TU WIEN Informatics

Rule Mining on Knowledge Graph Embeddings

Making Implicit Knowledge Explicit and Explainable

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Julian Vecera, BSc BSc
Matrikelnummer 01627770

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Prof. Dr. Emanuel Sallinger
Mitwirkung: Dipl.-Ing. Aleksandar Pavlović

Wien, 20. August 2024

Julian Vecera	Emanuel Sallinger

TU WIEN Informatics

Rule Mining on Knowledge Graph Embeddings

Making Implicit Knowledge Explicit and Explainable

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Julian Vecera, BSc BSc
Registration Number 01627770

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Dr. Emanuel Sallinger
Assistance: Dipl.-Ing. Aleksandar Pavlović

Vienna, 20th August, 2024

Julian Vecera	Emanuel Sallinger

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Julian Vecera, BSc BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 20. August 2024

Julian Vecera

Kurzfassung

Die Erklärbarkeit in maschinell gelernten Modellen, wie etwa bei der Verwendung von Wissensgraphen, stellt nach wie vor ein Problem dar. Ansätze wie die Einbettung dieser Modelle können implizites Wissen lernen, funktionieren jedoch oft wie Black-Box-Modelle. Diese Arbeit zielt darauf ab, das latente Wissen explizit zu machen und dadurch eine Form von Erklärbarkeit für die Schlussfolgerungen dieser Modelle zu schaffen, indem intuitiv verständliche Regeln bereitgestellt werden.

Dies wird durch den neuartigen „Rule Mining"-Algorithmus ExpressivE RM erreicht, der in dieser Arbeit vorgestellt wird. Der Algorithmus kombiniert die Vorteile von Regelgenerierungsalgorithmen, wie etwa deren Erklärbarkeit, und das erlernte latente Wissen und die hohe Vorhersagekraft von eingebetteten Wissensgraphen.

ExpressivE RM basiert auf dem zugrunde liegenden Modell ExpressivE und verwendet eine zusätzliche Abstraktionsschicht, um Regeln aus dem Basismodell zu extrahieren. Dies stellt einen bedeutenden Schritt auf dem Weg zu erklärbaren Schlussfolgerungen des Basismodells und ähnlicher Modelle dar.

Die hier präsentierten Ergebnisse zeigen, wie ExpressivE RM explizite Regeln erstellt, die zur Vervollständigung des Wissensgraphen verwendet werden können. Dabei erreicht der Algorithmus eine vergleichbare Leistung wie der „State-of-the-Art"-Algorithmus AnyBURL. Unser Algorithmus erzeugt zusätzlich Regeln, die fehlende Verbindungen im Graphen ergänzen können, welche von AnyBURL nicht abgedeckt werden. Dies zeigt, dass der Ansatz in der Lage ist, exklusive Regeln und zusätzliches latentes Wissen zu liefern.

Die Leistungswerte von ExpressivE RM betragen hits@1, hits@3, hits@10 und einen MRR von .3226 für das vollständige Set an Regeln und .2773 für das exklusive Set an Regeln. Dies beweist, dass die geometrische Interpretation von ExpressivE verwendet werden kann, um ein Set aus Regeln zu erzeugen, das fehlende Verbindungen im Wissensgraph mit hoher Vorhersagekraft in kurzer Zeit herstellen kann.

Abstract

Explainability in knowledge graph-based machine learning algorithms is a problem because approaches such as knowledge graph embeddings can learn implicit knowledge but act like black box models. This work aims to make the latent knowledge explicit and, therefore, explainable by providing intuitive rules.

This is achieved through the novel ExpressivE RM algorithm, a unique approach that combines the benefits of rule miners' explainability and knowledge graph embeddings' latent knowledge and high performance. The algorithm, named after its base KGE model with an additional abstraction layer to mine rules from the embedding, is a significant step towards making knowledge graph-based machine learning algorithms more explainable.

Our research shows that the ExpressivE RM algorithm performs at a comparable level to the state-of-the-art rule miner AnyBURL, with its own distinct advantages. We created a rule set of symmetry rules that competes with AnyBURL's rules, and our evaluation showed that our algorithm provides rules of strong performance in knowledge graph completion tasks, which are not covered by AnyBURL. This indicates that our approach can provide exclusive rules and additional (latent) knowledge, with scores of hits@1, hits@3, hits@10 and MRR of .3226 for the complete rule set and .2773 for the exclusive rule set.

ExpressivE RM proves that the geometric interpretation of ExpressivE can provide a performant rule set for prediction tasks with latent knowledge available and it provides explainability using rules in a short time.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

In recent years, one topic has caused much attention in the car industry, finance, governments, and even the medical sector. It is said to be a topic that will make a big difference in the following decades, yet the discipline is familiar, and scientists have been researching it for years. The topic this work will address differs from the currently discussed climate change, even though the technology can help in the fight. It can potentially find or improve cures for many diseases in the medical domain. The following work will go deep into computer science and statistics, addressing techniques essential for survival that are older than humankind itself, namely learning and pattern recognition. The topic this work will address is machine learning and artificial intelligence.

Two heavily used buzzwords in the last few years have arisen with the increasing computing power of modern machines and the potential that comes with them. Even though machine learning arose in the 1960s [Fra20], considerable interest is coming up in the discipline, which is successfully used in self-driving cars [FHY19], image recognition, and even healthcare [Kon01].

Modern machine learning algorithms are already capable of beating humans in various games [GBP17], face-recognition, and even in detecting skin cancers based on images [EKN+17], yet they are not perfect and often provide excellent results, but lack explainability, which is crucial for interpreting the algorithms reasoning, especially when used in critical situations.

This work focuses, in particular, on knowledge graphs. A knowledge graph consists of nodes and edges. The edges connect nodes and represent relations [SGT+09]. This straightforward concept is capable of holding different kinds of information. An example is two people, each represented by a node, which have a relation to each other. They are married (undirected edge, two-way-relationship), or one person is the child of the other

(directed edge, one-way-relationship). This information is stored as triple, containing both entities and their relationship. This method can also be used to represent other kinds of information, like transactions in the finance sector, connections in social networks, and knowledge generally because the information is basically always linked to any kind of other information in a specific way.

The resulting graphs can be used to learn from and predict new triples that are not already given but may exist in the real world. This is a valuable technique used in social networks to recommend new friends [Liu13] or in medicine to find new treatments.

Finding new triples in this context is called knowledge graph completion (KGC). It builds on the observation that knowledge graphs are often inherently incomplete [PS23]. For example, only 25 % of the people modeled in the Freebase dataset have their nationality assigned [WGM+14]. However, the knowledge graph offers information to predict it for each person to a certain level of likeliness. The use of already-known information, such as the language a person speaks, can help to fill in the missing information. A very promising approach to make this latent coherence tangible is the use of knowledge graph embedding models (KGEs), which embed entities and relations of a given knowledge graph in a way that makes it possible to quantify the likelihood of the real-world-existence of yet unknown triples by computing scores using the learned embedding [WMWG17]. However, machine learning models are black box models, and it is still more challenging to explain the reasoning of the model in a way that logical reasoning-based models could offer.

Another knowledge graph completion approach is called rule mining. The focus here lies on explainability and overcoming the just-mentioned limitations of KGE models. Nonetheless, the downside of rule mining models lies in not having available the latent knowledge in a way that KGEs do, as seen by their usually weaker performance.

The two state-of-the-art models for the respective approaches are ExpressivE from Pavlović and Sallinger and AnyBURL from Meilicke et al. The first is a recently developed knowledge graph embedding model, which can capture various inference patterns jointly and provide intuitive interpretations of the captured patterns. The latter, AnyBURL, is a state-of-the-art rule mining approach that learns directly on a knowledge graph. Nevertheless, ExpressivE performed better on the datasets for which both algorithms were tested.

This thesis combines the advantages of both knowledge graph completion approaches using the state-of-the-art KGE model ExpressivE and compares results to those of the state-of-the-art rule miner AnyBURL. Furthermore, it will be investigated if combining both approaches yields better results by combining AnyBURL rules mined from the knowledge graph itself and ExpressivE's rules from the latent knowledge in the model.

The model, released by Pavlović and Sallinger, offers graphical interpretations of patterns through hyper-parallelograms and their spatial relations to each other. The resulting geometric interpretation is used to build the proposed algorithm, which will be called ***ExpressivE RM***. ExpressivE RM stands for ExpressivE Rule Mining. Using the

geometric interpretation of the ExpressivE embedding, with its high knowledge-capturing capabilities, we create a rule set, which can further be used for knowledge graph completion (KGC) tasks, offering the ability to explain predictions while at the same time using the promising performance of the base model to get high-quality rules. The goal is to show that mining rules on ExpressivE is possible and to make latent knowledge learned by the knowledge graph embedding explicit by adding another abstraction layer and creating easily understandable and interpretable rules. This overcomes the problem of the non-explainability of predictions in a knowledge graph and the problem of latent knowledge not being available for state-of-the-art rule mining algorithms that mine directly on the dataset. The work is considered a success if it can be shown that it is possible to mine rules on ExpressivE, and the results compete with AnyBURL, the best-performing rule mining algorithm as of now.

1.2 Research Questions

- (RQ1) Can the geometric interpretation of ExpressivE be used to mine rules with prediction performance in knowledge graph completion?
- (RQ2) To what extent can the runtime of ExpressivE RM be improved while maintaining practical applicability?
- (RQ3) In what ways do rule sets generated by ExpressivE RM differ from those produced by state-of-the-art rule miners?
- (RQ4) Can combining rule sets from ExpressivE RM and the state-of-the-art rule miner, boost overall performance in knowledge graph completion?

1.3 Methodological Approach

This thesis uses the following methodological approach:

1. **Literature review**
   The first step is to conduct a literature review. The literature review will focus on knowledge graphs, knowledge graph embeddings and rule mining algorithms. The main target is to identify the best-performing state-of-the-art models and classify these methods.

2. **Development of rule mining algorithm**
   Based on the literature review, the most common techniques will be extracted to mine rules, especially those on knowledge graph embeddings. The main step is the development of an algorithm, which uses a knowledge graph embedding model as a base for mining rules according to the model's parameters.

3. **Implementation**
   The designed algorithm will be implemented. The evaluation of the state-of-the-art model will be adapted to have comparable results.

4. **Evaluation and interpretation**
   The hypothesises for the research questions are constructed. The experiments will be evaluated and interpreted. Lastly, the research questions are answered.

1.4 Structure of the Work

This thesis is structured as follows:

- **Background**
  Chapter 2 introduces essential concepts of knowledge graphs, knowledge graph embeddings, and evaluation metrics. Details such as inference patterns, which are crucial for the proposed method, are described in this chapter. Lastly, the dataset that is used will be described here.

- **Related work**
  Chapter 3 presents current state-of-the-art knowledge graph embedding models and rule miners.

- **Proposed approach**
  Chapter 4 introduces the developed algorithm to mine rules from the knowledge graph embedding model ExpressivE. The standard approach will be presented, and additionally, the extended algorithm, for more rules at the cost of resources and speed.

- **Evaluation**
  Chapter 5 describes the results and compares them to one of the best-performing state-of-the-art rule mining algorithm AnyBurl 2.

- **Conclusion**
  The final Chapter 6 answers the proposed research questions using the results and outcomes of this thesis. Lastly, additional possible improvements and future research directions are discussed.

