules.

Noisy-or and Logistic Regression

Let NO denote the learnable Noisy-or model as specified in equation (6.1). As discussed in Section 6.2.3, the objective based on this model can lead to poor training behaviour due to the product term multiplying many probabilities. In fact, during the training, we observed that the term often is rendered to be zero due to underflow. This leads to gradients that are zero and no learning can occur. To make training possible, we use the PyTorch clamping function and its approximate gradients such that the product term remains larger than zero. Additionally, we have experimented with a form of negative regularization that adds, to each rule, a hypothetical negative example that is predicted by this rule exclusively. We found that this procedure is beneficial for the NO model but not for the logistic regression variants. Therefore, for the NO model, we treat the number of these hypothetical negatives that are generated per rule per training epoch as a hyperparameter, β^- .

Furthermore, we let LR and LR+ denote the logistic regression models without and with positive sign constraint as described in Section 6.2.4. There is a high class imbalance in our labeled datasets. For every correct fact from the training KG, many facts are predicted which do not exist. Additionally, the PCA paradigm can fail and facts might be labeled erroneously as negatives. To mitigate this, we introduce a weighting hyperparameter α^+ that scales the loss of positive examples. Finally, the last hyperparameter used is the learning rate, lr , that determines the step size of the gradient-based updates when optimizing the loss.

Finally, we investigate if the models learn complementary aspects of the data as described in Section 6.2.6. We therefore create an ensemble model combining LR, LR+, and NO. This does not require to retrain the models. We select the best of the three models for every relation and query direction based on the validation MRR without additional computational efforts.

Latent-based Models

For the sparse-aggregator, the hyperparameters that we are concerned with are dropout on the latent features, the latent dimension d , and the learning rate lr . The dense-aggregator follows in its architecture the PyTorch BERT encoder with the modifications as explained in Section 6.4. It uses the same hyperparameters as the sparse-aggregator and additionally, we have to specify the number of attention heads, the number of attention layers, and the feed-forward layer dimensionality.

6.5.4 Training Details

For all the models, early stopping based on the MRR on the validation sets is performed. The models are trained with the PyTorch Adam optimizer. For the latent-based models we report average results over 3 training runs whereas we could not identify any notable variance of Noisy-or and logistic regression models. Different training specifications will be described in the following.

Noisy-or and Logistic Regression

For the NO, LR, and LR+ models, a batch size of 4096 is used and the models are trained under the BCE loss as described in Section 6.2. For each benchmark, we report results under two different training settings:

Hyperparameter search (H): We search for the best specification of hyperparameters for each model on each dataset. For LR and LR+ we search over $a^+ \in \{5, 15, 50, 100\}$. Other settings are fixed, and we use $lr = 0.001$ and maximal 60 training epochs. For NO and the datasets FB15K-237 and WN18RR we search over $lr \in \{0.005, 0.01\}$, $\alpha^+ \in \{5, 30, 100\}$, $\beta^- \in \{10, 30, 60\}$. For Codex-M, $lr \in \{0.005, 0.01, 0.02\}$, $\alpha^+ \in \{1, 5, 10\}$ and $\beta^- \in \{0, 10, 30, 60\}$. The models are trained for a maximum of 50 epochs. Nevertheless, in almost all training runs the best validation MRR is reached in earlier epochs.

Single runs (S): We use the same set of hyperparameters for each model over all benchmarks to investigate the influence of the hyperparameter search. The values are based on the best specification found for Codex-M in the hyperparameter search. Particularly, for LR and LR+, we set α^+ to 5, use a learning rate of 0.001, and a maximum number of epochs of 60. For NO, we set $\alpha^+ = 5$, $\beta^- = 60$, $lr = 0.005$.

Latent-based Models

For the sparse- and dense-aggregators, we started with initial hyperparameter configurations inspired by best practices in the machine learning community. We then narrowed the search space by manually inspecting validation results of different configurations.

The sparse-aggregator is trained on the MR as described in Section 6.3.3. On Fb15k-237, we set $d = 40$, dropout = 0.4, and $lr = 0.02$. For WNRR, we set $d = 50$, dropout = 0.4 and $lr = 0.03$. For Codex-M, we set $d = 40$, dropout = 0.4 and $lr = 0.02$. For all the experiments, we use a value of 5 for λ when training on the MR.

The dense-aggregator from Section 6.4 is trained on the CE loss. We found the same set of hyperparameters resulting the in best validation results for all datasets. In particular, we we use the same set of hyperparameters, $d = 56$, dropout=0.15, and $lr = 0.005$. Finally, we use 4 attention heads and 4 attention layers throughout all the experiments. The dimensionality of the feed-forward layers within the attention layers is 256.

6.6 Experimental Results

After presenting a summarizing figure, the results will be discussed in two tables. In Table 6.1, we will continue the comparison of rule-based approaches and KGE models. Additionally, we will discuss the benefits of learning the aggregation compared to using pre-computed confidences presented in earlier chapters. We also put particular emphasis on the comparison to Max+ aggregation, which was considered the state-of-the-art strategy at the start of our research period. In Table 6.2, we include more model classes including other rule-based approaches, differentiable rule learning, and GNNs. We will start by summarizing the results and the comparison of the rule-based approach with KGE models in the next section.

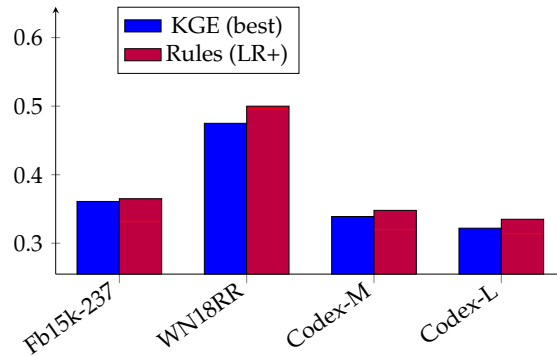


FIGURE 6.2: The best KGE model compared with the learned aggregation. The red lines depict Max+ aggregation results.

6.6.1 KGE Models vs. Learned Aggregation

Figure 6.2 shows a summary of the results when comparing the learned rule-based approach to KGE models. To ensure a fair comparison, we pick the best KGE model on each dataset and compare it with the best approach for learning the aggregation (LR+ with hyperparameter search). For Codex-L, due to the large size of the benchmark, we use the hyperparameters from Codex-M. We show the final test MRRs on each benchmark and the red lines depict the Max+ aggregation results. The gap between the best KGE model specification and the rule-based approach can be closed, when the aggregation functions are learned. The MRR result on Codex-L is 0.335; detailed comparisons for the remaining benchmarks will be discussed below.

Table 6.1 shows the detailed results. In the first table section, the KGE model results from Chapter 3 are shown. The next sections of the table depict the aggregation functions with pre-computed confidences as discussed in Chapter 5, the latent-based models from Sections 6.3 and 6.4, and the LR and NO models from Section 6.2, respectively.

The best rule-based specification, LR+ (H), achieves a higher H@1, H@10, and MRR on all benchmarks when compared to any of the KGE models. When considering the NO and LR models with optimized hyperparameters (last section of the table), every specification reports higher results than the KGE models on Fb25k-237 except for NO which is on par with RESCAL and Hitter*. The results for Codex-M are similar, although here also the NO model outperforms the KGE models.

When comparing the KGE models with the sparse model, only RESCAL and Hitter* report superior results on Fb15k-237 and Codex-M. The dense model, on the other hand, performs worse. All the learned aggregation functions are superior to the KGE models on WN18RR. However, on this benchmark, the improvements over using pre-computed confidences are marginal which will be discussed in the following sections.

To summarize, learning the aggregation functions improves the rule-based approach and performs at least on par with classical KGE models on the benchmarks on which using pre-computed confidences is inferior.

6.6.2 Benefits of Learning the Aggregation

Next, we will highlight the most interesting results when comparing the different approaches proposed in this chapter to using pre-computed confidences. We will exclude the dense-aggregator from this section; it will be included in Section 6.6.4.

When comparing the learned aggregation functions to Max+ aggregation, the sparse-aggregator achieves approximately two percentage points improvement on Fb15k-237 and Codex-M. When trained with a single hyperparameter configuration, the NO and LR models achieve from approximately two to three percentage points improvement on Fb15k-237 and approximately 2.5 percentage points improvement on Codex-M. When tuning the hyperparameters of these specifications, further benefits can be observed (last section of the table). For example, the LR+ model achieves an MRR of 0.365 on Fb15k-237 and 0.348 on Codex-M whereas Max+ aggregation achieves 0.331.

The ensemble specification discussed in Section 6.2.6 achieves only a small improvement over the best performing LR model (last row of the table). This suggests that the models learned relatively similar confidences. The slightly weaker results of the NO model are likely based on inferior training dynamics as discussed in Section 6.2. Additionally, the sparse model is mostly dominated by the LR and NO models. Nevertheless, representing the rules with latent features allows potentially also for additional feature modalities such as textual entity descriptions (Kochsiek et al., 2023).

In Chapter 5, it was already discussed that the Noisy-or top-h specifications can improve slightly over Max+ aggregation. Naturally, the improvement of the learned aggregation functions over the Noisy-or top-h specifications is smaller. The best performing configuration, LR+, still performs more than 1.5 percentage points higher than the best Noisy-or top-h specification on Fb15k-237 and more than 2 percentage points higher on Codex-M.

For the WN18RR benchmark, on the other hand, there is only a marginal improvement when comparing Max+ aggregation with the learned aggregation functions. However, on this dataset, the rule-based approach in general has a strong predictive performance when compared, for instance, with the KGE model results. Nevertheless, the results on WN18RR will be discussed in more detail below and also in Chapter 7.

6.6.3 Other Works

The comparison with other model classes is provided in Table 6.2. We compare against the differentiable rule-learning approaches DRUM (Sadeghian et al., 2019) and Neural LP (Yang, Yang, and Cohen, 2017). Furthermore, we include the approaches KBLRN (Alberto Garcia-Duran, 2018), GPFL (Gu, Guan, and Missier, 2020), RuleN (Meilicke et al., 2018), RLvLR (Omran, Wang, and Wang, 2018), SAFRAN (Ott, Meilicke, and Samwald, 2021), and CNN (Ferré, 2020). Finally, we include the GNN-based models RGCN (Schlichtkrull et al., 2018), NBFNet (Zhu et al., 2021), and A*Net (Zhu et al., 2024).

From the previously mentioned approaches, the only model for which publicly available results exist for Codex-M is SAFRAN. It achieves results for H@1, H@10, and the MRR of 0.253, 0.449 and 0.320, respectively. These results are higher than Max+ aggregation but inferior to each model specification in regard to the learned aggregation functions.

In general, the learned aggregation functions are mostly superior to most other works except for two of the GNNs. For instance, the differentiable rule learner DRUM achieves a better result than Max+ aggregation on Fb15k-237, reporting an MRR of 0.343, but it is inferior to Max+ aggregation on WN18RR (0.486 vs. 0.497). When compared to the learned aggregation functions, it is inferior on all specifications.

		Fb15k-237			WN18RR			Codex-M		
Approach		H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
KGE	RESCAL	0.263	0.541	0.357	0.439	0.521	0.468	0.244	0.456	0.317
	TransE	0.221	0.497	0.313	0.053	0.526	0.227	0.223	0.454	0.303
	DistMult	0.250	0.531	0.343	0.413	0.531	0.452	-	-	-
	ComplEx	0.253	0.534	0.348	0.438	0.543	0.477	0.262	0.476	0.337
	ConvE	0.248	0.521	0.339	0.411	0.508	0.447	0.239	0.464	0.318
	TuckER	0.259	0.536	0.352	0.430	0.514	0.459	0.259	0.458	0.328
	HittER*	0.268	0.549	0.361	0.437	0.531	0.469	0.262	0.486	0.339
Section 5.5	Max	0.236	0.496	0.321	0.442	0.561	0.482	0.240	0.443	0.309
	Max+	0.246	0.506	0.331	0.457	0.574	0.497	0.249	0.456	0.320
	Noisy-or	0.251	0.499	0.333	0.391	0.560	0.446	0.219	0.427	0.290
	NO top-5	0.260	0.524	0.347	0.458	0.578	0.499	0.244	0.468	0.320
	NO top- h^*	0.263	0.524	0.349	0.459	0.578	0.499	0.253	0.464	0.326
Sections 6.3 + 6.4	Sparse	0.266	0.526	0.352	0.459	0.574	0.499	0.266	0.467	0.335
	Dense	0.245	0.510	0.335	<u>0.466</u>	<u>0.587</u>	<u>0.507</u>	0.261	0.465	0.331
	D+S	0.267	0.527	0.354	0.469	0.593	0.511	0.273	0.476	0.342
Section 6.2	NO (S)	0.262	0.523	0.350	0.451	0.576	0.494	0.275	0.476	0.344
	LR (S)	0.264	0.507	0.347	0.459	0.581	0.501	0.267	0.472	0.337
	LR+ (S)	0.273	0.532	0.359	0.459	0.576	0.500	<u>0.278</u>	0.476	<u>0.348</u>
	NO (H)	0.268	0.535	0.357	0.453	0.575	0.496	0.277	<u>0.478</u>	0.346
	LR (H)	0.275	0.532	0.362	0.457	0.581	0.500	0.268	0.477	0.339
	LR+ (H)	<u>0.279</u>	<u>0.538</u>	<u>0.365</u>	0.458	0.575	0.500	<u>0.278</u>	0.477	<u>0.348</u>
	Ensemble	0.283	0.541	0.368	0.459	0.584	0.503	0.280	0.480	0.349

TABLE 6.1: MRR, Hits@1, Hits@10 results for Fb15k-237, WN18RR, and Codex-M.

6.6.4 WN18RR and GNNs

There are notable differences when comparing the results of the WN18RR benchmark with the other datasets. For WN18RR, using pre-computed rule confidences and Max+ aggregation is sufficient to outperform the KGE models. However, learning the aggregation functions does not provide substantial benefits.

Learning on WN18RR

The only noteworthy improvement over Max+ aggregation, reported in Table 6.2, is given by the dense-aggregator and the combination with the sparse-aggregator (D+S). While being inferior on Codex-M and Fb15k-237, the specifications achieve the highest results on WN18RR. Although the improvement is small, the question arises if the model learned patterns that cannot be expressed by the other models.