CHAPTER 2

Background

2.1 Knowledge Graphs

A knowledge graph (KG) is a way of modeling and storing information using entities and relations, which connect entities and, therefore, model a connection between the items. Knowledge graphs have been used for storing different information as examples like Freebase [lai16], and WordNet [Uni] show. They represent data useful for tasks like providing recommendations, question answering, information retrieval and natural language processing [PS23]. A knowledge graph can be represented as a set of triples $r_i(e_h, e_t)$ over relations $r_i \in \mathbf{R}$ and entities $e_h, e_t \in \mathbf{E}$ [PS23].

2.2 Knowledge Graph Embeddings

A knowledge graph embedding, called KGE, is a model in which entities and relationships are modeled in vector spaces. A model tries to capture the semantic relations between entities of a given KG. It enables simple manipulations of data and better interpretable coherences. According to Wang et al. (2017), KGEs quickly gained attention in research after the first approaches. The learned embeddings can be used for various tasks, such as knowledge graph completion, relation extraction, entity classification and entity resolution [WMWG17]. KGEs are, therefore, tools that represent KGs in the best possible structure while simplifying various computations and permutations, which is especially beneficial for large datasets. There are several popular KGE models, such as TransE [BUG+13], RESCAL [NTK+11], DistMult [YYH+14], ComplEx [TWR+16], and many others. Fundamental in this work are the KGE models BoxE [ACLS20] and ExpressivE [PS23].

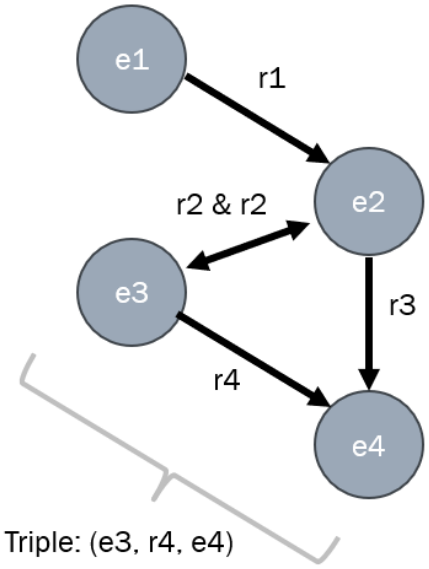

Figure 2.1: Abstract representation of a knowledge graph with 4 entities and 4 relations

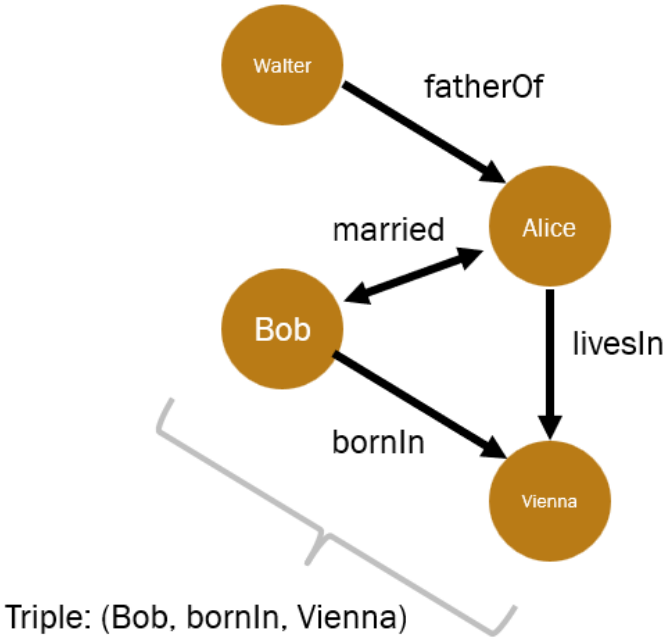

Figure 2.2: An example instance of a (heterogeneous) knowledge graph with 4 entities and 4 relations

2.3 Knowledge Graph Completion

KGs store large amounts of structured data. However, there may still be missing parts or incomplete information. This is where knowledge graph completion (KGC) comes into play. The task of knowledge graph completion aims to fill the missing pieces of information by using machine learning algorithms to predict new information based on the known data in the KG [PS23]. West et al. (2014) point out that, for example, the Freebase dataset lacks the nationality of 75 % of people represented in the dataset [WGM+14]. Using already known information, such as the languages people speak, can help to fill in the missing information. Significant progress has been made towards using KGEs to do this task by embedding entities and relations to quantify the probability of unknown triples before. According to Wang et al. (2017), this technique is up-and-coming, providing good results [WMWG17].

Nevertheless, filling in missing data is not bound to KGEs, yet rule sets can help with this task. For example, AnyBURL and AMIE3 are models that "complete" KGs using rule sets. This thesis aims to find the rules which can be used for prediction tasks on a KGE.

2.4 Inference Patterns

An inference pattern in the context of KGs refers to specific patterns that KGEs learn to make predictions about missing entities or relationships in a KG. More scientifically speaking, a model captures an inference pattern if a set of parameters satisfies the pattern exactly and exclusively [SDNT19]. The patterns are learned from the structure and information of the data stored in the respective KG. They are furthermore used to predict missing information. So, inference patterns are a vital part of KGC tasks. Table 2.1 lists various inference patterns that KGE models can mine, yet not every KGE model can capture all patterns [PS23]. A variety of KGE models can capture some inference patterns. Challenging in this regard is the ability to capture multiple inference patterns simultaneously in a model [ACLS20]. Even though there were many improvements made to early models, no model could overcome all limitations until the release of ExpressivE, which is able to capture all listed inference patterns, with which earlier or similar KGE models were struggling [PS23].

The importance of inference patterns becomes clear when thinking of a dataset containing entities of people being married to each other. Under the open world assumption (OWA; see 2.9), a symmetry pattern may be learned, indicating that an entity A representing a person and having a "married"-relationship to an entity, representing person B, also implies a "married" relationship from B to A. Similarly, a person A having B as a parent implies that B cannot be a parent of A in the real world. This is an example of an anti-symmetric relationship. Figure 2.3 shows many prominent inference patterns and their representation in a KG.

Inference Pattern
Symmetry: $r_1(X,Y) \Rightarrow r_1(Y,X)$
Anti-symmetry: $r_1(X,Y) \Rightarrow \neg r_1(Y,X)$
Inversion: $r_1(X,Y) \Leftrightarrow r_2(Y,X)$
Compositional definition: $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_3(X,Z)$
General composition: $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$
Hierarchy: $r_1(X,Y) \Rightarrow r_2(X,Y)$
Intersection: $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$
Mutual exclusion: $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$

Table 2.1: This Table Lists Patterns That Several KGEs Can Capture (As Seen in Pavlović and Sallinger [PS23])

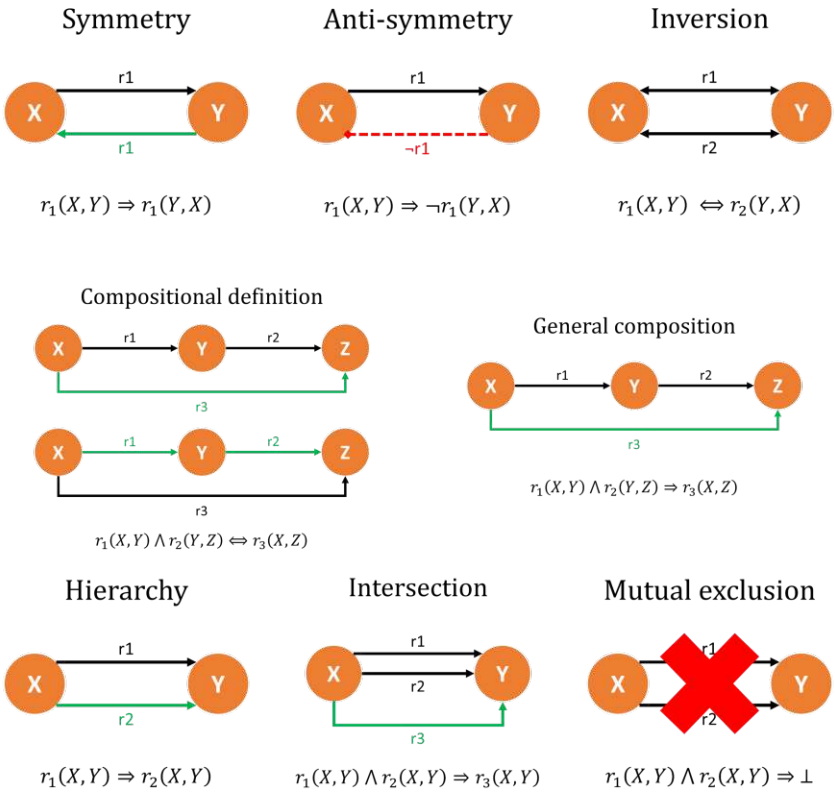

Figure 2.3: The illustration shows many prominent patterns that several KGEs can capture

2.5 Expressiveness

An exciting feature of KGEs is their expressiveness. According to Pavlović and Sallinger (2023) [PS23], "a KGE is fully expressive if, for any finite set of disjoint true and false

triples, a parameter set can be found such that the model classifies the triples of the set correctly." A fully expressive model can represent a whole graph without information loss. An essential difference to this is the inference capability of a KGE. Expressiveness and inference capability do not come in hand with each other [ACLS20]. Fully expressive KGE models can still lack generalisations among data sets [ACLS20]. However, not fully expressive models can lead to significant underfits of training data to a model, according to Abboud et al. (2020). Therefore, full expressiveness and support of essential inference patterns are crucial for high-quality prediction tasks [PS23].

2.6 Confidence of Rules

As summarised by Lajus et al. (2020), confidence in rules can be represented by various metrics such as the following. Let $\vec{B} \Rightarrow r(x, y)$ be a horn rule [LGS20].

2.6.1 Support and Head Coverage

Galárraga et al. (2013) [GTHS13] define **support** "as the number of distinct pairs of subjects and objects in the head of all instantiations that appear in the knowledge base" with z_1, ..., z_m being the variables of the rule apart from x and y:

$$supp(\vec{B} \Rightarrow r(x, y)) := \#(x, y) : \exists z_1, ..., z_m : \vec{B} \wedge r(x, y)$$

A proportional version of support that does not need to know the absolute size of the knowledge base is **head coverage**. It is "the proportion of pairs from the head relation that are covered by the predictions of the rule":

$$hc(\vec{B} \Rightarrow r(x, y)) := \frac{supp(\vec{B} \Rightarrow r(x, y))}{\#(x', y') : r(x', y')}$$

2.6.2 PCA Confidence

The partial completeness assumption (PCA) is the assumption, that if *r(x,y)* is true in the knowledge base for some *x, y*, then we know all r-attributes of x, which is represented as follows [GTHS13]:

$$\forall y' : r(x, y') \in KBtrue \cup NEWtrue \Rightarrow r(x, y') \in KBtrue$$

Under this assumption, the confidence is not normalised by the whole set of facts but only by the facts "of which we know that they are true, together with the facts of which we assume that they are false" [GTHS13]. The PCA confidence is defined as follows, with

KBtrue being known true facts and NEWtrue being new facts that are yet unknown to the knowledge base:

$$pcaconf(\vec{B} \Rightarrow r(x,y)) := \frac{supp(\vec{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, ..., z_m, y' : \vec{B} \wedge r(x,y')}$$

2.6.3 CWA Confidence

The closed world assumption confidence takes all facts that are non-existent in the knowledge base as negative evidence, which makes it a ratio of its predictions that appear in the knowledge base [GTHS13].

$$conf(\vec{B} \Rightarrow r(x,y)) := \frac{supp(\vec{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, ..., z_m : \vec{B}}$$

2.6.4 GPRO Confidence

The GPRO confidence is a refinement of the PCA confidence since it can "underestimate the likelihood of a prediction in the presence of non-injective mappings" [GTHS13]. The quality of a predicted fact can be rated by calculating the GPRO confidence on the first and second variables of the head atom of the rule. The definition can be found in *Graph Pattern Entity Ranking Model for knowledge graph completion* [EI19].

2.6.5 GRANK Confidence

The GRANK is another refinement. It uses the GPRO metric and takes the number of instances of the variables of a rule into account that are not in the head atom [GTHS13, EI19].

2.7 Predictive Quality

An appropriate form of quantitative metrics will be used for evaluation purposes to allow comparison and interpretation of results. Literature provides many different metrics, from which the following will be used to determine the quality of the proposed algorithm and for a side-to-side comparison to similar approaches.

2.7.1 Hits@k

The metric and its description were taken from Bordes et al. (2013) [BUG+13]. According to the paper, the hits@k metric is a performance metric used in many knowledge base completion and rule mining models. It is a proportional measurement of correct predictions in the first k positions of a ranked list. An evaluation algorithm first generates a ranked list of predictions in ascending order. It stores the number of hits, so correctly

predicted items in the top k positions of the resulting list. The number of correct predictions is then divided by k, resulting in the respective hits@k score. The mean of all hits@k scores will be used to evaluate a set of queries. The metric, therefore, tells the effectiveness of predictions of a specific algorithm or model. The ranks are not present in the metric; they are only the number of correct predictions and total predictions per prediction query.

$$\text{Hits@k for the rule set} = \frac{\text{Count of correct predictions in the top k positions}}{\text{k}}$$

$$\text{Overall Hits@k} = \frac{1}{N}\sum_{i=1}^{N}\text{Hits@k for rule set i}$$

2.7.2 Mean reciprocal rank (MRR)

The metric and its description were taken from Bordes et al. (2013) [BUG+13]. According to the paper, the mean reciprocal rank or MRR is a metric using the result lists of prediction tasks. It takes the position of the first correctly predicted item in the result list. Likewise, to hits@k, the result list is used to calculate this metric, but regardless of the top k results. The first correctly predicted item for this metric will be determined in a result list. The rank is then taken, and its inverse value is the final result. Again, the mean of these results is the evaluation result for the whole prediction task.

$$RR_i = \frac{1}{\text{Position of first correct prediction in rule set i}}$$

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N} RR_i$$

2.7.3 Summary

For the evaluation, the metrics hits@k and MRR are used. These metrics are famous for similar approaches, allowing easy comparison and interpretability. While the hits@k metric describes how well predictions are in total, the MRR metric indicates how well the algorithm ranks correct predictions at the top of the result lists. Both metrics are beneficial for prediction tasks in the context of KGC and rule mining models. The metrics provide valuable insight into the model's performance and where optimizations can be made. Additionally, they allow comparison across various algorithms.

2.8 Datasets

For evaluating the proposed algorithm, a dataset is needed, representing data in the form of a knowledge graph, with facts being connected via relations to each other. A dataset

in a relational database is, therefore, not necessarily sufficient. The literature proposes many datasets. The one used for evaluation in this thesis is described here.

2.8.1 WN18RR

A typical dataset in similar approaches is WN18RR [PS23, ACLS20, SDNT19, MCFS20, RBF+21]. It reaches back to WordNet, an extensive database of the English language. Word types, such as nouns, verbs, adjectives, and adverbs, are categorised and grouped into specific sets, addressing the word's relations to other words and meanings [Uni]. Derived from this set is WN18 [BUG+13], a dataset following a strict hierarchical structure. It consists of 18 relations, 40.943 entities and 151.442 triples. The authors Dettmers et al. (2018) found similar drawbacks that FB15k-237 had before and, therefore, introduced the WN18RR dataset [DMSR18]. The goal of this dataset is not simply to offer the possibility for algorithms to complete missing information by single rules but also to require the whole knowledge graph to be modeled and inspected. The final dataset WN18RR, as found in the literature, consists of 93.003 triples, 40.943 entities and 11 relations. The words of the English language represent the entities. The relation set includes relations such as also_see, interstance_hypernym or verb_group, a relation for two verbs, similar in meaning [ABH19].

Triple
(00082563, _synset_domain_topic_of, 00612160)
(07803545, _hypernym, 07802417)
(00634472, _derivationally_related_form, 05651680)
(06488880, _hypernym, 06481320)
(00972621, _hypernym, 00955060)
(00631391, _also_see, 01878466)
(02461014, _member_meronym, 02461128)

Table 2.2: Some Triples as Seen in the WN18RR Training Set

2.8.2 Summary

This work uses WN18RR as a dataset for evaluation. We use it due to its correction, which prevents inverse relation test leakage, and its wide usage across similar approaches in the literature. This makes the here proposed algorithm comparable to other approaches, especially AnyBURL.

2.9 Open World Assumption/Closed World Assumption

According to Galarraga et al. (2013) [GTHS13], the open world assumption (OWA) and closed world assumption (CWA) are important assumptions which have to be considered when designing KGC algorithms. While the open world assumption states that every

entity, relation, or triple is known knowledge and missing information is just unknown, meaning it can be true or false, the closed world assumption states that the information in a knowledge graph is complete. This further implies that every triple which does not exist is false. This results in KGs under the OWA assumption storing more data by nature since negative examples are stored implicitly in KGs under the CWA by just not providing a particular triple or fact.

CHAPTER 3

Related Work

This chapter overviews current state-of-the-art models and approaches. The strengths and weaknesses of each model and approach will be determined to take them into account for the next steps of this thesis.

3.1 Early Approaches

As predecessors and relevant approaches upon which current state-of-the-art knowledge graph embedding models such as BoxE and ExpressivE build, models such as TransE, DistMult and ComplEx can be named. These models are classified into Functional, Bilinear, and Spatial (or Region-based) models [ACLS20, PS23]. TransE is the pioneering functional model which influenced many other extensions, such as RotatE [SDNT19], MuRP [BAH19a], RotH [CWJ+20], HAKE [ZCZW20], ConE [BYRL21], which embed entities as points and relations as translations in a high-dimensional vector space. Improved models such as RotatE emerged later on, trying to solve the limitations of TransE, such as the issue that the model cannot capture symmetric relations.

Bilinear Models, on the other hand, embed knowledge graphs as bilinear products between the entities and the relations. These models can be fully expressive, yet they cannot capture all inference patterns proposed in this work. Models in this family are for example RESCAL [NTK+11], DistMult [YYH+14], ComplEx [TWR+16], SimplE [KP18], and TuckER [BAH19b] according to Pavlović and Sallinger (2024) [PS24].

Spatial or region-based models try to embed relations and entities in a graphical embedding space, with certain representations indicating specific inference patterns [ACLS20, PS23]. This can be done with hyper-rectangles (boxes) that embed entity classes and show their hierarchies through geometric subsumptions of the respective boxes, such as many authors already showed [SC18, VLMM18, LVZ+19].

Inspired by these predecessors, the state-of-the-art model ExpressivE [PS23] emerged, having the just mentioned models as closely related approaches and using the advantages of spatial and functional models to capture many inference patterns with the same model. Pavlović and Sallinger (2024) [PS24] call it spatio-functional models.

3.1.1 BoxE

The 2020 released spacial model BoxE [ACLS20] by Abboud et al. uses a similar approach in this work. BoxE embeds facts as triples, aiming to inject facts into the embedding model. The model represents each entity using two vectors, where one represents the base position of an entity and the other represents the so-called translational bump, which translates all entities co-occurring in a fact with the entity from the base position to the final embedding. Furthermore, every relation is embedded in hyper-rectangles. The boxes define regions, and a fact holds when the final embeddings of certain entities appear in the corresponding box. BoxE uses a scoring function supported by a distance function that evaluates entity positions relative to the box positions. Like ExpressivE, this offers easy interpretability by looking at the box configurations. Another common factor with ExpressivE is the ability to be fully expressive. However, the model is not capable of capturing composition patterns. Also, the way of embedding relations and entities differs in BoxE and ExpressivE. The latter is influenced by spatial models, such as BoxE and functional models, combining the advantages of both worlds.

3.2 State-of-the-Art Rule Mining Approaches

As seen in Lajus et al. (2020) [LGS20], Inductive Logic Programming (ILP) is the task of learning rules from positive and negative samples. First-generation rules mining systems were developed before the time of large knowledge bases. However, these systems do not scale well to millions of facts and do not account for the Open World Assumption made by current knowledge bases [LGS20]. For example, an approach called FOIL [Qui90] cannot be used directly on a knowledge base since it needs explicit counter-examples for rules provided by the user. On the other hand, WARMR [Han02] assumes the knowledge base is complete by building on a closed world assumption (CWA), which generates negative evidence. Based on this, there are similar approaches, such as creating negative evidence from random facts. This strategy has been observed to work not as well on knowledge bases as the partial completeness assumption (PCA), which was designed for knowledge bases.

Second-generation rule mining systems explicitly target large knowledge bases and have proven to be more efficient and faster than first-generation systems. However, the process of rule mining can still take an extended amount of time, depending on the size of the knowledge base. Approximation and parallelisation strategies tackle these problems in recent approaches [LGS20]. RudiK [OMP18] uses the partial completeness assumption (PCA) to generate semantically related counter-examples for data. Using a heuristic approach, the strategies can be described as finding all the rules necessary to predict

positive examples. The approach is non-exhaustive since it focuses on rules that make good predictions and not all available rules above a certain threshold. However, AMIE 3 promises to outperform RudiK in runtime while being an exhaustive rule mining algorithm.

3.2.1 AMIE 3

The 2013 proposed Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases approach (short: AMIE) by Galarraga et al. [GTHS13] focuses on generating rules by relying on the Open World Assumption trying to find true facts unknown to the knowledge base (called NEWtrue) and false facts not known by the knowledge base (called NEWfalse). AMIE's challenge lies in predicting unknown facts while dealing with the fact that semantic knowledge bases do not contain negative evidence. The algorithm uses metrics such as support, which quantifies the number of correct predictions, defined as the number of distinct pairs of subjects and objects in the head of all instantiations appearing in the knowledge base. The second metric used is head coverage, a proportion version of support. AMIE generates rules by iteratively extending its rules using mining operators, namely "Add Dangling Atom" (OD), "Add Instantiated Atom" (OI) and "Add Closing Atom" (OC). Rules are then pruned by discarding rules that have a head coverage below a certain threshold or if they do not have higher confidence than shorter rules.

AMIE was twice improved by its successors AMIE+ [GTHS15] and AMIE 3 [LGS20]. The latter introduces optimizations and pruning techniques, enabling the algorithm to work on large Knowledge Bases. AMIEs strengths lie in the ability to mine rules without parameter adjustment and without any other input data than the knowledge graph.