Potentially, a transformer architecture can express complex dependencies between the latent rule representations due to the self-attention mechanism. When we originally performed these experiments, we also performed extensive debugging efforts to analyse the model behaviour. While we were not able to answer the question if the model learned some particular regularities, we observed that the model predicted high negative values when prompted with certain randomly selected rule inputs. At this point, we were uncertain about the interpretation of these results. Moreover, the logistic regression formulation, without the sign constraint (which was

		Fb15k-237			WN18RR		
Approach		H@1	H@10	MRR	H@1	H@10	MRR
Other	DRUM	0.255	0.516	0.343	0.425	0.586	0.486
	Neural LP	-	0.362	0.240	0.371	0.566	0.435
	KBLRN	0.220	0.482	0.306	-	-	-
	GPFL	0.247	0.504	0.322	0.449	0.552	0.480
	RuleN	0.182	0.420	-	0.427	0.536	-
	RLvLR	-	0.393	0.240	-	-	-
	CNN	0.222	0.446	0.296	0.444	0.519	0.469
	SAFRAN	0.269	0.513	0.351	0.459	0.578	0.502
	RGCN	0.182	0.456	0.273	0.345	0.494	0.402
	NBFNet	0.321	0.599	0.415	0.497	0.666	0.551
	A*Net	0.321	<u>0.586</u>	<u>0.411</u>	<u>0.495</u>	<u>0.659</u>	<u>0.549</u>
Section 5.5	Max	0.236	0.496	0.321	0.442	0.561	0.482
	Max+	0.246	0.506	0.331	0.457	0.572	0.497
	Noisy-or	0.247	0.494	0.329	0.391	0.559	0.446
	NO top-5	0.260	0.524	0.347	0.458	0.578	0.499
	NO top- l^*	0.263	0.524	0.349	0.459	0.578	0.499
Sections 6.3 + 6.4	Sparse	0.266	0.526	0.352	0.459	0.574	0.499
	Dense	0.245	0.510	0.335	0.466	0.587	0.507
	D+S	0.267	0.527	0.354	0.469	0.593	0.511
Section 6.2	NO (S)	0.262	0.523	0.350	0.451	0.576	0.494
	LR (S)	0.264	0.507	0.347	0.459	0.581	0.501
	LR+ (S)	0.273	0.532	0.359	0.459	0.576	0.500
	NO (H)	0.268	0.535	0.357	0.453	0.575	0.496
	LR (H)	0.275	0.532	0.362	0.457	0.581	0.500
	LR+ (H)	0.279	0.538	0.365	0.458	0.575	0.500
	Ensemble	<u>0.283</u>	0.541	0.368	0.459	0.584	0.503

TABLE 6.2: MRR, Hits@1, Hits@10 results for Fb15k-237 and WN18RR.

published later), can also potentially learn negative rule weights and does not achieve substantial improvements on WN18RR.

Predictive Performance of GNNs

The fact that learning the aggregation function does not improve the rule-based approach on WN18RR may suggest an upper-bound of what is possible on this dataset. Therefore, the results of the GNNs are even more surprising. In particular, NBFNet and its successor A*Net improve over the rule-based approach by more than five percentage points, achieving an MRR of 0.55. Also, on Fb15k-237, the models are state-of-the-art and achieve an MRR of more than 0.41.

In the earlier years of our research, the general perception was that GNN-based approaches were inferior when used for KGC (compare, for instance, the results of RGCN in Table 6.2). The publication of the NBFNet model in the work of Zhu et al. (2021) questioned this perception. In general, these models are not interpretable and expensive to train. However, the raw performance improvements are interesting and highly relevant for the discussion of the expressiveness of rule-based approaches and

KGE models. Naturally, the question arises about what the GNNs learned from the data and why it is not captured by the rule-based approach. These questions will be discussed in more detail in Chapter 7.

6.7 Conclusion

Using pre-computed rule confidences with the standard aggregation functions can be problematic. First, the assumptions made by the aggregation functions are too restrictive. Second, the confidence of a rule is calculated independently of all the remaining rules. If two rules make predictions for mostly the same reasons and their confidences are calculated in isolation, their values may be inflated. Moreover, pre-computing the confidences does not align with the respective assumptions made by the aggregation functions. Therefore, in this chapter, we explored different approaches to learn rule confidences or rule parameters directly from the data using discriminative modelling.

The simplest extension of Noisy-or with pre-computed confidences is to relearn them on the data by training the models to provide high scores (or probabilities) for correct facts and low scores for negative ones. Thus, we presented a method to construct a labeled dataset based on making rule predictions on the training KG on which they were originally learnt. We discussed extensions of the Noisy-or model by generalizing the scoring formulation into linear scoring functions and investigated different settings such as constraining the learned parameters to be positive.

We also demonstrated how to formulate and effectively train more complex model formulations based on latent rule representations that can take into account dependency assumptions with a Max operator. We showed how these models can be trained effectively on the MR while using custom-designed gradients. Although simpler formulations excel in terms of predictive performance, using latent rule representation can allow for an interface for multi-modal (KGC) that takes into account other data types such as textual inputs or even images (Zhang et al., 2024).

The findings of Chapters 3 and 4 suggested that the learned set of rules should potentially be sufficient to achieve the predictive performance of the best KGE models on all benchmarks. Indeed, our empirical results show that the rule-based approach, when the aggregation is learned, performs comparably or marginally better than even the best KGE models.

Nevertheless, on one dataset, WN18RR, only a more complex model based on an ensemble including a transformer-based architecture (Devlin et al., 2019) has shown improvement over Max+ aggregation and Noisy-or top-5. Interestingly, especially on this dataset, the GNN architectures NBFNet (Zhu et al., 2021) and A*Net (Zhu et al., 2024), which we introduced in Section 2.8, reported state-of-the-art results. We will therefore seek to answer in the next chapter what these models have learned from the data that is not captured by the rule-based approach.

Chapter 7

Comparisons Beyond KGE Models

One research goal of this thesis is to establish the strengths and to identify potential limitations of the rule-based approach. Along these lines, the predictive performance of the best KGE models was matched by the learned aggregation functions in the previous chapter. However, to fully uncover the capabilities of the rule-based approach, we have to look beyond the comparison with simple embedding models. In particular, the GNN architectures NBFNet (Zhu et al., 2021) and A*Net (Zhu et al., 2024) showed to be superior for the completion task in the previous chapter. The architectures were introduced in Section 2.8. Interestingly, they share properties with both the rule-based approach and KGE models. Like a rule-based approach uses the training KG to make predictions for facts from the test set, GNNs perform message passing over the training graph for calculating fact scores at inference time. KGE models only rely on the learned embeddings and do not utilize the training KG at inference time. On the other hand, the GNNs, like KGE models, rely on latent representations for encoding the elements of the KG in contrast to the symbolic representations of the rule-based approach. Nevertheless, to improve our understanding of the rule-based approach, we need to identify what the GNNs learned from the data and if it can be expressed with the rules.

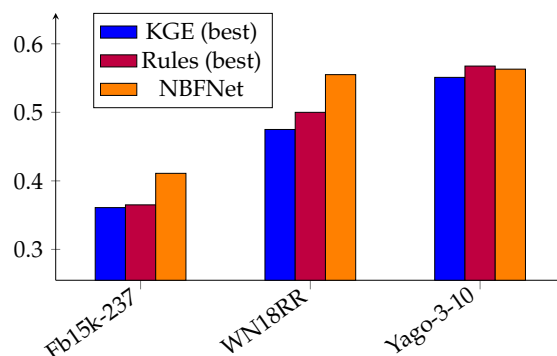


FIGURE 7.1: Summarized MRR results.

Context and Goals

The results regarding the GNNs and the rule-based approach from the previous chapter are summarized in Figure 7.1 where Yago3-10 is also included to provide a complete picture. Apart from Yago, the GNN reports substantial improvements over the rule-based approach. As mentioned in Section 6.6.4, these results are somewhat surprising, especially for WN18RR: On this benchmark, learning the aggregation provides only little benefit over using pre-computed rule confidences which is already superior to the KGE models.

If we want to further improve the rule-based approach or discover its natural limitations, we need to understand these differences in the predictive performance. Therefore, in this chapter, we seek to answer the question if the GNNs learned patterns that cannot be expressed with the rules or if they simply excel in aggregating patterns that potentially could also be expressed by the rules.

To that end, we will present a variety of different analyses in the following sections regarding the benchmarks Fb15k-237 and WN18RR to truly understand the performance differences. The experiments are structured into three parts for each benchmark as described in the following paragraph.

First, we introduce synthetic datasets that allow by their simplicity to identify structural differences between the model classes. For one dataset, we will show that it can be solved easily by the GNNs but not by the KGE models or the rule-based approach. For the other dataset, we show how the GNNs drastically change their behaviour, when a particular hyperparameter setting is used. Second, we perform perturbation-based experiments with respect to the GNNs on the real benchmarks. These experiments are inspired by the protocol that was used in Chapter 3. In particular, we investigate the score changes of the GNNs before and after a small perturbation of the training graph. The perturbations are used to activate or deactivate certain patterns and can therefore help to identify if the GNNs have learned these patterns. Finally, based on the findings, we discuss methods of how to imitate the observed GNN behaviour with the rule-based approach.

Contribution

The results from this chapter originate from experiments performed with the goal of completing the research questions of this thesis. As we perceived the findings to be particularly useful for the KG community, the contents of this chapter have been pre-released in an Arxiv paper in form of a technical report (Betz et al., 2024a). The contributions of this chapter will be summarized in the following.

We find that approximately half of the performance difference between the GNNs and the rule-based approach can be explained by one simple negative pattern on each dataset which cannot be expressed with the rules. A negative pattern is a regularity that, when activated, leads to a score decrease of a certain candidate. In particular:

- (i) On WN18RR, the exploitation of an exclusion rule, derived from the particular structure of 1-to-N relations, significantly boosts the performance of the GNNs.
- (ii) For Fb15k-237, the particular benchmark construction induced a structural bias disallowing connections between entity pairs at inference time when they are already connected in the training graph. Despite the pattern being unrealistic, it can seemingly be exploited by the GNNs when trained under a particular hyperparameter setting. When the setting is turned off, however, a significant decrease in predictive performance is observed.
- (iii) The predictive performance of the (supervised) rule-based approach can be significantly improved when the negative patterns are taken into account. Either by augmenting the model formulation with existential features (WN18RR) or by a post-hoc procedure that enforces compliance with the negative pattern (Fb15k-237).

These findings add a novel perspective to the understanding of the differences in predictive performance between different models that has not been discussed so far

in this thesis: A model might rank a correct candidate prediction with the best rank because it learned to penalize the scores of the false candidates instead of providing a high score for the correct candidate.

The remainder of the chapter is structured as follows. The next section will introduce a synthetic dataset that contains artificial patterns relevant for WN18RR. Subsequently, in Section 7.2, we discuss the GNNs based on experiments on WN18RR and propose augmentations for the rule-based approach. Section 7.3 will introduce a synthetic dataset for Fb15k-237 and Section 7.4 will discuss the GNN performance on this benchmark. Finally, the last section concludes.

7.1 The Synthetic Zoo Dataset

We will start with a synthetic dataset that will be relevant for the discussion of the WN18RR benchmark. Figure 7.2 shows the Zoo KG, which only contains two relations and describes students who visit the zoo. They walk in a chain-like structure such that each student has one follower and follows one other student. The dataset that we use in the synthetic experiments described below is exactly as shown in the figure with the exception that we use 100 student nodes and the last student in the chain follows the first student in the chain.

Each student visits the zoo (dashed relation) such that message passing over the whole graph is possible. The fact *follows(bobby, anna)* is removed and taken as an evaluation fact, creating a visual gap in the graph. After the training phase on the KG, a model has to propose candidate answers based on the head query *follows(?, anna)* and the tail query *follows(bobby, ?)* formed from the evaluation fact.

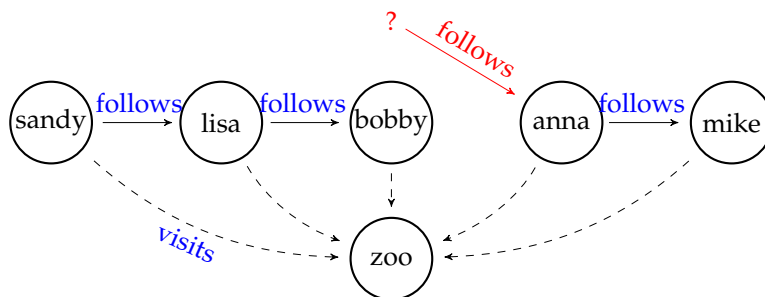


FIGURE 7.2: Graphical representation of the Zoo dataset. The task is to determine who follows the node *anna*. The correct answer *bobby* seems to be most intuitive as it visually closes the gap of the graph. However, it is less obvious which specific pattern in the data might support this conclusion and can be learned by a KGC model.

7.1.1 Experiments

We train different models on the dataset and subsequently compare them by the MRR based on the evaluation fact. Due to the symmetry of the dataset, we will only focus on the head direction in the following. The discussion is equivalent for the tail direction.

The head MRR is based on the single question "Who follows Anna?" given by the query *follows(?, anna)*. It is calculated from the ranking position of *bobby* as $\frac{1}{\text{rank}(\text{bobby})}$ where the best result of one is achieved when *bobby* is ranked at the first position.

We learn rules with AnyBURL and use the LR model without positive parameter constraint which was introduced in equation (6.5) in Section 6.2. The dataset construction for training the model follows the procedure explained in Section 6.5.1. We compare the rule-based approach with NBFNet and A*Net and we additionally include ComplEx. During training, no validation set is used and the qualitative results do not depend on different hyperparameter configurations. We report averages over 10 independent training runs for all models.

	Approach	MRR Tail	MRR Head	MRR Joint
	Rules (LR)	0.03 ± 0.02	0.02 ± 0.02	0.03 ± 0.01
	ComplEx	0.02 ± 0.01	0.02 ± 0.02	0.02 ± 0.02
ROH off	NBFNet	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	A*Net	0.93 ± 0.20	1.00 ± 0.00	0.97 ± 0.10
ROH on	NBFNet	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	A*Net	1.00 ± 0.00	0.95 ± 0.15	0.98 ± 0.08
	LR + \exists .-feat.	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00

TABLE 7.1: MRR results for the zoo dataset. We report averages (\pm std. dev.) over 10 runs. Random ranking in both directions results in an MRR of approximately 0.02.

7.1.2 Results

Table 7.1 shows the results. The first two rows describe the rule-based approach and ComplEx, respectively. Rows three to six contain the GNN results. The specifications are identical except for the binary hyperparameter *remove-one-hop* (ROH) which does not influence the qualitative results on this dataset and will be discussed in Section 7.3. Likewise, the last row will be discussed in Section 7.2.5.

The rule-based approach and the KGE model are not able to learn anything meaningful for answering the evaluation query. Randomly ranking all nodes would result in an MRR of $\frac{1}{50.5} \approx 0.02$ as there are 100 students plus the *zoo* node. In fact, the only sensitive behaviour that these two model classes show is never scoring the *zoo* node with the highest likelihood. The GNNs, on the other hand, easily learn to rank Bobby at the top position where only A*Net exhibits some noise. However, a fine-grained analysis, in which we calculate the MRR after every training epoch, reveals that an MRR (both for head and tail) of one is reached in almost every epoch also for A*Net. In contrast, the rule-based approach and KGE model never achieve anything higher than reported under various hyperparameter settings.