3.2.2 AnyBURL

According to Meilicke et al. (2019) [MCRS19], creating triples for incomplete knowledge graphs can be done with the additional help of resources such as information from web pages or databases, but also by using the triples in a given knowledge graph and inferring new triples from it. The latter approach relies on statistics, patterns, distribution, or other patterns in the knowledge graph's data. For a long time, approaches tried to learn symbolic representations, for example, in logic programming [MDR94] and relational association rule mining [DT01]. Current approaches use low-dimensional, sub-symbolic representations of knowledge graphs. Examples such as RESCAL [NTK+11] and TransE [JGO22] lay the foundation for many similar models developed recently. This underrepresents symbolic approaches. However, AnyBURL promises rule mining using specific language bias and generalisation of sampled paths into rules.

Meilicke et al. (2019) [MCRS19] propose an anytime bottom-up technique for learning logical rules from knowledge graphs. They generate rules by sampling paths from a given knowledge graph, creating ground path rules (bottom rules), which can then be generalised and used for knowledge graph completion. The AnyBURL algorithm was further improved

in 2020 [MCFS20] by adding the so-called Object Identity, initially proposed by Semeraro et al. (1994) [SEM+94] and adapted to the approach and reinforcement learning as optimization for rule generation. The first is done by adding additional inequality constraints to each rule, and the latter is a technique to speed up the path sampling problem.

AnyBURL generates rules by sampling paths from the knowledge graph and then creating a bottom rule for each path. Starting from every bottom rule, three constraints are used to generate more general rules, making it a bottom-up rule learning approach. Each rule gets a confidence score assigned to choose the best rule (meaning the one with the highest confidence) in the knowledge graph completion task. The confidence, used by the authors of AnyBURL, is based on the support metric (see 2.6.1) of the rule set, which describes the explanatory quality by reconstructing triples of a given knowledge graph using the rules of the set. This metric is also used in the reinforcement optimization.

AnyBURLs strengths lie in the predictive quality of rules, fast computation time with low resources and explainability, and the ability to work on large knowledge graphs, making the approach the most interesting to compare with in the evaluation of this thesis.

3.3 State-of-the-Art Knowledge Graph Embedding Models

3.3.1 ExpressivE

Pavlović and Sallinger (2023) presented a spatio-functional embedding model for knowledge graph completion called ExpressivE [PS23] referencing the model's expressiveness. The approach overcomes challenges such as fully capturing vital inference patterns, capturing prominent patterns jointly and offering an intuitive interpretation of the pattern captured. ExpressivE embeds pairs of entities in a coordinate system and relations as hyper-parallelograms in the so-called "virtual triple space". This technique allows a graphical interpretation of entities and relations and their relationships. The model can compete with state-of-the-art models and outperform them on specific datasets.

ExpressivE models are generated using a scoring function based on distances, which uses the distance between an embedding of a triple and a hyper-parallelogram representing a relation. Triples have to satisfy certain inequalities to be considered true in ExpressivE. The properties of a knowledge graph are visualised in the virtual triple space, offering a geometric interpretation of embeddings, making it the perfect candidate model for applying rule mining on the model. Rules mined can then be used for the knowledge graph completion task, offering the advantage of explainability and potentially high-quality rules.

The strengths of ExpressivE lie in the ability to capture various patterns and offer full expressiveness, meaning that an ExpressivE model can represent any given graph in its geometric interpretation.

3.4 Rule Mining on Knowledge Graph Embedding Models

3.4.1 A Rule Mining Approach on Knowledge Graph Embeddings

Josang et al. (2022) [JGO22] evaluated the effectiveness of knowledge graph completion using rule mining. In detail, they used different knowledge graph embedding approaches, such as TransE, DistMult and ComplEx, together with three different selection methods: least frequent, most frequent and random. The researchers used the above-mentioned AMIE3 algorithm to generate rules on the incomplete knowledge graph using PCA confidence (see 2.6) as a scoring function. Then, the authors used the respective KGE models to predict triples with high confidence. These rules were then compared to the rules of the completed knowledge graph using the KGE approaches. One extension of each base dataset was created for each combination of KGE and entity selection method. Tested on the datasets WN18RR and the family KG, the results showed that using this technique, huge differences were found between extracted rules depending on the KGC model used. However, the work demonstrates the potential of using rule mining and knowledge graph embedding models. It highlights the significant difference in the number and quality of rules depending on the selected base model. Nevertheless, the experiments were run on relatively small datasets and need to be examined in future by larger datasets, reducing the number of candidates generated.

3.5 Summary

This section shows on which basis state-of-the-art models are built. With continuous improvements and a variation of embedding techniques, KGE models are promising for knowledge graph completion. However, although some of these models offer a graphical interpretation of facts, they still easily lack explainability. Nevertheless, having rules can overcome this burden of "black box models" which are fed with data and return specific new predictions. Rule mining models can use a bottom-up approach, such as AnyBURL or a top-down approach, such as AMIE3 and its predecessors.

CHAPTER 4

Proposed Method

4.1 Challenge

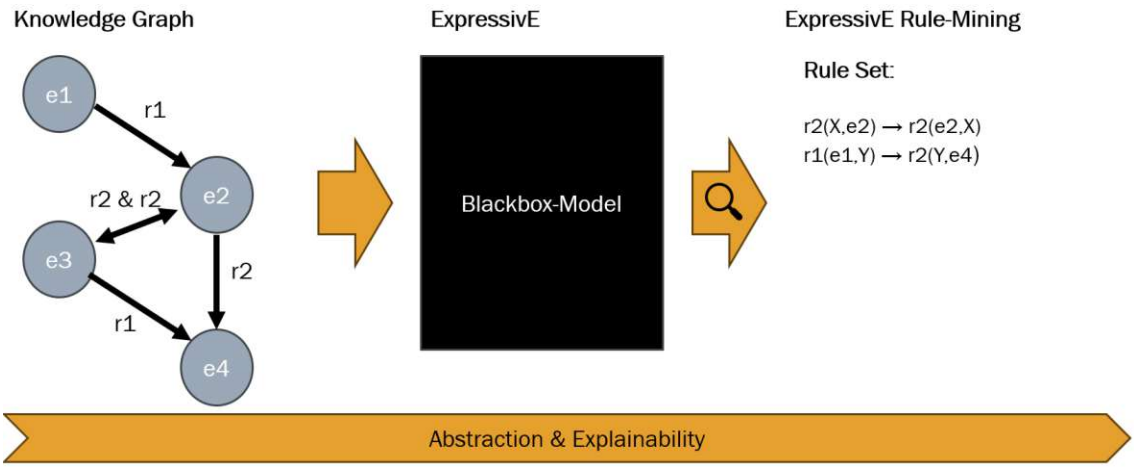

Figure 4.1: Illustration of the proposed method to improve explainability by another abstraction layer

As illustrated in Figure 4.1, the ExpressivE Model implicitly saves data by using abstractions as geometric models in multiple dimensions. This abstraction layer allows the use of implicit data of the base knowledge graph to predict relations between entities. However, it is possible to see the steps of the algorithm, but since data is embedded abstractly, it is hardly interpretable. We aim to enhance explainability by adding another abstraction layer that uses the geometric interpretations of the embedding model. This will create clear rules to explicitly represent the information and knowledge that the model has learned implicitly.

4.2 Proposed Method

We chose the KGE model ExpressivE [PS23] for our methodology as it is a state-of-the-art KGE model with many favorable properties:

First, the ExpressivE model is capable of capturing inference patterns, such as symmetry, anti-symmetry, inversion, compositional definition, general composition, hierarchy, intersection and mutual exclusion simultaneously. ExpressivE was the first model with these capabilities.

Second, ExpressivE can capture prominent patterns jointly, such as hierarchy and composition.

Third, ExpressivE achieves this while being fully expressive, so it can find a parameter set that the model can classify all triples of the given set correctly to embed any training data set.

The fourth point leading to the proposed model as a base model is that ExpressivE is competitive with other state-of-the-art KGE models and outperforms them significantly on specific datasets.

The model, released by Pavlović and Sallinger offers geometric interpretations of patterns through hyper-parallelograms and their spatial relations to each other. The resulting geometric interpretation is used to build the here proposed algorithm and method, which will be further called ***ExpressivE RM***. ExpressivE RM stands for ExpressivE Rule Mining. Using the geometric interpretation of the ExpressivE embedding, having the discussed high knowledge capturing capabilities, is used to create a rule set, which can further be used for knowledge graph completion (KGC), offering the ability to explain predictions while at the same time using the promising performance of the base model to get high-quality rules.

The proposed algorithm ExpressivE RM will be evaluated on the WN18RR [DMSR18] dataset. ExpressivE will use d=32 dimensions as seen in SpeedE [PS24], since it was shown that ExpressivE can reach comparable results with fewer dimensions.

The resulting rule set will be compared to the embedding model of ExpressivE used as a base for prediction quality. Furthermore, the results will be compared to the well-performing rule mining model AnyBURL, particularly to prediction quality and performance using the metrics seen in Chapter 2.7. The resulting rule sets will be compared in detail to provide a deeper understanding of the similarities and differences of the algorithms. This is done by comparing the sets rule by rule and grouping rules by the inference patterns they cover. The rules that both approaches have in common will be identified, and those that they do not have in common will be analyzed to find differences and ways to improve the algorithm further. The comparison gives unique insights because AnyBURL is not based on a KGE model but learns rules from a KG directly, as discussed in Section 3.

Inference Pattern	ExpressivE	BoxE	RotatE	TransE	DistMult	ComplEx
Symmetry: $r_1(X,Y) \Rightarrow r_1(Y,X)$	✓	✓	✓	✗	✓	✓
Anti-symmetry: $r_1(X,Y) \Rightarrow \neg r_1(Y,X)$	✓	✓	✓	✓	✗	✓
Inversion: $r_1(X,Y) \Leftrightarrow r_2(Y,X)$	✓	✓	✓	✓	✗	✓
Comp. def.: $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_3(X,Z)$	✓	✗	✓	✓	✗	✗
Gen. comp.: $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$	✓	✗	✗	✗	✗	✗
Hierarchy: $r_1(X,Y) \Rightarrow r_2(X,Y)$	✓	✓	✗	✗	✓	✓
Intersection: $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$	✓	✓	✓	✓	✗	✗
Mutual exclusion: $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$	✓	✓	✓	✓	✓	✓

Table 4.1: This table lists patterns that several KGEs can capture. Specifically, ✓ represents that the pattern is supported and ✗ that it is not supported. Furthermore, "Comp. def." stands for compositional definition and "Gen. comp." for general composition. (As seen in Pavlović and Sallinger [PS23])

4.2.1 Prerequisites

As seen in the Proposition section of ExpressivE [PS23], the following patterns can be captured.