Why should Bobby be the Follower of Anna?

Bobby is an intuitive answer for the question of who follows Anna: It visually closes the gap in the training graph. However, it is not straightforward to formulate which evidence the GNNs might have been used to confidently score Bobby higher than all other student nodes. When inspecting which nodes typically are followers, we find that everyone who visits the Zoo could be a follower. Additionally, everyone who is followed by somebody, is a potential follower themselves. These observations

can help to discriminate Bobby from the zoo node (the zoo probably does not follow Anna) but it does not discriminate Bobby from any of the remaining students. Bobby differs from the other students by the fact that he does not follow anyone. As mentioned, the first student follows the last student in our synthetic dataset. However, this stems from the selection of the evaluation fact in the first place. We cannot draw any conclusion as the data does not contain another student that resembles Bobby in this regard.

Negative Patterns

The previous discussion shows that there does not exist positive evidence in the graph (in form of facts) that helps to score Bobby with a higher likelihood than the other student nodes. However, the behaviour of the GNNs is quite significant suggesting they learned a non-random regularity. To find an explanation, we have to consider which facts are not present in the graph. For instance, Sandy follows Lisa but she does not follow Bobby, Anna or any of the other students. Similar statements can be made for all the students (except of Bobby) as each student only follows one other student. From this regularity, we can formulate a negative inference rule:

If a student follows someone already, they do not follow someone else.

We will formalize this inference rule in Section 7.2 and demonstrate its relevance for WN18RR. There is not a single case in the data that violates the rule (Bobby does not follow anyone in the data). In fact, we can easily exploit the pattern and create a scoring mechanism that replicates the behaviour of the GNNs at inference time for the target query. We assign every student node (not the *zoo* node) a random score from $[0, \epsilon]$, for $\epsilon \in \mathbb{R}_+$, and subtract $\epsilon + 1$ whenever a node already follows another node in the training graph. In the end, Bobby will have assigned the highest score. Remarkably, Bobby is not ranked at the first position because he is *ranked up* due to positive evidence, but because the other nodes are *ranked down*.

7.1.3 Discussion

Due to the simplicity of the dataset, there is no other pattern that helps to discriminate all the other students from Bobby for the target query except of the fact that they do already follow some other node than Anna. Therefore, we conjecture that the GNNs exploited this pattern to achieve the performance results. Indeed, when we modify the dataset and let Bobby follow some other random student except of Anna and subsequently re-train the GNNs, then the achieved MRR collapses.

Intuitively, GNNs can exploit the negative pattern as message passing allows a node at inference time to be informed via the follows relation from its successor that it already follows them. We can view this as a kind of lookup in the training data. Interestingly, although a connection between the source node Anna and the candidate node is needed, the pattern is independent of their particular joint path representation.

The negative pattern will be made explicit in the training data as training involves the construction of negative examples via perturbing the head or tail slot of existing facts (compare Section 2.7.2). For the rule-based approach, on the other hand, we can be certain that the pattern cannot be exploited. The language bias is exactly defined by the allowed rule syntax which does not include negative rules. Also the ComplEx

model shows the same behaviour. A KGE model does not have access to the training data (in contrast to the GNNs) at inference time, which makes it hard to exploit the negative pattern.

In database terms, the *follows* relationship is, viewed from head-to-tail, a N-to-1 relationship¹ which means that each head entity is associated with exactly one but not more tail entities. In particular, the previous discussion showed how a model might be able to exploit this by the use of a negative pattern formulation which will be discussed in more detail in the next section.

7.2 Negative Patterns on WN18RR

On the WN18RR test set, the GNNs achieve a joint MRR that is approximately five percentage points higher compared to the rule-based approach (compare Figure 7.1). We have identified a structural difference between the model classes in the previous section already. The GNNs are able to correctly rank Bobby on top by penalizing the scores of all remaining entities.

We will start with a more fine-grained analysis regarding the performance differences of the model classes on the WN18RR dataset. Subsequently, we will introduce the Only-One-Tail (OOT) rule that generalises the regularity that we observed for the Zoo KG.

7.2.1 Fine-Grained Model Comparison

We re-trained the GNNs with the implementation from Zhu et al. (2024) while using the original configuration files. The models are evaluated with the evaluation code of the PyClause library. There is no noticeable difference to the originally reported results. For the rule-based approach, we use the LR model from Section 6.2.6. Results are shown in Table 7.2 where we show the head (H) and tail (T) MRR calculated separately. In the first two rows, results over all relations are shown and in the last column the head results for the relation *hypernym* is shown.

Approach	All Relations		<i>hypernym</i>
	MRR (T)	MRR (H)	MRR (H)
Rules (LR)	0.530	0.471	0.129
NBFNet	0.561	0.541	0.274
A*Net	0.559	0.543	0.280

TABLE 7.2: MRR head (H) and tail (T) results for WN18RR.

The difference in predictive performance for the head direction is more than twice as large as for the tail direction. We will therefore have a closer look at the head direction. Figure 7.3 shows the relative performance differences (NBFNet is one) of the three approaches for the relations with the most facts in the test set for the head MRRs. WN18RR consists of 11 relations and from the 3134 test facts, 1251 belong to the

¹Due to the symmetry of the dataset, we even have that N=1 and the relationship is 1-to-1. However that does not affect the results of this section. If we add additional nodes that randomly follow exactly one other node the MRR Head results are the same for the GNNs.

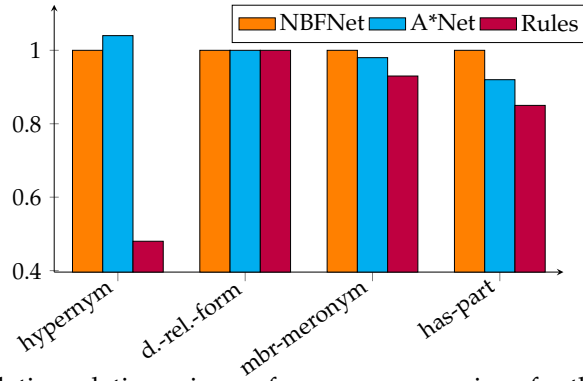


FIGURE 7.3: Relative relation-wise performance comparison for the head direction. NBFNet is one. The first four relations with the most facts on the test set are shown (largest first). *Hypernym* consists of 1251 facts while *has-part* only consists of 172 facts.

hypernym relation. For the relation with the second most facts (1074), the performance is the same and the third highest relation only has 253 facts in the test set.

Therefore, the differences in predictive performance are mostly based on the head direction of the *hypernym* relation. To further emphasize this, we calculate a hypothetical MRR of the rule-based approach where we keep all relation and direction results constant but plug in the MRR head results for the *hypernym* relation of NBFNet. This results in a joint MRR of 0.533. **Therefore, more than 60 percent of the improvement of the GNNs can be traced back to the improvement with respect to the *hypernym* relation in head direction.**

7.2.2 Only-One-Tail Rule

The *hypernym* relation describes a semantic supertype relationship between a specific and a more generic term, such as *hypernym(red, color)* and *hypernym(orange, fruit)*. A closer inspection of the WN18RR training set reveals that the relation is of type N-to-1, that is, each head entity can only be associated with exactly one tail entity. While the structural composition of such a relation type cannot be explicitly used by the model classes, it can be reformulated to a negative inference rule:

Definition 22 (Only-One-Tail rule). *Let x, y, z be variables and p be a relation. The Only-One-Tail (OOT) rule w.r.t. p is given by:*

$$\forall x, y : (\exists z p(x, z)) \rightarrow \neg p(x, y) \quad (7.1)$$

The OOT rule is satisfied in the Zoo KG with respect to the *follows* relationship (you cannot follow Anna when you follow someone else already). It is also satisfied for the WN18RR dataset with respect to the *hypernym* relation as it is a N-to-1 relation. The rule is relevant for the head direction, i.e., for the head rankings. It holds within the training graph and it also translates from the training to the test graph. For instance, an entity c cannot be the answer to the test query *hypernym(?, o)* if it has a *hypernym* already in the training graph.

Clearly, the GNNs cannot execute the negative rule in a crisp fashion as written in 7.1. However, we have already seen for the Zoo KG that they are able to exploit such a pattern. Therefore, we hypothesize that one reason for the performance improvements of the GNNs lies in the fact that they are able to penalize the scores of candidates that violate the OOT rule.

7.2.3 Perturbation Experiments

For the Zoo KG, one can easily draw conclusions about what the models learned, given the simplicity of the dataset. For a real benchmark like WN18RR, on the other hand, many different patterns and connection between entities exists and it is not straightforward to isolate model behaviour with respect to one particular inference rule.

Therefore, we use perturbation-based experiments that are inspired by the protocol that we used in Chapter 4. An already trained GNN performs message passing over the training graph for answering test queries at inference time. This allows to query the GNNs based on small perturbations of the training graph. The experiments are based on the idea that removing or adding a fact that activates or deactivates the OOT rule should be reflected by significant score changes if the rule is indeed exploited by the model. In fact, if the rule is activated, the score of a test fact should decrease while it should increase if the rule is deactivated.

Experimental Protocol

Generally, all the experiments are independently performed with respect to the 1251 test facts of the *hypernym* relation. First, we define as **test fact** one of these 1251 facts. We define as **base fact** the fact on which we perform a score analysis by first scoring it with the original training graph and subsequently scoring it after a perturbation of the training graph. The **base fact** will be the same as the **test fact** for the *Add* experiments and it will be derived from the **test fact** for the *Del* experiments; details are described below. We say that the **influential fact** is a fact that activates the OOT rule for a base fact. For instance, for the base fact *follows(lisa,anna)*, the influential fact *follows(lisa,bobby)* exists in the train graph of the zoo dataset while *follows(lisa,mike)* is an influential fact that does not exist. The *Add* experiments are based on adding an influential fact when it does not yet exist while the *Del* experiments are based on removing one that already exists. Further note that the GNNs calculate directed scores. The head score for a fact $p(s, o)$ is based on the head query $p(?, o)$. The tail score for the fact $p(s, o)$ is a separate calculation based on the tail query $p(s, ?)$. We will only focus on the head scores in the following given the structure of the OOT rule.

Add

For each test fact $hypernym(s, o)$, we set the base fact to be the same as the test fact. We calculate the original head score of this base fact and store it. Subsequently, we select a random entity o' and perturb the training graph by adding the influential fact $hypernym(s, o')$ and calculate the score again. The average difference in scores is termed $\Delta Attack$. We compare this to a random baseline where we sample o' as before but then we also sample a random relation $p' \neq hypernym$ and add the non-influential fact $p'(s, o')$ to the graph. The score difference is termed $\Delta Random$. All perturbations are based on the original training graph such that side effects can be ruled out.

	Model	Avg. Score	Δ Attack	Δ Random
Add	NBFNet	2.39 ± 1.93	-3.86 ± 1.35	-0.64 ± 0.84
	A*Net	3.53 ± 2.59	-2.89 ± 1.93	-0.68 ± 0.82
Del	NBFNet	-3.94 ± 0.77	2.70 ± 0.94	0.41 ± 0.54
	A*Net	-2.79 ± 1.55	1.54 ± 1.57	0.10 ± 0.55

TABLE 7.3: Average head score changes on WN18RR after perturbation experiments with respect to target queries $_hypernym(?, entity_i)$ based on the test set.

Del

It was explained above that the OOT rule is satisfied in the dataset for the *hypernym* relation. Therefore, for a test fact $hypernym(s, o)$, the head entity s does not have a *hypernym* in the training set. In other words, there does not exist a correct fact for which an influential fact regarding the OOT rule with the *hypernym* relation exists. Therefore, we first have to construct a base fact with an existing influential fact in the training graph. For each test fact $hypernym(s, o)$, we first sample some fact $hypernym(s', e) \in \mathcal{G}^{train}$ with some entities $e, s' \neq s$, which will serve as the influential fact for a constructed base fact. Subsequently, we construct the base fact $hypernym(s', o)$ based on the original object of the test fact. We calculate the score of $hypernym(s', o)$ under the original training graph. Subsequently, we delete $hypernym(s', e)$ from the training graph and calculate the score again. The difference is termed $\Delta Attack$. For the random comparison, we leave the influential fact in the training graph and delete a random triple that contains s' instead.

7.2.4 Perturbation-Based Results

Table 7.3 shows average score changes and standard deviations over all tested base facts. Adding the influential fact (activating the negative rule) leads to a score decrease of 3.86 for NBFNet and 2.89 for A*Net, respectively. Compared to the averaged initial scores the magnitude of the effects is substantial. For instance, for NBFNet the average score decreases from 2.39 to a negative value of -1.47. Deleting the influential fact (deactivating the negative rule), on the other hand, leads to a substantial score increase. The effects are significantly stronger compared to the random perturbation although the signs of the effects are the same. In general, it is likely possible that there exist other negative patterns in the dataset which will also affect the random baseline. We additionally investigate the effects on the predictive performance of the GNNs. For the *Add* setting, we calculate rankings based on the scores computed after the perturbation. Note that after processing and scoring of a test fact, the graph is reset to the original training graph to exclude side effects. Table 7.4 shows the results. The left-hand-side depicts results for the target relation in head direction and the right-hand side shows overall MRR effects. For all other relations except *hypernym*, the scoring and ranking is unchanged compared to the original model.

For NBFNet, the head MRR for *hypernym* decreases from 0.274 to 0.088 while it decreases from 0.281 to 0.121 for A*Net. The effects for the overall MRR and Hits values are expectedly smaller as the scoring is unchanged for all other relations. Nevertheless, the MRR decreases from 0.551 to 0.500 for A*Net and it decreases to 0.514 for NBFNet. The effects of the random perturbation are significantly smaller. For instance, the MRR for the *hypernym* head direction is decreased from 0.274 to

Setting		<i>hypernym</i> Head			Overall		
		MRR	H@1	H@10	MRR	H@1	H@10
NBFNet	Original	0.274	0.197	0.428	0.551	0.496	0.661
	Attack	0.088	0.066	0.131	0.500	0.458	0.586
	Random	0.227	0.159	0.374	0.526	0.474	0.633
A*Net	Original	0.281	0.207	0.420	0.551	0.498	0.655
	Attack	0.121	0.081	0.192	0.514	0.468	0.603
	Random	0.242	0.169	0.371	0.535	0.482	0.638

TABLE 7.4: Effects on the MRR for the head direction of the *hypernym* relation and overall when adding the influential fact or a random fact individually w.r.t each test fact.

0.227 for NBFNet under the random perturbation and it is decreased from 0.281 to 0.242 for A*Net.

We conclude that the GNNs comply with the induced behaviour of the OOT rule: The score significantly decreases if the rule is activated by a perturbation and it increases if the rule is deactivated for a base fact. Moreover, the observed score changes have significant implications for the predictive performance.

7.2.5 Augmenting the Rule-Based Approach

We have seen that the GNNs are sensitive to perturbations that activate or deactivate the OOT rule. The next step of our analysis concerns investigating whether the rule-based approach can be augmented such that it is able to express the negative pattern. This requires to penalize scores of candidates that are excluded by the rule because they already have assigned a *hypernym* in the training set.

A simple approach is given by augmenting the learned aggregation functions with additional feature types. In particular, we focus on the LR model without positive sign constraint that was already used for the synthetic dataset in Section 7.1.1. The model was introduced in Section 6.2 and defined in equation (6.5) where each learned rule constitutes one feature and is assigned with a weight. We discussed already that this approach cannot express the OOT rule for the *hypernym* relation as the rule cannot be expressed in the language bias. Therefore, we will augment the equation with existence features with respect to the current target fact.

Let f be the target fact with respect to the *hypernym* relation for which the LR model has to calculate a score. We define a binary feature $z(f) \in \{0, 1\}$ that evaluates to one, if the head entity already exists in the head position of the *hypernym* relation in some fact of the training KG. The resulting fact scoring function is given as

$$\phi_{LR}(f; \Omega) = \sigma \left(\sum_{i=1}^N \tilde{\omega}_i x_i + \omega_0 + \beta z \right). \quad (7.2)$$

The additional weight $\beta \in \mathbb{R}$ is learned in conjunction with the rule weights using the same training procedure as described in Section 6.5.

The new scoring function exclusively affects the scores with respect to head queries of the *hypernym* relation. All other scores are unchanged. However, we can additionally generalize the concept of existence features and define them for

different positions (head or tail) and for different relation pairs. Although the OOT rule is defined recursively, it is also possible that more general patterns exist that are exploited by the GNNs. For instance, a negative rule could state that an entity cannot exist in the head position of some relation if it already exists in the tail position of some other relation. We show all possible existence features that can be created in Table 7.5.

Target Fact	Type	Condition
$p(s, o)$	head, same position	$\exists e : q(s, e) \in \mathcal{G}$
$p(s, o)$	tail, same position	$\exists e : q(e, o) \in \mathcal{G}$
$p(s, o)$	head, inverse position	$\exists e : q(e, s) \in \mathcal{G}$
$p(s, o)$	tail, inverse position	$\exists e : q(o, e) \in \mathcal{G}$

TABLE 7.5: Possible existence feature types that can be created for one target fact where p and q are relations. The overall number of features is $4 |\mathcal{P}|^2$.

In the table, \mathcal{G} refers to the training KG on which the rules are applied. In the second column, the type of the feature is displayed and the third column shows the condition when the feature evaluates to one. When the condition is not met, the feature evaluates to zero. For instance, the type *head, same position* describes a feature that is one if the target fact to be scored contains relation p and the head entity of the target fact already appears in the head position in a fact with relation q . For $p = q = \textit{hypernym}$, the single feature that was described in equation (7.2) is obtained.

We will perform two experiment specifications. First, we only include the feature regarding the head direction of the *hypernym* relation as described above. With this experiment, we investigate if the model is able to exploit the OOT rule by learning a high negative value for the weight of the feature from the data. Second, we augment the model with all possible feature types for all relation pairs as described in Table 7.5. This experiment provides insights into the question if there exist additional patterns like the OOT rule that provide benefits for the predictive performance. The results will be presented in the next section.

Results

Before we turn to the results based on WN18RR, we will complete the discussion regarding the Zoo KG from Section 7.1.1. The last row of Table 7.1 shows an experiment in which we added an existence feature that evaluates to one, if an entity already follows another entity in the data. The feature evaluates to one for all students except of Bobby. The learned weight of the feature value is high negative and it helps to answer the test query correctly, i.e., the induced behaviour now aligns with the GNNs.

Table 7.6 shows the result for WN18RR after training the models with the augmented scoring formulations. The second last row shows the specification in which we only added one additional feature for the *hypernym* head direction. This improves the overall MRR already from 0.500 to 0.525. The effect is even more clear when only considering the test facts that are composed of the *hypernym* relation as only these facts are affected. Here, the result almost doubles from 0.129 to 0.238. Inspecting the learned weight reveals that it is large negative as expected. The last row of the table shows the specification in which we use all possible existence features. We observe an additional improvement with an MRR of 0.534. However, the largest improvement

is reached by the single feature for the *hypernym* head direction. Nevertheless, the results suggest that there exist more negative patterns that can be exploited.

Approach	Overall			<i>hypernym</i>
	MRR (T)	MRR (H)	MRR	MRR (H)
Rules (LR)	0.530	0.471	0.500	0.129
NBFNet	0.561	0.541	0.551	0.274
A*Net	0.559	0.543	0.551	0.280
LR + \exists -hyper.	0.530	0.520	0.525	0.238
LR + \exists -all	0.544	0.524	0.534	0.245

TABLE 7.6: MRR results for WN18RR. MRR (T) and MRR (H) are tail and head MRRs over all relations. The third column is the joint MRR.

The experiment results suggest that the initial rule-based approach does not exploit the OOT rule. Adding the existence features, on the other hand, allows it to exploit the negative pattern for the *hypernym* relation to a large extent.

7.3 The Synthetic Uni Dataset

We will now introduce a synthetic dataset that is relevant for the discussion of the predictive performance differences between the rule-based approach and the GNNs on the Fb15k-237 benchmark. It will help us to discover structural model differences with respect to particular training settings of the GNNs.

Figure 7.4 shows the Uni KG, where students ask a question to a professor and the professor answers to some of the students. All students (including Anna and Bernd) and the professor are *member* of the university which is expressed with dashed arrows where some of them are excluded in the figure to prevent occlusion. We duplicate the pattern shown in the figure three times, such that there are three groups of students and a professor (including three versions of Anna and Bernd) but all of them are member of the global *uni* entity.

The fact *answered(professor, anna)* is removed from the graph and used as an evaluation fact. For simplicity, we will focus on the tail direction in the following. After the training stage, from the evaluation fact, the models have to answer the query *answered(professor, ?)*. The only two relevant candidates are Bernd and the correct answer Anna. The remaining *student_i* entities are already known to have received answers from the professor in the training data. Therefore, they are filtered out by the evaluation protocol (compare Section 2.2.1).

When looking at the data, it becomes apparent that Anna is a more reasonable choice than Bernd: There exists a clear pattern which says that an entity that asked a question also receives an answer. The pattern can be described by a **B**-rule with length one, *answered(X, Y) ← asked(X, Y)*, and it is supported by all the student entities. While Anna asked a question, nothing is known about Bernd. Therefore, the data provides clear evidence that Anna should be ranked above Bernd. Conversely, there is no reason why Bernd should be assigned a higher score than Anna for the query *answered(professor, ?)*.

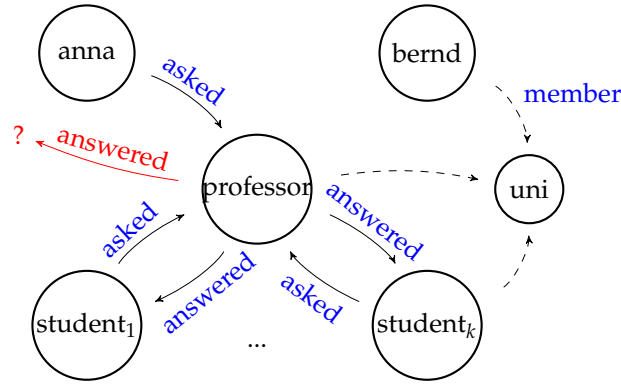


FIGURE 7.4: Graphical representation of the uni dataset. Each node is connected via the *member* relation (dashed arrows) to the *uni* node. The task is to move the red arrow pointing either to *bernd* or *anna*. The dataset regularities clearly suggest *anna*; there is no reason why the arrow should be pointing to *bernd*.

7.3.1 Experiments on the Uni KG

Table 7.7 shows the results for the evaluation fact. The selected models and experimental details are the same as described in Section 7.1.1. We will focus on the tail direction in the following. The discussion is the same for the head direction. In the first four rows, all specifications achieve a tail MRR of 1.0 after every training run. That is, each of the different models confidently assigns a higher score to Anna than to Bernd, given the query $answered(professor, ?)$.

	Approach	MRR Tail	MRR Head	MRR
ROH off	Rules (LR)	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	ComplEx	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	NBFNet	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	A*Net	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
ROH on	NBFNet	0.22 ± 0.04	0.33 ± 0.00	0.28 ± 0.02
	A*Net	0.18 ± 0.30	0.22 ± 0.13	0.20 ± 0.19

TABLE 7.7: MRR results for the uni dataset. We report averages (\pm std. dev.) over 10 runs.

The last two rows of Table 7.7 show the results for the GNNs when the ROH parameter is turned on. It is a hyperparameter that can be turned on or off in the configuration files of the implementation provided by Zhu et al. (2021). It only affects the training stage of the GNNs and it has no influence at test time. In particular, during training, for a given training fact $p(s, o)$, all direct connections between entities s and o are removed, i.e., each fact $p'(s, o)$ with $p' \neq p$ is removed from the training graph for the current batch updates.

Recall from Section 7.1 that for the Zoo KG, the parameter does not have any effect on the experiments. However, for the Uni KG, it drastically alters the evaluation results. We pick again the tail direction as an example, but the effects hold for both directions. For instance, the average MRR over 10 runs decreased to 0.18 for A*Net. In fact, in 9 out of 10 runs, Anna is assigned a lower score than Bernd (and some other student nodes that do not have a direct connection to the professor). Furthermore,

for NBFNet, in each of the 10 training runs, Bernd is ranked higher than Anna.

7.3.2 Discussion

It is not straightforward to exactly describe the induced behaviour when the GNNs are trained under the ROH hyperparameter. However, we can say that the model is never shown a true fact $p(s, o)$ where s and o are directly connected via some other relation.

Due to the simplicity of the uni dataset, there is only one edge that distinguishes Bernd from Anna and it is the edge that directly connects Anna with the professor with respect to the *asked* relation. Therefore, from the perspective of the query $answered(professor, ?)$, the direct connection between Anna and the *source* node *professor* must be the reason for the lower score despite the fact that it actually describes an important positive data regularity. A possibility of replicating the behaviour of the GNNs is given by scoring Anna and Bernd randomly from $[0, \epsilon]$ and then subtracting $\epsilon + 1$ from the score whenever a direct connection between the source node (the professor) and the candidate is given.

7.4 Negative Pattern on Fb15k-237

The rule-based approach achieves an MRR that is more than 5 percentage points lower compared to the GNNs on the Fb15k-237 test set (compare Figure 7.1). While we have performed an initial fine-grained comparison of the predictive performances of the models, we could not find any structural differences. The differences are rather evenly distributed over different relations and query directions.

We observed in Section 7.3 for the Uni KG that a direct connection between a source entity (*professor*) and a candidate entity (Anna) leads to a score decrease for a respective fact when the GNNs are trained under the ROH hyperparameter. In contrast, such a behaviour is not shown by the rule-based approach and the KGE model. In fact, positive rules with only one body atom are important for the overall predictive performance in general. We will now introduce the Only-One-Link rule:

Definition 23 (Only-One-Link rule). *Let x, y be variables and let \mathcal{P} be a set of relations of a KG. Let $\mathcal{P}_c \subseteq \mathcal{P}$ denote a set of (constraint) relations and $p^* \in \mathcal{P}$ be a target relation. The Only-One-Link (OOL) rule w.r.t. target p^* and constraints \mathcal{P}_c is given by:*

$$\forall x, y : \left(\bigvee_{p \in \mathcal{P}_c \setminus \{p^*\}} p(x, y) \right) \rightarrow \neg p^*(x, y) \quad (7.3)$$

The rule simply says that two entities cannot be connected via a relation p^* if they are already connected by a relation from \mathcal{P}_c . Unfortunately, the rule does not have relevance for the training set of the Fb15k-237 KG alone. Here, it is possible that two entities are directly connected by different relations. However, due to the specific construction of the entire data set, the rule applies from the training set to the test set for all relations.

In particular, two entities e_1, e_2 that appear in some fact with any relation, e.g., $p(e_1, e_2) \in \mathcal{G}^{train}$, will never appear together in a fact in the test set by construction (Toutanova and Chen, 2015). We have already mentioned this briefly in Section 2.2.3 when describing the construction of Fb15k-237. While the authors mention that such multiple direct links are very likely in real world scenarios, they constructed

the dataset in that way to decrease to possibility of inducing trivial patterns. However, a model that is able to penalize a candidate score when source and target entity are already connected in the training graph, will have a significant advantage.

7.4.1 Perturbation-Based Experiments

Not surprisingly, in the original configuration files of the GNNs, the ROH parameter is turned off for WN18RR and it is activated for Fb15k-237. We employ perturbation results similar to Section 7.2.3 to investigate if the model specifications for Fb15k-237 exploit the OOL rule. As before, we re-train the models with the code provided by the original publication (Zhu et al., 2024) and we use the evaluation code from the PyClause library.

We also train and evaluate the models with ROH turned off (Table 7.9). The perturbation experiments are based on the facts of the test set. The definitions from Section 7.2.3 apply, e.g., we term a fact that activates the negative rule an **influential fact** and the score analysis is based on a **base fact**.

Add

Every fact from the test set is used as a base fact. For each test fact $p(s, o)$, we calculate the original head and the tail scores. Subsequently, we sample a relation p' and add the influential fact $p'(s, o)$ to the training graph. Then we recalculate the scores for $p(s, o)$. We report the difference of scores before and after the perturbation (Δ Attack). For the random specification, we sample p'_1, p'_2, e_1 and e_2 and add two triples to the graph instead. In particular, we add $p'_1(s, e_1)$ and $p'_2(e_2, o)$ and then recalculate the head and tail score for $p(s, o)$. We do this in favour of the random baseline as the GNNs calculate directed triple scores.

Del

For every test fact $p(s, o)$, we first have to find a base fact for which an influential fact exists. By construction of the KG, if $p(s, o)$ is in the test set, then s, o will not appear in another fact in the training set. Therefore, for the tail direction, we sample an existing fact $p'(e, o) \in \mathcal{G}^{train}$ that contains the tail entity of the test fact; it serves as the influential fact. We then create an artificial base fact $p(e, o)$ based on the target relation and the two entities e and o . The initial tail score for $p(e, o)$ is calculated where only facts with a score higher than one are considered to have somewhat realistic artificial base facts. Subsequently, we delete the influential fact $p'(e, o)$ from the training set and recalculate the score for $p(e, o)$ leading to the difference in scores. For the random specification, we delete some other triple from the training set instead which contains o but not e . Note that the score difference is still based on the artificial base fact $p(e, o)$ before and after the perturbation. For the head direction, we use the same approach but we start with the head entity of the test fact when sampling the influential fact.