Symmetry Symmetry patterns are represented by parallelograms, which can be perfectly mirrored across the identity line, as seen in the following illustration. The relation represented is considered symmetric if the pattern holds for every dimension. The according proof and the interpretation of non-perfect inference patterns can be found in Proposition F.1 (Symmetry (Exactly)) [PS23]

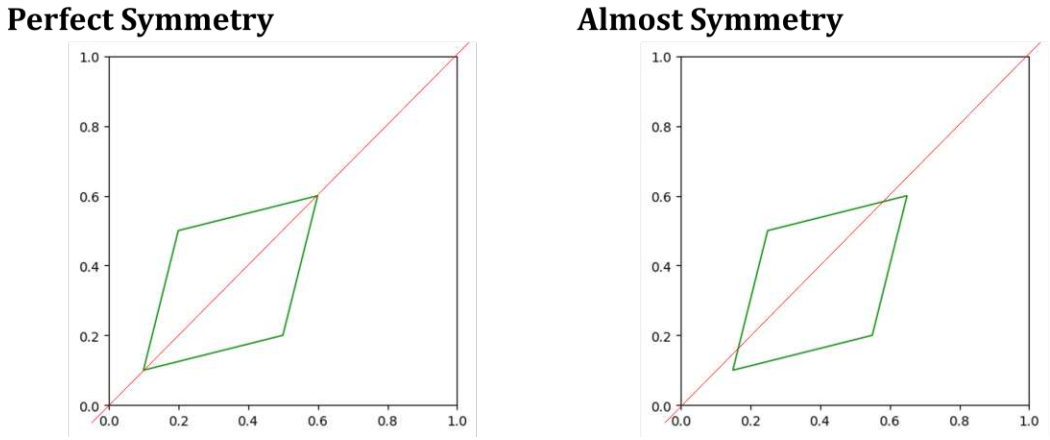

Figure 4.2: An example of a symmetric relation in a single dimension d with scores of 1.0 (left) and 0.53 (right)

Anti-symmetry An anti-symmetric relation is represented by a parallelogram, which is not symmetric across the identity line. A relation is considered anti-symmetric if it is not symmetric in at least one dimension. The according proof and the interpretation of

non-perfect inference patterns can be found in Proposition F.2 (Anti-Symmetry (Exactly)) [PS23]

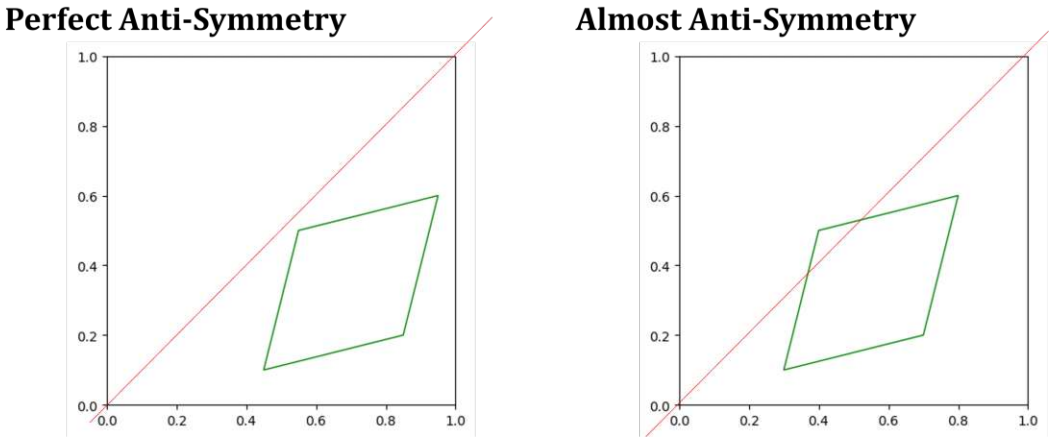

Figure 4.3: An example of an anti-symmetric relation in a single dimension d with scores of 1.0 (left) and 0.47 (right)

Inversion Two inverse relations are represented by parallelograms, which overlap exactly if mirrored across the identity line. For two relations to be inverse, they must fulfill this requirement in every dimension. The according proof and the interpretation of non-perfect inference patterns can be found in Proposition F.3 (Inversion (Exactly)) [PS23]

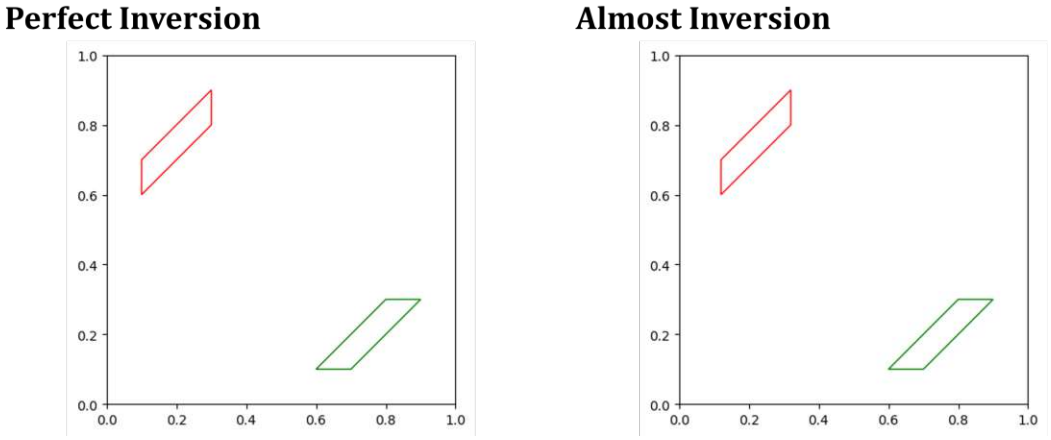

Figure 4.4: An example of two relations that are inverse to each other in a single dimension d with scores 1.0 (left) and 0.72 (right)

Hierarchy A relation is hierarchical with another one if the first one is entirely subsumed by the other one in every dimension, as seen in the following illustration. The according proof and the interpretation of non-perfect inference patterns can be found in Proposition F.4 (Hierarchy (Exactly)) [PS23]

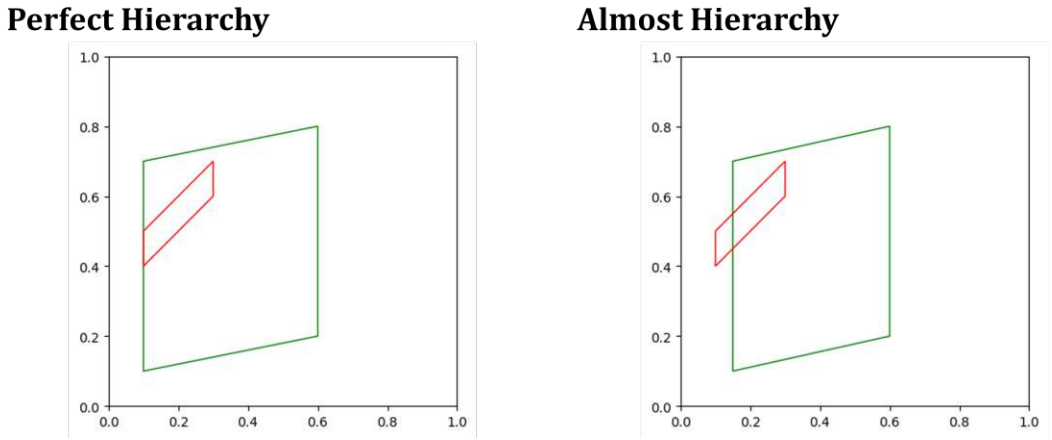

Figure 4.5: An example of hierarchical relationships in a single dimension d with scores 1.0 (left) and 0.75 (right)

Intersection For the intersection pattern, three relationships are involved, namely two relationships creating an overlapping area. The third relation, represented by a parallelogram is considered an intersection if it matches the area of the overlapping area exactly in every dimension. The according proof and the interpretation of non-perfect inference patterns can be found in Proposition F.5 (Intersection (Exactly)) [PS23]

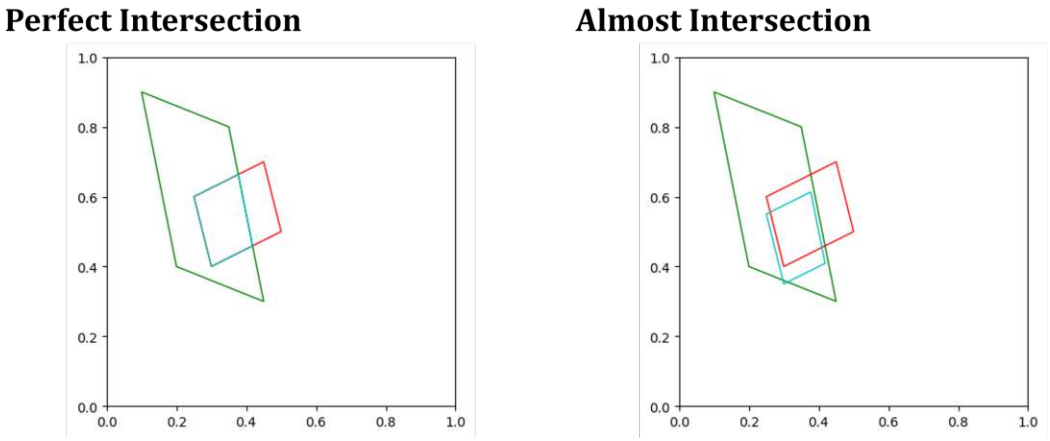

Figure 4.6: An example of the intersection pattern in a single dimension d with scores 1.0 (left) and 0.71 (right)

Mutual Exclusion We consider a mutual exclusion pattern of two relations that are represented by parallelograms and do not have an intersection area in at least one dimension, as illustrated for one dimension as follows. The according proof and the interpretation of non-perfect inference patterns can be found in Proposition F.6 (Mutual Exclusion (Exactly)) [PS23]

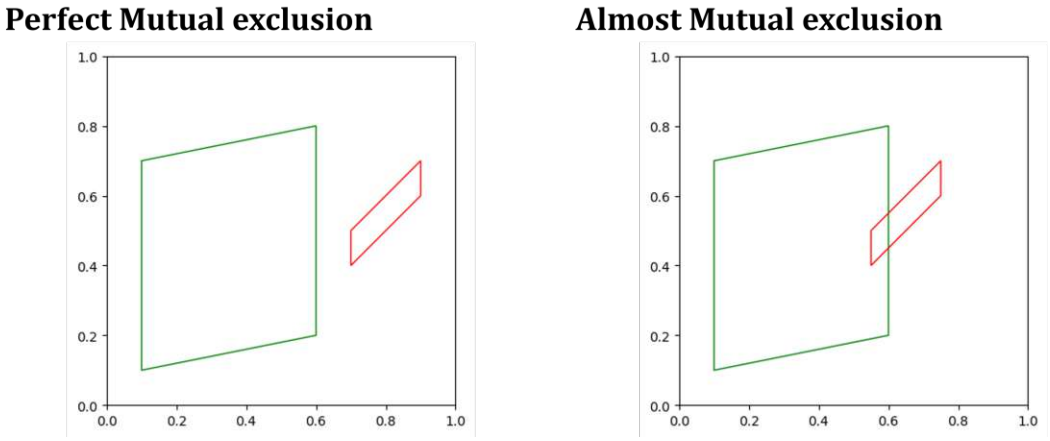

Figure 4.7: An example of mutual exclusion in a single dimension d with scores 1.0 (left) and 0.98 (right)

General Composition The general composition inference pattern consists of three relations, with two relations creating an area entirely subsumed by a third relation in every dimension, as seen in the following illustration. The according proof and the

interpretation of non-perfect inference patterns can be found in Proposition F.7 (General Composition (Exactly)) [PS23]