7.4.2 Perturbation-Based Results

Table 7.8 shows average score changes for the tail direction after the perturbation as described above. The results for the head direction are qualitatively identical. We

	Model	Avg. Score	Δ Attack	Δ Random
Add	NBFNet	2.91 ± 2.88	-1.78 ± 2.20	-0.08 ± 0.33
	A*Net	2.89 ± 3.14	-2.44 ± 2.32	-0.05 ± 0.68
Del	NBFNet	0.97 ± 0.92	1.49 ± 1.49	-0.01 ± 0.11
	A*Net	1.07 ± 1.01	1.73 ± 1.59	-0.01 ± 0.06

TABLE 7.8: Fb15k-237 (Tail direction) perturbation experiments for the default models trained with ROH=on.

observe that activating the OOL rule by adding a direct connection of source and candidate entity leads to a significant decrease in the score. Deactivating the rule by removing an influential fact, on the other hand, leads to an increase. Additionally, the effect of the random specification are marginal although in the *Add* random setting even two facts are added. The standard deviations that we observe are relatively high. For instance, not in every test case of the *Add* setting a score decrease is observed. One reason can be that adding the influential fact might induce overlapping effects. Although the OOL is activated it also may activate some other positive pattern that was learned on the training set. However, the experiments clearly suggest that the GNNs have a tendency to penalize a direct connection between the source and the candidate entity.

In Table 7.9, we show the effects on the joint MRR after the *Add* perturbations. Additionally, we re-train the models with ROH turned off and perform the *Add* perturbation for these specifications. Note that the *Add* attack setting essentially leaks the correct answer by the fact that is added to the graph. For instance, for a test fact $p(s, o)$ leading to the tail query $p(s, ?)$ with correct answer o , we add $p'(s, o)$ with $p' \neq p$ to the graph. We can view this as informing the model about the correct answer via adding a random connection. Nevertheless, adding the influential fact leads to a significant score decrease as shown above. **Likewise, we observe that the MRR of NBFNet decreases from 0.416 to 0.287 despite leaking in the correct answer.**

Finally, in the right-hand side of Table 7.9, we report the MRR of the models when trained without the ROH parameter. The predictive performance decreases and it is only slightly better than the rule-based approach or a good KGE model. Noteworthy, when we repeat the *Add* experiments for these specifications, the effect turns around. For instance, for NBFNet the MRR increases from 0.371 to 0.454. Intuitively, this behaviour is more natural as informing the model with the correct answer by a random connection should not decrease the ability of the model to find it.

7.4.3 Post-Hoc Experiments

We discussed previously that the OOL rule is not reflected within the training set of Fb15k-237. Therefore, we are not able to learn weights for existence features on the dataset. While it is possible to add artificial negative facts to the training procedure, we choose the most pragmatic approach that does not involve manipulating the training data. If external knowledge is given about the structure of the OOL rule, we can simply remove all candidate proposals at inference time that already have a connection to the current source entity. By this procedure, we strictly enforce the pattern. For the rule-based approach, we use the re-implementation of the LR model as in Section 7.2.5.

Model	Setting	ROH=on (default)			ROH=off		
		MRR	H@1	H@10	MRR	H@1	H@10
NBFNet	Original	0.416	0.324	0.596	0.371	0.271	0.573
	Add	0.287	0.184	0.497	0.454	0.339	0.674
	Random	0.410	0.316	0.594	0.368	0.267	0.570
A*Net	Original	0.412	0.323	0.589	0.370	0.276	0.562
	Add	0.256	0.178	0.410	0.406	0.291	0.635
	Random	0.410	0.321	0.586	0.368	0.273	0.558

TABLE 7.9: MRR and Hits effects on Fb15k-237 when adding the influential fact or adding a random fact. Left: The original model specification. Right: The original specification while ROH is turned off. The add scenario essentially leaks the correct answer by a random connection. For the ROH specification this leads to a decrease in the performance (left) whereas it leads to an improvement when the hyperparameter is turned off (right).

For the post-hoc filter, we choose the following approach exemplified with the tail direction: For a query $p(s, ?)$ with correct answer o formed from the test fact $p(s, o)$, we delete every answer candidate e from the ranked list of candidates if it already has a connection to s via some fact in the training set. The approach is the same for the head direction starting with a head query instead.

If a model does not exploit the OOL rule, the filtering approach should lead to a significant boost in the predictive performance. If a model uses the rule already, on the other hand, the filtering should not affect the evaluation results.

Setting	Approach	Original			Post-hoc Filter		
		MRR	H@1	H@10	MRR	H@1	H@10
	LR	0.361	0.275	0.534	0.385	0.304	0.548
ROH on	NBFNet	0.416	0.324	0.596	0.417	0.325	0.597
	A*Net	0.412	0.324	0.589	0.415	0.325	0.592
ROH off	NBFnet	0.371	0.271	0.573	0.415	0.322	0.596
	A*Net	0.370	0.276	0.562	0.411	0.324	0.583

TABLE 7.10: MRR results for Fb15k-237 and post-hoc filter experiments for all model classes. The post-hoc filter does almost have no have effect for the original GNN specifications as the OOL rule is already exploited. If ROH is turned off, enforcing the OOL pattern by the filter leads to an increase in performance.

Table 7.10 contains the results. The first row contains the results of the LR model of the rule-based approach. The MRR is increased by 2.3 percentage points to 0.385 when applying the post-hoc filter which is a substantial improvement. We conclude that half of the improvement over a rule-based approach can be explained by exploiting the OOL rule.

The two middle rows of the table show the GNNs trained under their original hyperparameters, i.e., with the ROH parameter activated. The post-hoc filter has almost no effect on the predictive performance results. This provides further evidence

to the hypothesis that the GNNs already exploit the OOL pattern. Additionally, when considering the last two rows of the table in which the ROH parameter is not activated, the results change significantly. In this case, applying the post-hoc filter leads to an increase of the MRR of more than four percentage points for both models. This, again, aligns with the hypothesis that the activation of the ROH parameter leads to the exploitation of the OOL rule whereas the pattern is not exploited when the parameter is turned off during training.

7.5 Discussion and Conclusion

In this chapter, we have taken a closer look at the WN18RR and FB15k-237 benchmarks and investigated the differences in predictive performances between the GNN models A*Net (Zhu et al., 2024) and NBFNet (Zhu et al., 2021) and the rule-based approach. We provided empirical evidence that the GNNs are able to exploit certain negative patterns that cannot be expressed by the rule types considered within this thesis. However, we also showed that the performance improvements are based on trivial regularities to some extent. Additionally, our synthetic experiments suggest that for these cases, the behaviour of a KGE model like ComplEx (Trouillon et al., 2016) is more similar to the rule-based approach; this is in line with our empirical findings discussed in Chapters 3 and 4.

7.5.1 Differences in Predictive Performance

For WN18RR, our results clearly suggest higher expressivity of the GNNs compared to the rule-based approach. The GNNs are able to penalize incorrect candidates based on the OOT rule. With the current language bias of the rule-based approach, this behaviour cannot be replicated without a conceptual augmentation.

The conclusion for the Fb15k-237 benchmark is more ambiguous. On the one hand, the shown behaviour under the ROH setting is remarkable and helps to achieve a higher predictive performance. However, the underlying OOL rule is a structural bias introduced during the benchmark construction without a resemblance to a real-world regularity. Moreover, it cannot be learned from the training set directly since it only holds from the training to the test set. On a new dataset, it is unclear for a potential user of the system, if the models should be trained with the parameter activated or deactivated. Future work is required to better understand the resulting model behaviour.

Our experimental results suggest that more than 50 percent of the performance differences can be explained by the unique negative patterns on each dataset. While on WN18RR the patterns are based on only one relation and query direction, on Fb15k-237, they span over all relations.

Finally, it is likely that the patterns that are learned by the GNNs (positive or negative) might overlap. A fact could, for instance, simultaneously activate some positive pattern while also activating some other negative pattern. Our experimental results should therefore only be seen to identify tendencies of model behaviour. Moreover, the GNNs might have learned different negative rules and also more general formulations consisting of distinct relations or even multi-hop patterns.

7.5.2 Improving the Rule-Based Approach

After obtaining the insights based on our empirical analyses, we proposed methods to imitate the GNN behaviour with the rule-based approach. Figure 7.5 shows the

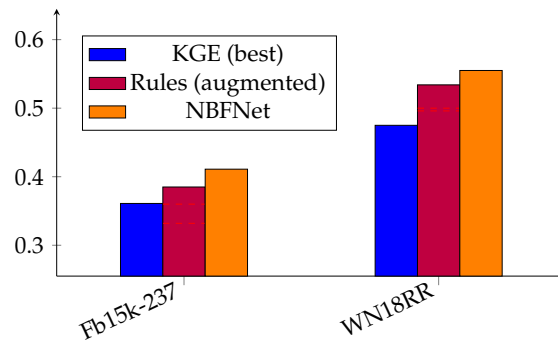


FIGURE 7.5: MRR results for WN18RR and Fb15k-237. The lower dashed lines depict the Max+ result. The upper dashed lines depict the LR results.

summarized MRR results where the lower dashed lines depicts the Max+ results and the upper dashed line the LR model from the previous chapter. In particular, for WN18RR, our analysis inspired the simple augmentation of the trained aggregation function with existence features. The resulting model achieves a substantial improvement. These results are especially interesting as learning the aggregation functions in Chapter 6 provided no significant benefits on this benchmark. While there is still an overall gap in the predictive performance, learning the LR models is simpler and remains interpretable even when existence features are included.

For Fb15k-237, we have shown how to easily augment the rule-based approach by a post-hoc filter such that the pattern is satisfied at inference time. In cases in which similar structural constraints exist in the data and are known beforehand, a post-hoc filter is a suitable method of enforcing the models to comply with these constraints. Although the gap to the GNNs was not closed fully, our results can be seen as pointers to possible future work for augmenting the rule based approach. This will also be discussed in the next chapter.

Chapter 8

Conclusion and Outlook

Rule-based approaches learn symbolic representations from KGs in the form of human-readable logic rules. In this thesis, we have explored how to employ them effectively for predicting missing facts given an incomplete KG.

We contrasted symbolic KGC with another popular method given by KGE models. While based on fundamentally different paradigms, our experimental results suggest that the rule-based approach remains relatively within the language bias of KGE models. Rule-based predictions are fully transparent, as they are accompanied by an interpretable rule and the facts that lead to a prediction. We demonstrated that the explanatory power can also be used to explain predictions made by KGE models. When predicting new facts under uncertainty with rules, an approach has to be chosen for how to combine them effectively. Therefore, we defined and analysed the confidence aggregation problem from an empirical and a theoretical perspective. Our theoretical results allow us to embed confidence aggregation into the correct context of logical reasoning under uncertainty. We showed that aggregating based on the rule with the highest confidence can likewise be expressed within the formal framework instead of being a computational heuristic. Additionally, we discussed the role of multi-step reasoning and its difficulties when KGs are large.

We proposed various methods for improving the confidence aggregation and empirically demonstrated the superiority of learning the confidences directly from the KG. Our results show that on the evaluated benchmarks, the rule-based approach is slightly superior to the basic KGE models. Finally, we discussed the differences between rules and a particular GNN architecture. Although the analysis revealed that part of the improvement of the GNNs is based on an artificial bias in the dataset and a particular hyperparameter setting, we found that the GNNs could exploit patterns that are hidden from the rule-based approach. Moreover, the analysis inspired simple augmentation strategies of the rule-based approach, leading to further improvements in the predictive performance.

8.1 Discussion and Limitations

While there is research that focuses on improving the quality of individual rules (Azevedo and Jorge, 2007; Ho et al., 2018), this thesis takes a task-oriented approach. We argue that a suitable strategy to evaluate the quality of learned rules is given by putting them into use for a given task and relying on task-specific evaluation metrics such as the predictive performance. Moreover, their characteristics and performance should be compared to alternative methods that are already proven to work well for the same task. Putting the rules to the test by these comparisons makes it possible to uncover specific strengths and potential shortcomings. For instance, the direct comparison to KGE models motivated us to further explore the confidence

aggregation problem and allowed us to highlight the benefits of rules in terms of interpretability. Moreover, the comparison with GNNs not only provided transparency to their respective training details but also allowed to discover limitations of the language bias of the rule-based approach.

However, the discussions around this thesis were mostly based on the rule types of the rule learner AnyBURL that is especially tailored towards the task of query answering. Moreover, when comparing rules with latent-based approaches, their usability is less flexible. Embeddings can be used off-the-shelf after training for downstream applications such as entity classification (Portisch, Heist, and Paulheim, 2022). While there is research on learning entity representations also with rules (Ismaeil et al., 2023), it is less straightforward. Therefore, important future work is required to increase the usability of rules for different tasks and settings which will be also discussed below.

In general, we believe that symbolic representations should play an important role in the context of knowledge acquisition due to their transparency; a characteristic that might be of higher importance for potential system users than plain predictive performance. After all, there is no need to trust a rule-based system. Predictions are fully transparent and can be modified easily by correcting the rules or the background knowledge.

8.2 Future Work

Finally, we highlight some future research directions that either have not been explored or can be build upon the results of this thesis.

8.2.1 Number of Learned Rules

Within the experimental sections of this thesis, we have seen that a large number of rules is required to be competitive regarding the predictive performance for KGC. While the rule-based approach provides local interpretability in terms of explaining individual predictions, global interpretability would rely on a more compact set of learned rules. While measures such as rule support may help with finding the most important patterns, rulesets with millions of rules can be cumbersome to handle. Although there is a trade-off regarding the predictive performance, an interesting direction could be to search for smaller subsets of rules based on a desired threshold for the target evaluation metric. For example, searching for the smallest possible subset of rules that still achieves an MRR of x on some dataset. An alternative direction could be to incorporate KGC evaluation into the rule learning process and assign costs to each rule that is added in order to balance the trade-off with the predictive performance.

8.2.2 Additional Applications for Rules

The main focus of this thesis is the task of KGC where new facts have to be predicted given an incomplete KG. In regard to the employed evaluation protocols, no external data sources can be employed and all entities of the test queries are known beforehand. This setting is also commonly termed the transductive setting. In the inductive setting, on the other hand, new entities are discovered at test time (Kochsiek and Gemulla, 2023; Liang et al., 2024). An interesting future direction is to evaluate how to apply the proposed methods for confidence aggregation on this task.

Moreover, augmenting KGs with multi-modal data sources (Zhang et al., 2024) and incorporating symbolic approaches could enrich the usability scope of rules. Along these lines, rules could be employed for other relational tasks such as visual relationship detection (Baier, Ma, and Tresp, 2017) when combined with suitable neural models such as object detectors (Ren et al., 2016).

Finally, the combination of information retrieval with language models received large attention in the last years (Lewis et al., 2020). While these approaches are not interpretable, there exist branches that incorporate KGs for having structured and correct background knowledge and combine the performed inference with logical reasoning (Walker, Ultes, and Lison, 2023; Edge et al., 2024). Rule-based approaches employing one-step entailment could provide an important interface to explainable inference, especially in large data settings.

8.2.3 Augmenting the Language Bias

Within this thesis, we have focused on the particular rule types supported by the rule learner AnyBURL. Naturally, the question arises if additional types can provide further benefits. Regarding positive rules, a suitable confidence aggregation potentially allows also to express patterns that cannot be expressed by the raw types. Although we did not encounter empirical evidences for such cases in real-world KGs, a starting point for evaluating this quantitatively could be given by the synthetic KGs provided in the work of Haiquan Qiu and Yao (2024) which contain patterns that are compositions of the raw rule types used within this thesis.

When also considering negative rules or patterns such as discussed in Chapter 7, the task gets more complicated. Although research exists of how to learn negative rules from KGs, it is not discussed how to incorporate them into the confidence aggregation problem for increasing the predictive performance of the set of rules (Ortona, Meduri, and Papotti, 2018). An entry point could be given by the theoretical framework of Chapter 5. The semantics could be augmented such that a fact is false if it is predicted by at least one negative rule. For Noisy-or aggregation, this would result in multiplying the Noisy-or product with the "confidence" of the negative rule, decreasing the final probability. It is an open question at this point how the Max assumption would be affected and, additionally, how explicit negative rules should be incorporated into the supervised confidence aggregation methods.

Bibliography

- Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org.
- Agarap, Abien Fred (2018). “Deep learning using rectified linear units (relu)”. In: *arXiv preprint arXiv:1803.08375*.
- Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami (1993). “Mining association rules between sets of items in large databases”. In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pp. 207–216.
- Alberto Garcia-Duran, Mathias Niepert (2018). “KBLrn: End-to-End learning of knowledge base representations with latent, relational, and numerical features”. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 372–382.
- Ali, Mehdi et al. (2021). “Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.12, pp. 8825–8845.
- Auer, Sören et al. (2007). “Dbpedia: A nucleus for a web of open data”. In: *The semantic web*. Springer, pp. 722–735.
- Azevedo, Paulo J. and Alípio Mário Jorge (2007). “Comparing rule measures for predictive association rules”. In: *ECML*. Vol. 4701, pp. 510–517.
- Baader, Franz, Ian Horrocks, and Ulrike Sattler (2008). “Description logics”. In: *Foundations of Artificial Intelligence* 3, pp. 135–179.
- Bacciu, Davide et al. (2020). “A gentle introduction to deep learning for graphs”. In: *Neural Networks* 129, pp. 203–221.
- Baier, Stephan, Yunpu Ma, and Volker Tresp (2017). “Improving visual relationship detection using semantic modeling of scene descriptions”. In: *Proceedings of the 16th International Semantic Web Conference*. Springer, pp. 53–68.
- Balazevic, Ivana, Carl Allen, and Timothy Hospedales (2019). “Tucker: Tensor Factorization for Knowledge Graph Completion”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5185–5194.
- Barceló, Pablo et al. (2020). “The logical expressiveness of graph neural networks”. In: *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.
- Batagelj, Vladimir and Matjaž Zaveršnik (2011). “Fast algorithms for determining (generalized) core groups in social networks”. In: *Advances in Data Analysis and Classification* 5.2, pp. 129–145.
- Bechler-Speicher, Maya, Amir Globerson, and Ran Gilad-Bachrach (2024). “The intelligible and effective graph neural additive network”. In: *Advances in Neural Information Processing Systems* 37, pp. 90552–90578.
- Bellodi, Elena and Fabrizio Riguzzi (2015). “Structure learning of probabilistic logic programs by searching the clause space”. In: *Theory and Practice of Logic Programming* 15.2, pp. 169–212.
- Bellodi, Elena et al. (2014). “Lifted variable elimination for probabilistic logic programming”. In: *Theory and Practice of Logic Programming* 14.4-5, pp. 681–695.

- Betz, Patrick, Christian Meilicke, and Heiner Stuckenschmidt (2022a). "Adversarial explanations for knowledge graph embedding models". In: *Proceedings of the 31th International Joint Conference on Artificial Intelligence*. Ijcai.org, pp. 2820–2826.
- (2022b). "Supervised knowledge aggregation for knowledge graph completion". In: *Extended Semantic Web Conference*. Springer, pp. 74–92.
- Betz, Patrick et al. (2024a). *A*Net and NBFNet learn negative patterns on knowledge graphs*. arXiv: 2412.05114.
- Betz, Patrick et al. (2024b). "PyClause: Simple and efficient rule handling for knowledge graphs". In: *Proceedings of the 33th International Joint Conference on Artificial Intelligence, demo track*. Ijcai.org.
- Betz, Patrick et al. (2024c). "Rule Confidence Aggregation for Knowledge Graph Completion". In: *International Joint Conference on Rules and Reasoning*. Springer.
- Bhardwaj, Peru et al. (2021). "Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 8225–8239.
- Blockeel, Hendrik et al. (1999). "Scaling up inductive logic programming by learning from interpretations". In: *Data Mining and Knowledge Discovery 3*, pp. 59–93.
- Bollacker, Kurt et al. (2008). "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250.
- Bordes, Antoine et al. (2013a). "A semantic matching energy function for learning with multi-relational data". In: *Machine Learning 94.2*, pp. 233–259.
- Bordes, Antoine et al. (2013b). "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems*, pp. 2787–2795.
- Bratko, Ivan (2001). *Prolog programming for artificial intelligence*. Pearson education.
- Broscheit, Samuel et al. (2020). "LibKGE-A knowledge graph embedding library for reproducible research". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 165–174.
- Burkart, Nadia and Marco F Huber (2021). "A survey on the explainability of supervised machine learning". In: *Journal of Artificial Intelligence Research 70*, pp. 245–317.
- Bylander, Tom et al. (1991). "The computational complexity of abduction". In: *Artificial Intelligence 49.1*, pp. 25–60.
- Chakraborty, Anirban et al. (2021). "A survey on adversarial attacks and defences". In: *CAAI Trans. Intell. Technol.* 6.1, pp. 25–45.
- Charpiat, Guillaume et al. (2019). "Input similarity from the neural network perspective". In: *NeurIPS 2019-33th Annual Conference on Neural Information Processing Systems*.
- Chen, Sanxing et al. (2021). "Hitter: Hierarchical transformers for knowledge graph embeddings". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10395–10407.
- Corso, Gabriele et al. (2020). "Principal neighbourhood aggregation for graph nets". In: *Advances in neural information processing systems 33*, pp. 13260–13271.
- De Raedt, Luc (2008). *Logical and relational learning*. Springer Science & Business Media.
- De Raedt, Luc, Angelika Kimmig, and Hannu Toivonen (2007). "ProbLog: A Probabilistic Prolog and Its Application in Link Discovery." In: *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2462–2467.

- Dettmers, Tim et al. (2018). "Convolutional 2d knowledge graph embeddings". In: *Proceedings of the thirty-second AAAI Conference on Artificial Intelligence*, pp. 1811–1818.
- Devlin, Jacob et al. (2019). "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186.
- Ding, Li et al. (2007). "Using ontologies in the semantic web: A survey". In: *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, pp. 79–113.
- Dong, Xin et al. (2014). "Knowledge vault: A web-scale approach to probabilistic knowledge fusion". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 601–610.
- Edge, Darren et al. (2024). "From local to global: A graph RAG approach to query-focused summarization". In: *arXiv e-prints*, arXiv–2404.
- Errica, Federico et al. (2020). "A fair comparison of graph neural networks for graph classification". In: *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*.
- Fan, Wenfei et al. (2022). "Discovering association rules from big graphs". In: *Proceedings of the VLDB Endowment* 15.7, pp. 1479–1492.
- Feng, Jun et al. (2016). "Knowledge Graph Embedding by Flexible Translation." In: *KR*, pp. 557–560.
- Ferré, Sébastien (2020). "Application of concepts of neighbours to knowledge graph completion". In: *Data Science*, pp. 1–28.
- Galárraga, Luis et al. (2015). "Fast rule mining in ontological knowledge bases with AMIE+". In: *The VLDB Journal* 24.6, pp. 707–730.
- Galárraga, Luis Antonio et al. (2013). "AMIE: association rule mining under incomplete evidence in ontological knowledge bases". In: *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 413–422.
- Gu, Yulong, Yu Guan, and Paolo Missier (2020). *Towards learning instantiated logical rules from knowledge graphs*. arXiv: 2003.06071.
- Haiquan Qiu Yongqi Zhang, Yong Li and Quanming Yao (2024). "Understanding expressivity of GNN in rule learning". In: *The Twelfth International Conference on Learning Representations*.
- Hanawa, Kazuaki et al. (2021). "Evaluation of similarity-based explanations". In: *The ninth International Conference on Learning Representations*.
- Himmelstein, Daniel Scott et al. (2017). "Systematic integration of biomedical knowledge prioritizes drugs for repurposing". In: *Elife* 6, p. 26726.
- Ho, Vinh Thinh et al. (2018). "Rule learning from knowledge graphs guided by embedding models". In: *International Semantic Web Conference*. Springer, pp. 72–90.
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation*. Vol. 9. 8. MIT Press, pp. 1735–1780.
- Hoffart, Johannes et al. (2013). "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia". In: *Artificial intelligence* 194, pp. 28–61.
- Huber, Mark and Nevena Marić (2019). "Admissible Bernoulli correlations". In: *Journal of Statistical Distributions and Applications* 6.1, pp. 1–8.
- Ismaeil, Youmna et al. (2023). "FeaBI: a feature selection-based framework for interpreting KG embeddings". In: *International Semantic Web Conference*. Springer, pp. 599–617.
- Joe, Harry (1997). *Multivariate models and multivariate dependence concepts*. CRC press.

- Kadlec, Rudolf, Ondrej Bajgar, and Jan Kleindienst (2017). "Knowledge base completion: Baselines strike back". In: *2nd Workshop on Representation Learning for NLP*.
- Kersting, Kristian and Luc De Raedt (2001). "Towards combining inductive logic programming with Bayesian networks". In: *Inductive Logic Programming: Proceedings of the 11th International Conference*. Springer, pp. 118–131.
- Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.
- Kochsiek, Adrian and Rainer Gemulla (2023). "A benchmark for semi-inductive link prediction in knowledge graphs". In: *Findings of the Association for Computational Linguistics: EMNLP*. Association for Computational Linguistics, pp. 10634–10643.
- Kochsiek, Adrian et al. (2023). "Friendly neighbors: Contextualized sequence-to-sequence link prediction". In: *The 61st Annual Meeting Of The Association For Computational Linguistics*.
- Koh, Pang Wei and Percy Liang (2017). "Understanding black-box predictions via influence functions". In: *International Conference on Machine Learning*. PMLR, pp. 1885–1894.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25.
- Lajus, Jonathan, Luis Galárraga, and Fabian Suchanek (2020). "Fast and exact rule mining with AMIE 3". In: *Proceedings of the Extended Semantic Web Conference*. Springer, pp. 36–52.
- Lancaster, HO (1963). "Correlation and complete dependence of random variables". In: *The Annals of Mathematical Statistics* 34.4, pp. 1315–1321.
- Lawrence, Carolin, Timo Sztyler, and Mathias Niepert (2021). "Explaining neural matrix factorization with gradient rollback". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. AAAI Press, pp. 4987–4995.
- Lehmann, Jens et al. (2015). "Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia". In: *Semantic web* 6.2, pp. 167–195.
- Lerer, Adam et al. (2019). "Pytorch-biggraph: A large scale graph embedding system". In: *Proceedings of Machine Learning and Systems* 1, pp. 120–131.
- Lewis, Patrick et al. (2020). "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in neural information processing systems* 33, pp. 9459–9474.
- Liang, Xinyu et al. (2024). "A survey of inductive knowledge graph completion". In: *Neural Computing and Applications* 36.8, pp. 3837–3858.
- Lin, Yankai et al. (2015). "Learning entity and relation embeddings for knowledge graph completion". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 29. 1.
- Liu, Shuwen et al. (2023). "Revisiting inferential benchmarks for knowledge graph completion". In: *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 461–471.
- Lundberg, Scott M and Su-In Lee (2017). "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30.
- Lundberg, Scott M et al. (2020). "From local explanations to global understanding with explainable AI for trees". In: *Nature machine intelligence* 2.1, pp. 56–67.
- Mahdisoltani, Farzaneh, Joanna Biega, and Fabian M. Suchanek (2015). "YAGO3: A Knowledge Base from Multilingual Wikipedias". In: *Seventh Biennial Conference on Innovative Data Systems Research, CIDR*. www.cidrdb.org.
- Mayer, Marta Cialdea and Fiora Pirri (1993). "First order abduction via tableau and sequent calculi". In: *Logic Journal of the IGPL* 1.1, pp. 99–117.

- Megiddo, Nimrod and Ramakrishnan Srikant (1998). "Discovering Predictive Association Rules". In: *KDD 1998*. Ed. by Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro. AAAI Press, pp. 274–278.
- Meilicke, Christian, Patrick Betz, and Heiner Stuckenschmidt (2021). "Why a naive way to combine symbolic and latent knowledge base completion works surprisingly well". In: *3rd Conference on Automated Knowledge Base Construction*.
- Meilicke, Christian et al. (2018). "Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion". In: *Proceedings of the International Semantic Web Conference*. Springer, pp. 3–20.
- Meilicke, Christian et al. (2019). "Anytime bottom-up rule learning for knowledge graph completion". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, pp. 3137–3143.
- Meilicke, Christian et al. (2024). "Anytime bottom-up rule learning for large-scale knowledge graph completion". In: *The VLDB Journal* 33.1, pp. 131–161.
- Miller, George A (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.
- Minervini, Pasquale et al. (2020). "Differentiable reasoning on large knowledge bases and natural language". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 5182–5190.
- Muggleton, Stephen (1991). "Inductive logic programming". In: *New generation computing* 8, pp. 295–318.
- (1996). "Stochastic logic programs". In: *Advances in inductive logic programming* 32, pp. 254–264.
- Muggleton, Stephen and Luc De Raedt (1994). "Inductive logic programming: Theory and methods". In: *The Journal of Logic Programming* 19, pp. 629–679.
- Nguembang Fadja, Arnaud and Fabrizio Riguzzi (2019). "Lifted discriminative learning of probabilistic logic programs". In: *Machine Learning* 108.7, pp. 1111–1135.
- Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel (2011). "A three-way model for collective learning on multi-relational data". In: *International Conference on Machine Learning*. Vol. 11, pp. 809–816.
- Nickel, Maximilian et al. (2015). "A review of relational machine learning for knowledge graphs". In: *Proceedings of the IEEE* 104.1, pp. 11–33.
- Niepert, Mathias, Pasquale Minervini, and Luca Franceschi (2021). "Implicit MLE: Backpropagating through discrete exponential family distributions". In: *Advances in Neural Information Processing Systems* 34, pp. 14567–14579.
- Noessner, Jan, Mathias Niepert, and Heiner Stuckenschmidt (2013). "RockIt: Exploiting parallelism and symmetry for MAP inference in statistical relational models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 27. 1.
- Omran, Pouya Ghasnezhad, Kewen Wang, and Zhe Wang (2018). "Scalable rule learning via learning representation". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 2149–2155.
- Ortona, Stefano, Venkata Vamsikrishna Meduri, and Paolo Papotti (2018). "Robust discovery of positive and negative rules in knowledge bases". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, pp. 1168–1179.
- Ott, Simon, Christian Meilicke, and Matthias Samwald (2021). "SAFRAN: An interpretable, rule-based link prediction method outperforming embedding models". In: *3rd Conference on Automated Knowledge Base Construction*.
- Ott, Simon et al. (2023). "Rule-based knowledge graph completion with canonical models". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 1971–1981.