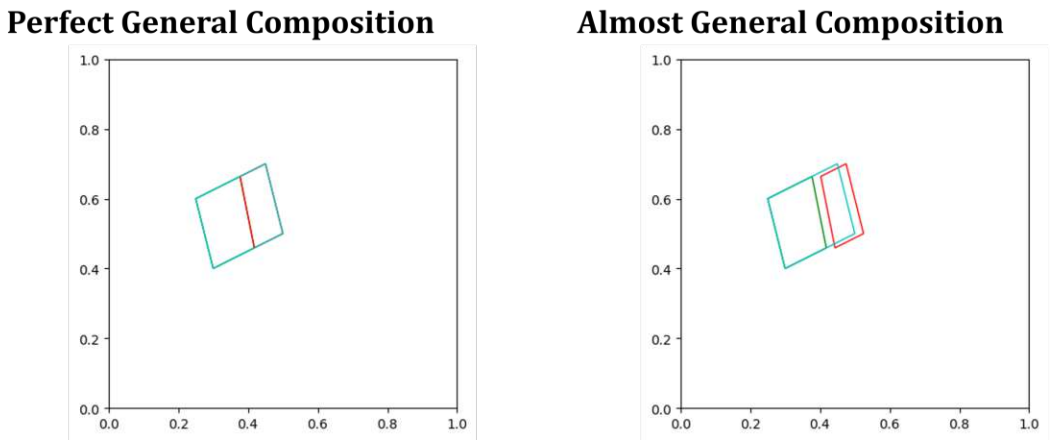

Figure 4.8: An example of the general composition pattern in a single dimension d with scores 1.0 (left) and no assigned score (right)

Compositional Definition The compositional definition is similar to the general composition with three relations involved. However, in this definition, the presence of the third relation also implies the presence of the other two relations, so the implication works in both directions. The according proof and the interpretation of non-perfect inference patterns can be found in Proposition F.8 (Compositional Definition (Exactly)) [PS23]

Introducing scores like these is necessary because the underlying model is not exact but relies on approximations. The exact calculation of the scores for rule candidates can be found in the Pre-processing Section 4.2.3. Both proposed approaches have four phases: Import, Pre-processing, Rule Mining, and Post-Processing. This section shows every phase in detail. The extended approach will additionally run the Rule Mining part using the Largest Rectangle Algorithm.

4.2.2 Import and Configuration

The algorithm is built on top of an ExpressivE model. The first part provides the config file and the checkpoint file, which includes the relations, entities, and the model. The run1. profile file contains the according variables that need to be set: checkpoint_path and config_path. After the embedding model with its entities and relations is ready, the data is ready for pre-processing.

4.2.3 Pre-Processing

In the pre-processing step, following the approach of [PS23], its inverse relation has been added to the set of relations for each relation. Having double the relation count as a result, the algorithm starts by creating a set of combinations between each relation with each other than itself. Additionally, by combining them, they get a confidence score assigned. The score calculation depends on the inference pattern. Some patterns require their existence in at least one dimension; others need to be present in all dimensions. Details can be seen in Section Prerequisites 4.2.1. Depending on the type, the score is calculated as follows:

Score Calculation for Each Inference Pattern in a Single Dimension d

As discussed in Section Prerequisites 4.2.1, the inference patterns have specific properties to count as exact (or perfect) patterns or might still count as patterns but with a reduced certainty because not every captured triple in the certain form represents the inference pattern. This is why certainty scores are calculated based on the patterns that fit in the result model. The scores are calculated differently for each inference pattern. The exact calculation is as follows:

Symmetry To calculate the score of the symmetry inference pattern, the relation r is taken, and the embedding is inspected in each dimension. In each dimension d, the hyper-parallelogram representing r is split at the identity line (the red line in Figure 4.2). One of the cut sides will then be mirrored on the identity line. Then, the intersection of both sides will be determined. The area of the intersection multiplied by 2 is then divided by the sum of the area of both sides.

$$score_d = \frac{2 * \text{Intersection Area}}{\text{Area of one side} + \text{Area of mirrored side}}$$

Anti-Symmetry For the calculation of the score for the anti-symmetry pattern, the relation r is being taken and mirrored on the identity line. Then, one of the sides is taken and inspected. The mirrored part of the side and the non-mirrored part of the original embedding on the same side are taken, and their intersection is calculated. Then the intersection area is divided by both of the pattern areas on the side respectively. The mean of both sides is then taken as a result.

$$score_d = \frac{(\frac{\text{Intersection Area}}{\text{Original embedding on side}} + \frac{\text{Intersection Area}}{\text{Mirrored embedding on side}})}{2}$$

Inversion The inversion pattern needs two relations: *r1* and *r2*. The score is determined by mirroring *r2* on the identity line. Then, the intersection area will be determined. The intersection area is then divided by the sum of the hyper-parallelogram of *r1* and *r2*. The result of the previous operation is multiplied by two.

$$score_d = \frac{2 * \text{Intersection Area after mirroring } r2}{\text{Area of } r1 + \text{Area of mirrored } r2}$$

General Composition For General Composition, no score calculation is available.

Hierarchy For calculating the score of the hierarchy pattern, the embedding of *r1* and *r2* is inspected in each dimension. The intersection area is determined and divided by the area of *r2*.

$$score_d = \frac{\text{Intersection Area}}{\text{Area of } r2}$$

Intersection The score of the intersection inference pattern requires three relations *r1*, *r2* and *r3* and is calculated as follows. First, the intersection of *r1* and *r2* is determined. This pattern is now being taken, and the intersection of it and *r3* is determined. The area of this intersection is then divided by the area of the intersection of *r1* and *r2*.

$$score_d = \frac{\text{Area of intersection of } r1, r2 \text{ and } r3}{\text{Area of intersection of } r1 \text{ and } r2}$$

Mutual Exclusion For calculating the score of mutual exclusion of a relation *r1* and *r2*, the intersection of both is created. Then, the area of the intersection is divided by the area being embedded by *r1*. The result will be subtracted from 1.

$$score_d = 1 - \frac{\text{Intersection Area}}{\text{Area of } r1}$$

Final Score Calculation Across Dimensions

Score calculation for inference patterns that must be present in at least one dimension

$$Score = Max(score_1, score_2, ..., score_n)$$

Score calculation for inference patterns that must be present in each dimension

$$\overline{s} = \frac{1}{n} \sum_{d=1}^{n} score_d$$

$$Score = \frac{\overline{s}}{\frac{1}{n} \sum_{d=1}^{n} (score_d - \overline{s})^2}$$

Vertical-Axis Algorithm

The following algorithm is being run for each dimension. Each relationship pair, with its confidence score assigned, is being taken and filtered by a certain threshold that may be set. For the first run, the threshold may be 0. This will return all potential results but will need a very long computation time, depending on the dataset and hardware used. The intersection areas above the threshold will be taken and cut by several vertical lines in the coordinate system. The corner points of the intersection areas are inspected individually. If an intersection point and its intersection line parallel to the ordinate (*y-axis*) separate the intersection area into two areas, both having an area size greater than 0, and the respective separated areas are added to a new result set. The algorithm will also cut the resulting areas again, if possible until no cutting option fulfills the requirements available anymore. Each pattern in the new result set is the score of the parent relationship pair assigned. Finally, the whole coordinate system and its entities are rotated by 90° and the process is repeated once.

In the Python code, the entities are translated into another data structure for optimization purposes used by the ExpressivE RM extended approach. Additionally, the data is separated in chunks by dimension for parallelism purposes (for both approaches).

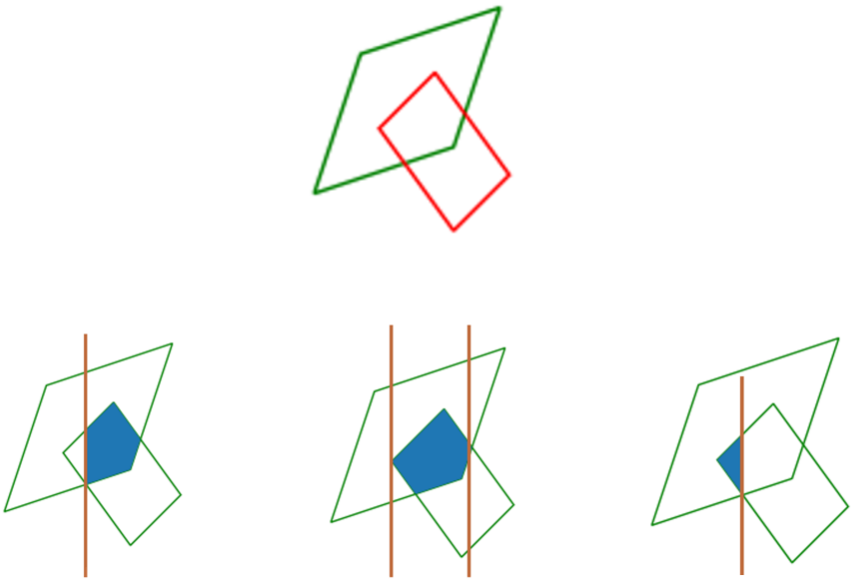

Figure 4.9: An example of the cutting process of two relationships in a single dimension d

4.2.4 Rule Mining

The rule mining process is being run for each dimension in parallel in the same way. However, the standard approach and the extended approach vary in this step.

4.2.5 ExpressivE RM Standard Approach

The standard approach will inspect each pattern of the previous step in the respective dimension in a coordinate system. All the in-lying entity representations are taken for each pattern to create a new rule candidate in dimension d. The rule candidate will look as follows, depending on whether the rotated or un-rotated coordinate system was taken.

For the un-rotated coordinate system

$$\text{Rule Candidate } r_1(e, Y) \rightarrow r_2(e, Y) \text{ for } d$$

For the rotated coordinate system

$$\text{Rule Candidate } r_1(X, e) \rightarrow r_2(X, e) \text{ for } d$$

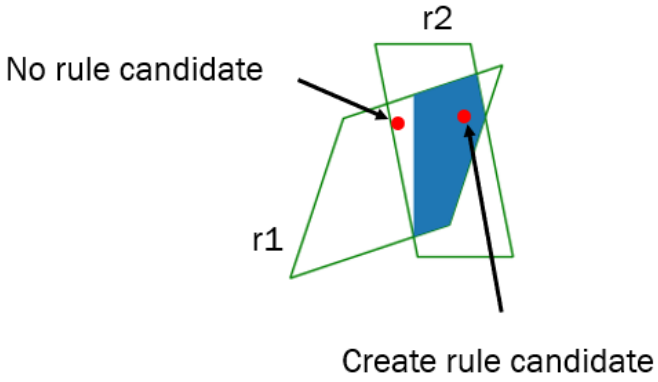

Figure 4.10: Standard rule mining process in a single dimension d with red points as embedded entities

4.2.6 ExpressivE RM Extended Approach (Optional)

The ExpressivE RM extended approach will additionally run the Largest Rectangle Algorithm at this stage.

Largest Rectangle Algorithm

In each result set from the pre-processing, every entity will be taken and used to create a rectangle in the following way: to provide rule candidates with different entities in the head and tail part, so as to extend the search space of rules. To achieve this, the representing point for the entity is being taken, and again, the intersecting vertical line parallel to the ordinate is being taken. This line will cut exactly two points from the current pattern in the loop. These two points will be a newly created rectangle's upper and lower limit. Horizontally, the line will be expanded to get the maximum size while

still being entirely contained by *r2* of the relationship pair. Now, every entity b in the entity set of the model will be checked if it is contained in the newly obtained rectangle. If this is the case, the following Rule Candidate in the respective dimension will be added to the Rule Candidate set.