- Pearl, Judea (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.
- Pellissier Tanon, Thomas, Gerhard Weikum, and Fabian Suchanek (2020). “Yago 4: A reason-able knowledge base”. In: *Proceedings of the Extended Semantic Web Conference*. Springer, pp. 583–596.
- Pezeshkpour, Pouya, Yifan Tian, and Sameer Singh (June 2019). “Investigating robustness and interpretability of link prediction via adversarial modifications”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.
- Pirró, Giuseppe (2020). “Relatedness and TBox-driven rule learning in large knowledge bases”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, pp. 2975–2982.
- Pogančić, Marin Vlastelica et al. (2020). “Differentiation of blackbox combinatorial solvers”. In: *International Conference on Learning Representations*. OpenReview.net.
- Portisch, Jan, Nicolas Heist, and Heiko Paulheim (2022). “Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction—two sides of the same coin?” In: *Semantic Web 13.3*, pp. 399–422.
- Quinlan, J. Ross (1990). “Learning logical definitions from relations”. In: *Machine learning* 5, pp. 239–266.
- Ren, Shaoqing et al. (2016). “Faster R-CNN: Towards real-time object detection with region proposal networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.6, pp. 1137–1149.
- Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). “Why should i trust you? Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.
- Richardson, Matthew and Pedro Domingos (2006). “Markov logic networks”. In: *Machine learning*. Vol. 62. 1-2. Springer, pp. 107–136.
- Riguzzi, Fabrizio (2016). “The distribution semantics for normal programs with function symbols”. In: *International Journal of Approximate Reasoning* 77, pp. 1–19.
- Riguzzi, Fabrizio et al. (2017). “A survey of lifted inference approaches for probabilistic logic programming under the distribution semantics”. In: *International Journal of Approximate Reasoning* 80, pp. 313–333.
- Rim, Wiem Ben et al. (2021). “Behavioral testing of knowledge graph embedding models for link prediction”. In: *3rd Conference on Automated Knowledge Base Construction*.
- Rocktäschel, Tim and Sebastian Riedel (2017). “End-to-end differentiable proving”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 3788–3800.
- Rolínek, Michal et al. (2020). “Optimizing rank-based metrics with blackbox differentiation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7620–7630.
- Rossi, Andrea et al. (2021). “Knowledge graph embedding for link prediction: A comparative analysis”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15.2, pp. 1–49.
- Ruffinelli, Daniel, Samuel Broscheit, and Rainer Gemulla (2020). “You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings”. In: *8th International Conference on Learning Representations*.
- Sadeghian, Ali et al. (2019). “DRUM: End-To-End differentiable rule mining on knowledge graphs”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 15321–15331.

- Safavi, Tara and Danai Koutra (2020). "CoDEX: A Comprehensive Knowledge Graph Completion Benchmark". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 8328–8350.
- Sato, Taisuke and Yoshitaka Kameya (1997). "PRISM: a language for symbolic-statistical modeling". In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. Vol. 97, pp. 1330–1339.
- Schlichtkrull, Michael et al. (2018). "Modeling relational data with graph convolutional networks". In: *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer, pp. 593–607.
- Shchur, Oleksandr et al. (2018). "Pitfalls of graph neural network evaluation". In: *Workshop on Relational Representation Learning, Neural Information Processing Systems*.
- Srinivasan, Ashwin (2000). *The Aleph Manual (Technical Report)*. Tech. rep. Computing Laboratory, Oxford University.
- Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum (2007). "Yago: A core of semantic knowledge". In: *Proceedings of the 16th International Conference on World Wide Web*. Association for Computing Machinery, 697–706.
- Suchanek, Fabian M et al. (2019). "Knowledge representation and rule mining in entity-centric knowledge bases". In: *Reasoning Web. 15th International Summer School, Tutorial Lectures*, pp. 110–152.
- Suchanek, Fabian M et al. (2024). "Yago 4.5: A large and clean knowledge base with a rich taxonomy". In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 131–140.
- Svatoš, Martin et al. (2020). "STRiKE: Rule-driven relational learning using stratified k-entailment". In: *Proceedings of the European Conference of Artificial Intelligence*.
- Tabacof, Pedro and Luca Costabello (2020). "Probability calibration for knowledge graph embedding models". In: *8th International Conference on Learning Representations*. OpenReview.net.
- Taisuke, SATO (1995). "A statistical learning method for logic programs with distribution semantics". In: *Proceedings of ICLP*. Citeseer, pp. 715–729.
- Tanon, Thomas Pellissier et al. (2017). "Completeness-Aware rule learning from knowledge graphs". In: *International Joint Conference on Artificial Intelligence*. Ijcai.org, pp. 507–525.
- Tena Cucala, David J., Bernardo Cuenca Grau, and Boris Motik (Aug. 2022). "Faithful approaches to rule learning". In: *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 484–493.
- Toutanova, Kristina and Danqi Chen (July 2015). "Observed versus latent features for knowledge base and text inference". In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Beijing, China: Association for Computational Linguistics, pp. 57–66.
- Trouillon, Théo et al. (2016). "Complex embeddings for simple link prediction". In: *International Conference on Machine Learning*, pp. 2071–2080.
- Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing system* 30.
- Vrandečić, Denny and Markus Krötzsch (2014). "Wikidata: a free collaborative knowledgebase". In: *CACM* 57.10, pp. 78–85.
- Walker, Nicholas Thomas, Stefan Ultes, and Pierre Lison (2023). "Retrieval-augmented neural response generation using logical reasoning and relevance scoring". In: *Proceedings of the 27th Workshop on the Semantics and Pragmatics of Dialogue*.

- Wang, Yanjie et al. (2018). "On evaluating embedding models for knowledge base completion". In: *4th Workshop on Representation Learning for NLP (Rep4NLP@ACL)*.
- Wang, Zhen et al. (2014). "Knowledge graph embedding by translating on hyperplanes". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 28. 1.
- Wu, Hong et al. (2022). "Learning typed rules over knowledge graphs". In: *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 494–503.
- Xue, Bingcong and Lei Zou (2022). "Knowledge graph quality management: a comprehensive survey". In: *IEEE Transactions on Knowledge and Data Engineering* 35.5, pp. 4969–4988.
- Yang, Bishan et al. (2015). "Embedding entities and relations for learning and inference in knowledge bases". In: *3rd International Conference on Learning Representations*.
- Yang, Fan, Zhilin Yang, and William W. Cohen (2017). "Differentiable learning of logical rules for knowledge base reasoning". In: *Advances in Neural Information Processing Systems*, pp. 2319–2328.
- Yu, Tong and Hong Zhu (2020). "Hyper-parameter optimization: A review of algorithms and applications". In: *arXiv preprint arXiv:2003.05689*.
- Zhang, Hengtong et al. (July 2019). "Data poisoning attack against knowledge graph embedding". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Ijcai.org, pp. 4853–4859.
- Zhang, Yichi et al. (May 2024). "Unleashing the power of imbalanced modality information for multi-modal knowledge graph completion". In: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 17120–17130.
- Zhu, Zhaocheng et al. (2021). "Neural bellman-ford networks: A general graph neural network framework for link prediction". In: *Advances in Neural Information Processing Systems* 34, pp. 29476–29490.
- Zhu, Zhaocheng et al. (2024). "A* net: A scalable path-based reasoning approach for knowledge graphs". In: *Advances in Neural Information Processing Systems* 36.
- Zupanc, Kaja and Jesse Davis (2018). "Estimating rule quality for knowledge base completion with the relationship between coverage assumption". In: *Proceedings of the 2018 World Wide Web Conference*, pp. 1073–1081.

Appendix A

Appendix

A.1 Proofs For Chapter 5

In the following, we provide the proofs for Chapter 5. We use the general notation used within the thesis and the notation specifically introduced in Section 5.2.1. Additionally, for brevity, we write $r_i \models_1 f$ for $\mathcal{G} \cup \{r_i\} \models_1 f$, i.e., the rule r predicts target fact f with respect to KG \mathcal{G} .

Proof of Proposition 1.

We show that to perform inference $P(f|\mathcal{G})$ with the probabilistic model, it is sufficient to perform marginal inference with respect to the rules that predict the target fact f . First, we plug in equation (5.3) into equation (5.5).

$$P(f|\mathcal{G}) = \sum_{z \in \{0,1\}^N} P(f|\mathbf{R}=z, \mathcal{G})P(\mathbf{R}=z|\mathcal{G}) \quad (\text{A.1})$$

$$= \sum_{\substack{z \in \{0,1\}^N \\ L_I(z) \models_1 f}} P(\mathbf{R}=z|\mathcal{G}) \quad (\text{A.2})$$

The target probability is the sum of all possible configurations in which f is one-step entailed by the subset of rules that are true.

We will now split the random vector \mathbf{R} of all rules and its realisations $z \in \{0,1\}^N$ into two vectors. The rules that predict the target $z^+ = (z_j)_{j \in I, r_j \models_1 f}$ and the rules that do not predict the target $z^- = (z_j)_{j \in I, r_j \not\models_1 f}$. Let $z^- || z^+ = z$ where $||$ denotes vector concatenation. For brevity, we will write $P(z|\mathcal{G})$ for $P(\mathbf{R}=z|\mathcal{G})$ in the following. Then, we can write equation (A.2) as:

$$\sum_{\substack{z^+ || z^- \in \{0,1\}^N \\ L_I(z^+ || z^-) \models_1 f}} P(z^+, z^-|\mathcal{G}) \quad (\text{A.3})$$

$$= \sum_{\substack{z^+ \in \{0,1\}^k \\ L_I(z^+) \models_1 f}} \sum_{\substack{z^- \in \{0,1\}^{N-k} \\ L_I(z^-) \not\models_1 f}} P(z^+, z^-|\mathcal{G}). \quad (\text{A.4})$$

Observe that the inner sum contains all possible values of z^- as $L_I(z^-) \not\models_1 f$ does not put a constraint on z^- . Continuing from equation (A.4), we can therefore apply reverse marginalization,

$$\begin{aligned}
&= \sum_{\substack{z^+ \in \{0,1\}^k \\ L_I(z^+) \models f}} P(z^+ | \mathcal{G}) \\
&= \sum_{z_f \in \{0,1\}^k} P(f | \mathbf{R}_f = z_f, \mathcal{G}) P(\mathbf{R}_f = z_f | \mathcal{G}). \quad \square
\end{aligned}$$

We show the procedure in an example with three rules where two of them predict the target.

Example 25. Let $\mathcal{R} = \{r_1, r_2, r_3\}$ and assume that r_1 and r_2 predict target f but r_3 does not predict f . For the target probability, we have:

$$\begin{aligned}
P(f | \mathcal{G}) &= \sum_{z \in \{0,1\}^N} P(f | \mathbf{R} = z, \mathcal{G}) P(\mathbf{R} = z | \mathcal{G}) \\
&= 0 \cdot P(R_1=0, R_2=0, R_3=0 | \mathcal{G}) + 0 \cdot P(R_1=0, R_2=0, R_3=1 | \mathcal{G}) \\
&\quad + 1 \cdot P(R_1=0, R_2=1, R_3=0 | \mathcal{G}) + 1 \cdot P(R_1=0, R_2=1, R_3=1 | \mathcal{G}) \\
&\quad + 1 \cdot P(R_1=1, R_2=0, R_3=0 | \mathcal{G}) + 1 \cdot P(R_1=1, R_2=0, R_3=1 | \mathcal{G}) \\
&\quad + 1 \cdot P(R_1=1, R_2=1, R_3=0 | \mathcal{G}) + 1 \cdot P(R_1=1, R_2=1, R_3=1 | \mathcal{G})
\end{aligned}$$

Now we will restructure the sum according to equation (A.4) and drop the terms that are zero:

$$\begin{aligned}
&= \left(P(R_1=1, R_2=0, R_3=1 | \mathcal{G}) + P(R_1=1, R_2=0, R_3=0 | \mathcal{G}) \right) \\
&\quad + \left(P(R_1=1, R_2=1, R_3=1 | \mathcal{G}) + P(R_1=1, R_2=1, R_3=0 | \mathcal{G}) \right) \\
&\quad + \left(P(R_1=0, R_2=1, R_3=1 | \mathcal{G}) + P(R_1=0, R_2=1, R_3=0 | \mathcal{G}) \right) \\
&= P(R_1=1, R_2=0 | \mathcal{G}) + P(R_1=1, R_2=1 | \mathcal{G}) + P(R_1=0, R_2=1 | \mathcal{G})
\end{aligned}$$

The last expression is the probability that at least one of the predicting rules is true, as described in the main text of the thesis. Finally, note that only r_3 can be marginalized out because it does not predict the target.

Proof of Theorem 1

This proof is a generalisation of the case with two rules from section 5.2.4. We first show that under maximal correlation only very specific realisations of \mathbf{R} can have non-zero probability if the distribution exists. Once this is established, we show the existence and uniqueness of $P(\mathbf{R} | \mathcal{G})$ and finally we derive the max-aggregation score from the marginal inference that at least one of the predicting rules is true. As previously we assume that k rules out of N predict the target fact f and that the rule marginals are given. The index set I indexes all rules and $I_f \subseteq I$ indexes the rules that predict target f where the reference to the KG is implicit. Moreover, \mathbf{R}_f is the random vector representing the k rules that predict f .

After we have specified $P(\mathbf{R} | \mathcal{G})$, by Proposition 1 and 2, we have to show that

$$1 - P(\mathbf{R}_f = \mathbf{0} | \mathcal{G}) = \max \{ P(R_j = 1) : r_j \models f \text{ and } j \in I_f \}, \quad (\text{A.5})$$

where $\mathbf{0}$ is the zero vector. We assume throughout the derivations the N variables $\{R_1, \dots, R_N\}$ are ordered by I (and likewise for I_f) such that R_1 is the rule with the highest marginal.

First we pick two rules represented by R_i, R_j with $j > i$ and $i, j \in I$. The correlation is defined as:

$$\rho_{ij} = \frac{\pi_{ij} - \pi_i \pi_j}{(\pi_i(1 - \pi_i)\pi_j(1 - \pi_j))^{1/2}} \quad (\text{A.6})$$

Where $\pi_{ij} = P(R_i=1, R_j=1|\mathcal{G})$ and, e.g., $\pi_i = P(R_i=1|\mathcal{G})$ is the rule marginal for i . We now make the assumption required by the theorem, that is, we assume that $\rho_{ij} = U(i, j)$ for every pair i, j from $I \times I$, where $U(i, j)$ is the upper correlation bound from Section 5.2.4. Therefore, we plug in $U(i, j)$ in the left-hand side of equation (A.6) and solve for π_{ij} which leads to

$$\pi_{ij} = \pi_j, \quad (\text{A.7})$$

i.e., we have the equality $P(R_i=1, R_j=1|\mathcal{G}) = P(R_j=1|\mathcal{G})$. After using marginalization, we can write this as

$$\begin{aligned} & \sum_{\mathbf{z}_{-ij} \in \{0,1\}^{N-2}} P(R_i=1, R_j=1, \mathbf{R}_{-ij}=\mathbf{z}_{-ij} | \mathcal{G}), \\ &= \sum_{\mathbf{z}_{-j} \in \{0,1\}^{N-1}} P(R_j=1, \mathbf{R}_{-j}=\mathbf{z}_{-j} | \mathcal{G}), \end{aligned} \quad (\text{A.8})$$

where $\mathbf{z}_{-j} = (z_i)_{i \in I \setminus j}$ is a vector of realisations with the j 'th component dropped from \mathbf{R} and equivalently $\mathbf{z}_{-ij} = (z_s)_{s \in I \setminus \{i,j\}}$ has both components dropped. Each addend in the left hand side is contained in the right hand side. Therefore, subtracting the left hand side from both sides of (A.8) yields a zero probability constraint:

$$0 = \sum_{\mathbf{z}_{-ij} \in \{0,1\}^{N-2}} P(R_i=0, R_j=1, \mathbf{z}_{-ij} | \mathcal{G}). \quad (\text{A.9})$$

As probabilities cannot be negative, each of the term in the sum must be zero. These constraint follow a clear structure. We picked the two rules with $j > i$ and therefore the marginal of rule i is larger than the marginal of rule j . The upper bound $U(i, j)$ contains a min operator and therefore depends on which rule has a smaller marginal. In fact, the zero constraint of equation A.9 has a clear interpretation: A configuration of truth values for the rules is impossible (has zero probability) in which a rule with a higher marginal is zero while a rule with a lower marginal is one.

We are, in fact, interested in all the realisations that may be different from zero after considering the constraints imposed by all possible rule pairs. From equation (A.9) and the rule ordering according to their marginals it follows that $P(\mathbf{R}|\mathcal{G})$ is not affected by the zero-constraint if

$$(R_s = 0) \implies (R_t = 0) \quad \forall t > s, \text{ with } s, t \in I. \quad (\text{A.10})$$

Note that each realisation $\mathbf{z} \in \{0, 1\}^N$ of \mathbf{R} which satisfies (A.10) is associated with a unique number of components (rules) that are one.

We will now focus on the probability configuration that can be non-zero. From (A.10), we can observe that there are only $N + 1$ of these configurations left and we will therefore introduce $N + 1$ variables.

Let $m \in \{0, 1, \dots, N\}$ and let g_m denote the variable for the probability for the configuration of realisations $\mathbf{z}^{(m)} \in \{0, 1\}^N$ of \mathbf{R} that contains m ones and satisfies (A.10). That is, we have $g_m = P(\mathbf{R} = \mathbf{z}^{(m)})$. In fact, $\mathbf{z}^{(m)} \in \{0, 1\}^N$ contains ones in the first entries and zeros up from the $m + 1$ 'th component until the last component of the vector due to the ordering of the rules. The realisation with $m = 0$ contains only zeros.

By requirement of the theorem, the marginals $\pi_i = P(R = 1|\mathcal{G})$ are given. Moreover, it is easy to verify that it must hold that $\pi_i = \sum_{s=i}^N g_s$ which gives us one equation for each of the N marginals. Additionally, we use the probability constraint $\sum_{s=0}^N g_s = 1$. Therefore, we can set up an equation system for the $N + 1$ variables:

$$A\mathbf{g} = \mathbf{\Pi}, \quad (\text{A.11})$$

In the equation system, A is an upper triangular coefficient matrix with all non-zero entries being one, $\mathbf{g} = (g_s)_{s=0}^N$ is the variable vector and $\mathbf{\Pi}$ is a vector that contains 1 at the first component and subsequently the N marginals. Given that A is invertible we established uniqueness and we established existence as the solution $\mathbf{g} = A^{-1}\mathbf{\Pi}$ satisfies the probability constraint $\sum_{s=0}^N g_s = 1$ while all g_m are between 0 and 1.

We will now derive the main result from (A.5). Plugging in the expression $\pi_i = \sum_{s=i}^N g_s$ for the marginals into the right-hand side of (A.5) yields

$$\max \left\{ \sum_{s=j}^N g_s : r_j \models_1 f \text{ and } j \in I_f \right\} = \sum_{s=s^*}^N g_s, \quad (\text{A.12})$$

where $s^* = \min\{j : r_j \models_1 f \text{ and } j \in I_f\}$ corresponds to the index for the rule with the highest marginal under the predicting rules.

For $1 - P(\mathbf{R}=\mathbf{0}|\mathcal{G})$ we have to sum up all probabilities of realisations where at least one of the predicting rules is one. By the structure of the constraint (A.10), in each of these realisations that has a non-zero probability, R_{s^*} must be one. On the other hand, in each realisation where R_{s^*} is one at least one rule predicts the target because R_{s^*} predicts the target. Given that the remaining probabilities are zero, we have that $\sum_{s=s^*}^N g_s = 1 - P(\mathbf{R} = \mathbf{0}|\mathcal{G})$. \square

We will conclude this section with an example where we calculate the full distribution based on the marginals and the assumptions made by the theorem in the case of three rules.

Example 26. Let $\mathcal{R} = \{r_1, r_2, r_3\}$ and assume the given marginals $\pi_1 = 0.9$, $\pi_2 = 0.7$, and $\pi_3 = 0.5$.

We show all the 2^3 probabilities in Table A.1. The specification leads to the equation system:

$$A\mathbf{g} = \mathbf{\Pi} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 1 \\ \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix}$$

Which we can solve with the inverse.

$$\begin{bmatrix} g_0^* \\ g_1^* \\ g_2^* \\ g_3^* \end{bmatrix} = A^{-1}\Pi = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix} = \begin{bmatrix} 1 - \pi_1 \\ \pi_1 - \pi_2 \\ \pi_2 - \pi_3 \\ \pi_3 \end{bmatrix}$$

Together with the zero probability constraints of the table, we fully specified the 2^3 values of the probability distribution. It is easy to verify that they sum to 1. Additionally, when calculating the target probability from the main text based on any subset of predicting rules, it will result in Max-aggregation of the marginals. For instance, assume rule two and three predict the target, then we need to calculate the probability that one of the rules is true based on the joint distribution. Summing all the realisations in which at least one of the rules is one (ignoring the 0 probability realisations) results in $g_2^* + g_3^* = \pi_2 = \max\{\pi_2, \pi_3\}$.

$P(\mathbf{R} = \mathbf{z} \mathcal{G})$	R_1	R_2	R_3	Realisation
g_0	0	0	0	$\mathbf{z}^{(0)}$
0	0	0	1	-
0	0	1	0	-
0	0	1	1	-
g_1	1	0	0	$\mathbf{z}^{(1)}$
0	1	0	1	-
g_2	1	1	0	$\mathbf{z}^{(2)}$
g_3	1	1	1	$\mathbf{z}^{(3)}$

TABLE A.1: All probability values.

Proof of Proposition 3

We directly start with the result from Proposition 1 to show that the inference model leads to the Noisy-or product:

$$P(f|\mathcal{G}) = \sum_{\mathbf{z}_f \in \{0,1\}^k} P(f|\mathbf{R}_f=\mathbf{z}_f, \mathcal{G})P(\mathbf{R}_f=\mathbf{z}_f|\mathcal{G}) \quad (\text{A.13})$$

$$= 0 \cdot P(\mathbf{R}_f=\mathbf{0}|\mathcal{G}) + \sum_{\substack{\mathbf{z}_f \in \{0,1\}^k \\ \mathbf{z}_f \neq \mathbf{0}}} P(\mathbf{R}_f=\mathbf{z}_f|\mathcal{G}) \quad (\text{A.14})$$

$$= 1 - P(\mathbf{R}_f=\mathbf{0}|\mathcal{G}) \quad (\text{A.15})$$

$$= 1 - \prod_{j \in I_f} (1 - P(R_j=1)) \quad \square$$

Proof of Proposition 5

Let \mathcal{G} be a KG and \mathcal{R} be a set of rules. Let \mathbf{R} be the random vector representing the N rules and let $\mathbf{z} \in \{0,1\}^N$ denote a vector of realisations. We are given the ProbLog program \mathcal{T} based on the rules and the KG as defined in the main text. Observe that the map $L(\mathbf{z})$ defined in equation (5.2) simply collects a logic program that is the

union of the KG and the rules that are one in \mathbf{z} . Therefore, using our notation, we can write the target probability calculated with ProbLog as

$$P(f|\mathcal{T}) = \sum_{\mathbf{z} \in \{0,1\}^N} \hat{P}(f|L(\mathbf{z}))P(L(\mathbf{z})|\mathcal{T}), \quad (\text{A.16})$$

where

$$\hat{P}(f|L(\mathbf{z})) = \begin{cases} 1, & \text{if } L(\mathbf{z}) \models t \\ 0, & \text{else.} \end{cases}$$

Furthermore, $P(L(\mathbf{z})|\mathcal{T})$ is the probability of a logic program as defined in equation (5.15) in the main text. The target probability is the sum over all logic programs that entail the target, which is also termed the probability that the target has a proof (De Raedt, Kimmig, and Toivonen, 2007).

We will now label different realisations of \mathbf{R} according to their logical properties. Let $\mathbf{z}^{(e)} \in \{0,1\}^N$ be a vector of realisations such that $L(\mathbf{z}^{(e)})$ entails but not one-step entails the target and let $\mathbf{z}^{(o)} \in \{0,1\}^N$ be the corresponding vector where $L(\mathbf{z}^{(o)})$ entails and one-step entails the target. Then, we can write equation (A.16) as:

$$\begin{aligned} P(t|\mathcal{T}) &= \sum_{\substack{\mathbf{z} \in \{0,1\}^N \\ L(\mathbf{z}) \models t}} P(L(\mathbf{z})|\mathcal{T}) \\ &= \sum_{\substack{\mathbf{z}^{(e)} \in \{0,1\}^N \\ L(\mathbf{z}^{(e)}) \models t \\ L(\mathbf{z}^{(e)}) \not\models_1 t}} P(L(\mathbf{z}^{(e)})|\mathcal{T}) + \sum_{\substack{\mathbf{z}^{(o)} \in \{0,1\}^N \\ L(\mathbf{z}^{(o)}) \models_1 t}} P(L(\mathbf{z}^{(o)})|\mathcal{T}) \\ &\geq \sum_{\substack{\mathbf{z}^{(o)} \in \{0,1\}^N \\ L(\mathbf{z}^{(o)}) \models_1 t}} P(L(\mathbf{z}^{(o)})|\mathcal{T}) \end{aligned}$$

The last expression is equal to the Noisy-or product over the rules that predict the target. To see this, we can plug in the probability of a logic program defined in equation (5.15) in the main chapter.

$$\sum_{\substack{\mathbf{z}^{(o)} \in \{0,1\}^N \\ L(\mathbf{z}^{(o)}) \models_1 t}} P(L(\mathbf{z}^{(o)})|\mathcal{T}) \quad (\text{A.17})$$

$$= \sum_{\substack{\mathbf{z}^{(o)} \in \{0,1\}^N \\ L(\mathbf{z}^{(o)}) \models_1 t}} \prod_{x_i \in L(\mathbf{z}^{(o)})} P(x_i) \prod_{x_j \notin L(\mathbf{z}^{(o)})} (1 - P(x_j)) \quad (\text{A.18})$$

The terms in the sum describe the full factorisation of the logic programs that one-step entail the target under the mutual independence assumption. The terms $P(x_i)$ are the rule confidences by requirement of the proposition if x_i is a rule. For the x_i corresponding to facts, we have $P(x_i) = 1$ when $x_i \in L(\mathbf{z}^{(o)})$ and $P(x_i) = 0$ otherwise. That is, the fact probabilities do not alter the expression. Therefore, the expression is the forward direction of the Noisy-or product based on all the rules. Finally, we

know by the proof of Proposition 1 that the confidences of the rules not predicting the target will marginalize out. \square

An example regarding ProbLog and Noisy-or aggregation that shows the individual logic programs and their probabilities to be summed up is shown in the main text of this thesis in Example 18.

A.2 Ablations for Chapter 6

Figure A.2 illustrates an ablation for the sparse-aggregator during Mean-Rank (MR) training, which is discussed in Section 6.3.3. The right-hand plot compares the validation MRR on Codex-M when using MR training and BCE training. MR training performs better than BCE training for the sparse-aggregator, leading to improvements over using pre-computed rule confidences. The left plot contrasts training with scaled gradients against the vanilla formulation. Although the used benchmark was not introduced in the main text of this thesis due to its limited size and high variance, generally, scaled gradients exhibit lower variance.

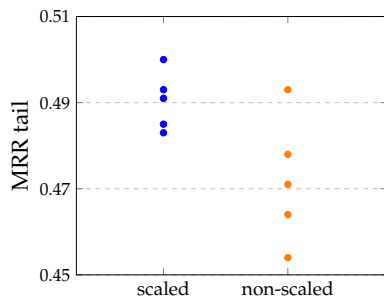


FIGURE A.1: MRR in tail direction for five runs when training under the scaled/non-scaled gradient.

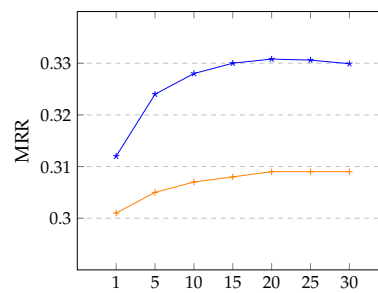


FIGURE A.2: Valid MRR per epoch on Codex-M for Mean-Rank training (★) and ordinary Cross-Entropy loss (+).