For the un-rotated coordinate system

$$\text{Rule Candidate } r_1(a, Y) \rightarrow r_2(b, Y) \text{ for } b \in B_a$$

For the rotated coordinate system

$$\text{Rule Candidate } r_1(X, a) \rightarrow r_2(X, b) \text{ for } b \in B_a$$

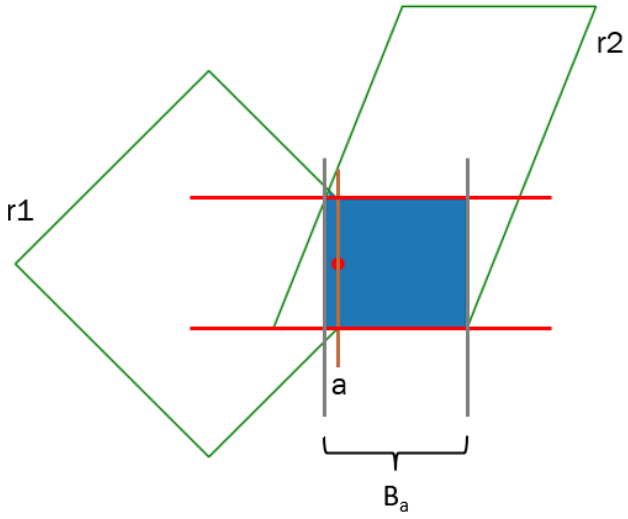

Figure 4.11: Extended rule mining process in a single dimension d with the red point as embedded entity

Note: As the extended algorithm increases the search space notably, the approach is expected to take much longer than the standard approach. However, it promises to allow us to mine rules in a more advanced form.

4.2.7 Post-Processing

The base for the post-processing is the completed rule mining part. The post-processing will be the same regardless of the standard or extended approach. According to Section F in the Appendix of Pavlović and Sallinger (2023) [PS23] and discussed in Section 4.2.1, the final rules being created should be valid in every dimension, or at least one, depending on the inference pattern. However, the model, in this sense, is an approximation of the real data, so in practice, this theory does not fit. Therefore, all rule candidates in each

dimension will be inspected and aggregated across the dimensions. The program allows the setting of a minimal threshold for the number of dimensions a rule candidate must be present to count as final rule. The rule candidates are aggregated as follows, with the lowest of all rule candidate scores being the final score.

d	Rule Candidate	Confidence
1	$r_1(X, 00581891) \rightarrow r_2(00581891, X)$	0.2
2	$r_1(X, 00581891) \rightarrow r_2(00581891, X)$	0.4
3	$r_1(X, 00581891) \rightarrow r_2(00581891, X)$	0.3
1	$r_3(02693319, Y) \rightarrow r_4(Y, 02693319)$	0.6
2	$r_3(02693319, Y) \rightarrow r_4(Y, 02693319)$	0.55
3	$r_3(02693319, Y) \rightarrow r_4(Y, 02693319)$	0.4
4	$r_3(02693319, Y) \rightarrow r_4(Y, 02693319)$	0.8

Table 4.2: An Example of a Rule Candidate Table in the Post-Processing Phase: Number of Dimensions = 4

Final rules	Confidence
$r_1(X, 00581891) \rightarrow r_2(00581891, X)$	0.2
$r_3(02693319, Y) \rightarrow r_4(Y, 02693319)$	0.4

Table 4.3: An Example of a Final Rule Table After the Post-Processing Phase: Number of Dimensions = 4, Confidence Threshold = 3 Dimensions

The final rule set will be exported for knowledge graph completion tasks. The results for applying the rules, mined from the ExpressivE model based on WN18RR, will be presented in the next section, Evaluation 5.

CHAPTER 5

Experimental Evaluation

This section aims to assess the performance and effectiveness of ExpressivE RM, which was introduced in the previous sections. The evaluation process involves training the embedding model, rule mining with ExpressivE RM, and running the state-of-the-art algorithm AnyBURL for comparison. To allow for a fair comparison, the algorithms are executed on the same machine, with an Intel(R) Xeon(R) Silver 4314 CPU @ 2.40 GHz with 64 available cores. The machine has 1008GB RAM and four GeForce RTX 2080 Ti GPUs, which is sufficient to test the algorithms under realistic conditions. To measure the prediction performance, the score metrics introduced in Section 2.7 are used and the values in the objectives Section 5.2 will be used to answer the research questions.

5.1 Experimental Setup

5.1.1 Training ExpressivE

The training process involved optimizing a self-adversarial negative sampling loss using the Adam optimizer, and during this phase, gradient descent was used to fine-tune ExpressivE's parameters [KB14]. The training was stopped early after 1000 epochs if there was no improvement of at least 0.5% in the h@k $k \in \{1, 3, 5, 10\}$ score for WN18RR. For training, the four GPUs were used. For training ExpressivE, we use the parameters of Pavlović and Sallinger (2024) [PS24] that lead to the best-published results for an ExpressivE embedding with dimensionality $d = 32$.

5.1.2 Rule Mining with ExpressivE RM

For rule mining with the proposed algorithm ExpressivE RM, the experiments were run with the following configuration file, which stops at the threshold of a minimum score of 40% and a minimum of 16 dimensions since there were no further improvements in any

scores at this point as observed in a pre-run. For mining the rules, we do not use GPUs, but only CPUs.

5.1.3 Rule Mining with AnyBURL

For evaluating AnyBURL, we used the parameters from the original paper [MCRS19] and a snapshot after 100 seconds with 16 worker threads. AnyBURL was run on the same machine as the other evaluations for a fair comparison setting. AnyBURL uses CPUs as well to mine rules.

5.1.4 Dataset

WN18RR

The WN18RR dataset (see 2.8.1) was used to evaluate the algorithm. The dataset is divided into a train, validation and test set, with 86835 triples, 3034 triples and 3134 triples, respectively. The same train-validation-test-split was used to evaluate AnyBURL, thus it makes results comparable.

5.2 Objectives Derived from the Research Questions

To answer the research questions by quantifiable parameters, the following hypotheses need to be discarded, or the following criteria must be met.

- (RQ1) h0: hits@1, hits@3, hits@10 and MRR equal all 0.
- (RQ2) h0: The runtime of the algorithm lies not within the same time range of either minutes, hours, days, or weeks compared to the state-of-the-art algorithm.
- (RQ3) Comparison by inference patterns and confidence scores.
- (RQ4) h0: Composed combined rule set hits@1, hits@3, hits@10 and MRR are smaller or equal to state-of-the-art algorithms.

5.3 Experimental Details

The parameters and details of the experimental setup and evaluation are stated in this section. Especially the model used and relevant parameters to reproduce the results.

5.3.1 Implementation Details and Reproducibility

Dependencies

The following dependencies must be available in either pip or in a conda environment to execute the code of ExpressivE RM in Python 3.9.

- Matplotlib 3.7.1
- Numpy 1.24.3
- Pandas 1.5.2
- Pykeen 1.7.0
- Python >= 3.9
- Pytorch 1.10.2
- Seaborn 0.13.0
- Shapely 2.0.1
- Tensorboard 2.8.0

For the evaluation, the following additional packages are needed:

- Reportlab

Furthermore, the evaluation needs to have the code of the evaluation of AnyBURL ready. Detailed instructions can be found on the project's homepage: `https://web.informatik.uni-mannheim.de/AnyBURL/`. The folder of the evaluation needs to be set in the profile file of ExpressivE RM. We used Java 20.0.1 2023-04-18 to run AnyBURLs evaluation code and verify the results.

Installation

After installing the dependencies and cloning the code, a profile file needs to be created. The source code provides a sample file which can be copied and adjusted. To use the evaluation step, the respective dependencies must be installed. The working directory should be set to a large drive since it will hold the raw rules and will save much temporary data. With the cleanup parameter, these files can be automatically deleted after usage. The caching option allows the persistence of all possible intermediate results to continue the algorithm from every step in the configuration. Not using caching will result in a high usage of RAM on the system. The result directory will hold the results, such as the confidence matrices, a log of each run, a statistic file (if filtering is enabled) and the generated rules.

Experiments

Base Model The base model ExpressivE is trained by running the *run_experiments.py* Python script. Before that, the according config files must be provided. The source code contains The best configuration files for WN18RR, which can be found in the *Best_Configurations* directory.

According to Pavlović and Sallinger (2023) [PS23] and as done in our experiments, the following parameters must be set:

- **config** contains the path to the model configuration

- **train** contains *true* if the model shall be trained, otherwise *false*.
- **test** contains *true* if the model shall be evaluated on the test set and otherwise *false*
- **expName** contains the name of the experiment
- **gpu** contains the id of the GPU that should be used for training the base model.
- **seeds** contains the seeds for repeated runs of the experiment to ensure reproducibility

AnyBURL (for comparison) For training AnyBURL, we used the settings on the project page. The number of *WORKER_THREADS* was set to 16 for both *config-learn.properties* and *config-apply.properties*.

ExpressivE RM standard approach The algorithm of the implemented approach provides profiles for usage. A profile is a text file separated into different stages, with the possibility of turning single features on and off. Each stage can be optionally executed, while some stages depend on others. The *enabled = true* or *enabled = false* runs or skips the respective stage. The following parameters are available for each stage:

- **Import**
  - **checkpoint_path** path to the checkpoint file of ExpressivE
  - **config_path** path to the config file of ExpressivE
- **Inference_Matrices**
  - Only contains the enabled parameter. This step will create inference pattern matrices and save them to the results directory.
- **Pre-Processing**
  - **preselect_min_score** Selects only patterns with a minimum native score of a value between *0.0* and *1.0*. The native score is not related to the score for each inference pattern but an alternative scoring method.
  - **min_score** Since the scores for each pair of relations are calculated beforehand, the parameter's method can skip patterns before further processing them. If the score of the inference patterns or the native score will be used for the final rules is decided in post-processing (see below).
- **Mine_Rules**
  - **extended_algorithm** Set to *true*, if the ExpressivE RM extended approach should be run additionally, otherwise *false*.

- **Post-Processing**
  - **min_dims** Select only rules that appear in a minimum of the hereby set number of dimensions. The parameter must be an integer between 0 and the total number of dimensions.
  - **min_score** Select only rules that meet a minimum score of the hereby set one.
  - **use_preselect_score** Set to *true*, if the native score should be used for the final rules. Set to *false* if the scores of the inference pattern matrix should be used.
  - **inverse_relations_mode** Must be set to an integer between *0-3*, representing the following options:
    - **0** reject all rules with inverse relations
    - **1** accept all rules with inverse relations
    - **2** exclusively accept rules with inverse relations
    - **3** exclusively accept rules with both relations being inverse
- **Filter**
  - **min_dims** Selects all rules that appear in at least as many dimensions as set here. The value must be between *0* and the total dimensions count.
  - **max_dims** Selects all rules that appear in at most as many dimensions as set here. The value must be between *1* and the total dimensions count. The parameter can be set to *-1* to use the total dimensions count.
  - **min_score** Selects all rules with a minimum score set here. The value must be between *0.0* and *1.0*.
  - **max_score** Selects all rules with a maximum score set here. The value must be between *0.0* and *1.0*.
  - **filter_out_FULL_INVERSE** If set to *true*, all rules where head and body were inverted in the pre-processing will be removed. Otherwise, set to *false*.
  - **filter_out_PARTLY_INVERSE** If set to *true*, all rules where either head or body were inverted in the pre-processing will be removed. Otherwise, set to *false*.
  - **filter_out_NON_INVERSE** If set to *true*, all rules where head and body were not inverted in the pre-processing will be removed. Otherwise, set to *false*.
  - **filter_out_SYMMETRY** If set to *true*, all rules that apply to the symmetry inference pattern will be removed. Otherwise, set to *false*.
  - **filter_out_INVERSION** If set to *true*, all rules that apply to the inversion inference pattern will be removed. Otherwise, set to *false*.

  - **filter_out_HIERARCHY** If set to *true*, all rules that apply to the hierarchy inference pattern will be removed. Otherwise, set to *false*.
  - **filter_out_OTHERS** If set to *true*, all rules that do not apply to the symmetry, inversion or hierarchy inference pattern will be removed. Otherwise, set to *false*.
  - **filter_out_ONE_CONSTANT** If set to *true*, all rules that have exactly one constant variable in head and body together will be removed. Otherwise, set to *false*.
  - **filter_out_TWO_CONSTANTS** all rules that have exactly two constant variables in head and body together will be removed. Otherwise, set to *false*.
  - **filter_out_THREE_CONSTANTS** all rules that have exactly three constant variables in head and body together will be removed. Otherwise, set to *false*.
  - **filter_out_FOUR_CONSTANTS** all rules that have exactly four constant variables in head and body together will be removed. Otherwise, set to *false*.

- **Evaluation**
  - **evaluation_mode** Can be set to either of the following options:
    - **0** uses group-by-group evaluation, which provides a fast, quick overview of results by evaluating each group separately.
    - **1** uses cumulative evaluation, which needs more time, yet provides the probably best results for final evaluation by descending from a perfect score (1.0) and descending from the maximum number of dimensions for each iteration and adding more rules in each step.
  - **threshold_group_size** Splits the rules by threshold with the hereby set group size. A threshold of 1 will create a group for each score *0.00-0.01, 0.01-0.02,....* If set to 2, then the groups will contain rules of the scores *0.00-0.02, 0.02-0.04,....*
  - **dim_cuts** Further divides the groups of rules by the number of dimensions the rules appear in. Provide the number of dimensions for separating the groups and list by using a colon in between. For example, *0, 8, 16, 24* will create four groups with rules appearing in *0-7, 8-15, 16-23, and 24+* dimensions.

- **General** (applies to all stages)
  - **caching** If set to *true*, each data chunk and the intermediate result will be saved to the hard disk instead of using RAM only. Use this to start an experiment from a later stage without always having to run the previous stages. Otherwise, set to *false*.
  - **threads** Set to an integer between *0* and the number of available CPU cores. The algorithm will use a maximum of now specified CPU cores.

- **cleanup** Will delete intermediate results in the working directory after finishing the execution if set to *true*. Otherwise, set to *false*.
- **working_directory_path** Provide a path to an empty directory for saving intermediate results and other temporary files. If the directory does not exist, it will be created.
- **result_directory_path** Provide a path to a directory for saving final results and evaluations. If the directory does not exist, it will be created.
- **evaluation_directory_path** Provide a path to the AnyBURL evaluation directory.

Reproducing the results

We used the following command for training the model:

```
python run_experiments.py gpu=0 train=true test=true seeds=2
config=Best_Configurations/ExpressivE/d32_WN18RR.json
expName=ExpressivE_d32_WN18RR
```

Furthermore, the following commands were used to train and evaluate AnyBURL:

```
java -Xmx12G -cp AnyBURL-23-1.jar de.unima.ki.anyburl.Learn
config-learn.properties
java -Xmx12G -cp AnyBURL-23-1.jar de.unima.ki.anyburl.Apply
config-apply.properties
java -Xmx12G -cp AnyBURL-23-1.jar de.unima.ki.anyburl.Eval
config-eval.properties
```

For running ExpressivE RM we used the following profile-file, named *final.profile*:

```
[Import]
enabled = true
checkpoint_path = /home/jvecera/ExpressivE/checkpoint.pt
config_path = /home/jvecera/ExpressivE/d32_WN18RR.json

[Inference_Matrices]
enabled = true

[Pre-processing]
enabled = true
preselect_min_score = 0.4
min_score = 0.0
```

```
[Mine_Rules]
enabled = true
extended_algorithm = false

[Post-processing]
enabled = true
min_dims = 0
min_score = 0.0
use_preselect_score = true
inverse_relations_mode = 1

[Filter]
enabled = true
min_dims = 0
max_dims = -1
min_score = 0.0
max_score = 1.0

filter_out_FULL_INVERSE = false
filter_out_PARTLY_INVERSE = false
filter_out_NON_INVERSE = false

filter_out_SYMMETRY = false
filter_out_INVERSION = false
filter_out_HIERARCHY = false
filter_out_OTHERS = false

filter_out_ONE_CONSTANT = false
filter_out_TWO_CONSTANTS = false
filter_out_THREE_CONSTANTS = false
filter_out_FOUR_CONSTANTS = false

[Evaluation]
enabled = trueevaluation)
evaluation_mode = 1
threshold_group_size = 1
dim_cuts = 0, 8, 16, 24

[General]
caching = true
threads = 16
cleanup = false
```

```
working_directory_path = /home/jvecera/workdir_final
result_directory_path = /home/jvecera/resultdir_final

evaluation_directory_path = /home/jvecera/evaluation
```

The algorithm was started using the following command:

```
python3.9 ExpressivE_RM.py final.profile
```

5.4 Results

The executions of the algorithms resulted in the following output. The ExpressivE standard approach successfully ran through its procedure with a runtime of less than 15 minutes. This result was made possible by many optimizations. However, the little-optimized extended approach, although theoretically promising, timed out after one week. We will optimize this approach in the future to leverage its theoretical benefits.

5.4.1 ExpressivE RM Standard Approach

Table 5.1: Results for the ExpressivE RM Standard Approach

Benchmark	Score (min)	Dimensions (min)	Rules	h@1	h@3	h@10	MRR
WN18RR	40	0	81118	.3226	.3226	.3226	.3226

Statistics				
	min	**max**	**avg**	**std**
Score	0.40948	0.40948	0.40948	0.0
Constants	2	2	2	0.0
Dims	4	25	18.2244	3.17611

Table 5.2: Statistics

The ExpressivE RM standard approach had its best results at a cutoff confidence score of 40 (see 4.2.3). Reducing the minimum threshold did not result in any better results on the used dataset WN18RR, but only in more rules without effect. The final result is as follows: ExpressivE RM creates a rule set with 81118 symmetry rules in its standard version. The hits@1, hits@3, hits@10 and MRR are respectively 0.3226, 0.3226, 0.3226 and 0.3226. Although it was expected that rules must be present in all dimensions, the results show that this is not the case. Each rule of the result rule set was inspected regarding in how many dimensions it is satisfied. The statistics table shows that there are rules that are satisfied in 4 dimensions. However, the largest number of dimensions in which a rule is satisfied is 25 and not 32. These results are likely due to the base model's approximation nature, which we shall investigate in future work. This knowledge on the

approximation nature of ExpressivE may be used to implement further optimizations, that reduce the runtime even more. Further inspection is necessary regarding the scores that are equal for all rules in the resulting rule set. The result rule set has only rules of a particular type, which needs to be analyzed in detail per rule. The following table shows how many rule candidates were created per dimension.

RULES PRESENT IN DIMENSIONS							
DIM	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	1758 2.17%	72460 89.33%	54180 66.79%	65214 80.39%	57514 70.9%	55362 68.25%	0 0%
DIM	**7**	**8**	**9**	**10**	**11**	**12**	**13**
	77858 95.98%	0 0%	0 0%	77736 95.83%	30196 37.22%	78074 96.25%	0 0%
DIM	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	76414 94.2%	37982 46.82%	74118 91.37%	0 0%	61776 76.16%	77000 94.92%	72316 89.15%
DIM	**21**	**22**	**23**	**24**	**25**	**26**	**27**
	64142 79.07%	32524 40.09%	0 0%	72730 89.66%	50786 62.61%	0 0%	57824 71.28%
DIM	**28**	**29**	**30**	**31**	**32**		
	48060 59.25%	52554 64.79%	78208 96.41%	51546 63.54%	0 0%		

Table 5.3: Rules Present in Different Dimensions

The first integer shows the number of rule candidates in the respective dimension, and below is the ratio of how big the set of rule candidates from the respective dimension is, compared to the final rule set. The table shows that some dimensions do not contain any rule candidates.

Table 5.4 shows the time taken for each processing step. As the timeline shows, rule mining and post-processing account for more than 80% of the total runtime. These were also the steps with the most optimizations and heavily depended on the hardware used and the capabilities of storing larger amounts of data. In total, the execution took 13m and 19s according to the log file output, making ExpressivE RM a very fast algorithm and allowing the quick creation of predictive rule sets.

Task	Time
Importing base model	58s
Creating inference matrices (scores)	14s
Pre-Processing	30s
Rule Mining	7m 31s
Post-Processing	3m 41s
Filtering	2s
Evaluating	18s

Table 5.4: Time Taken for Different Tasks

5.4.2 ExpressivE RM Extended Approach

Although theoretically promising, the extended approach was not optimized to meet the time constraints. The not-optimized extended approach reached the time out after one week. In the future, we will optimize the approach to provide results in a feasible time.

5.5 Comparison

5.5.1 Overview

Table 5.5: KGC performance under low dimensionalities (d=32) as seen in Pavlović and Sallingers work [PS24] compared to Rule Miners ExpressivE RM and AnyBURL

Family	Model	WN18RR			
		H@1	H@3	H@10	MRR
Rule miner	ExpressivE RM standard	.3226	.3226	.3226	.3226
	ExpressivE RM extended	-	-	-	-
	AnyBurl	.449	.499	.557	.484
Knowledge Graph Embedding	ExpressivE	**.485**	**.442**	**.499**	**.571**
	TuckER	.428	.401	-	.474
	MuRE	.458	.421	.471	.525
	RefE	.455	.419	.470	.521
	RotE	.463	.426	.477	.529
	AttE	.456	.419	.471	.526
	HAKE	.416	.389	.427	.467
	RotatE	.387	.330	.417	.491
	ComplEx-N3	.420	.390	.420	.460
	MuRP	.465	.420	.484	.544
	RefH	.447	.408	.464	.518
	RotH	**.472**	.428	**.490**	**.553**
	AttH	.466	.419	.484	.551
	ConE	.471	**.436**	.486	.537

5.5.2 ExpressivE RM Standard Approach and AnyBURL

ExpressivE RM standard approach mines symmetry rules based on the latent knowledge of the underlying embedding model. The state-of-the-art rule miner AnyBURL, however, also mines rules of other types, which raises the question of whether a combination of ExpressivE RMs and AnyBURLs rules provides a benefit.

Firstly, the rule sets of both approaches will be analysed for further combination strategies.

Table 5.6: Number of Mined Rules per Inference Pattern

Pattern	**WN18RR**	
	ERM Standard	AnyBurl
Symmetry	81118 (100%)	14405 (22%)
Inversion	0	37 (<1%)
Hierarchy	0	29 (<1%)
Other	0	50743 (>77%)
Overall	81118	65214

As Table 5.6 shows, the ExpressivE standard approach mines 81118 symmetry rules, whereas AnyBURL generates 14405 symmetry rules. The remaining 78% of rules of AnyBURL consist of other rule types.

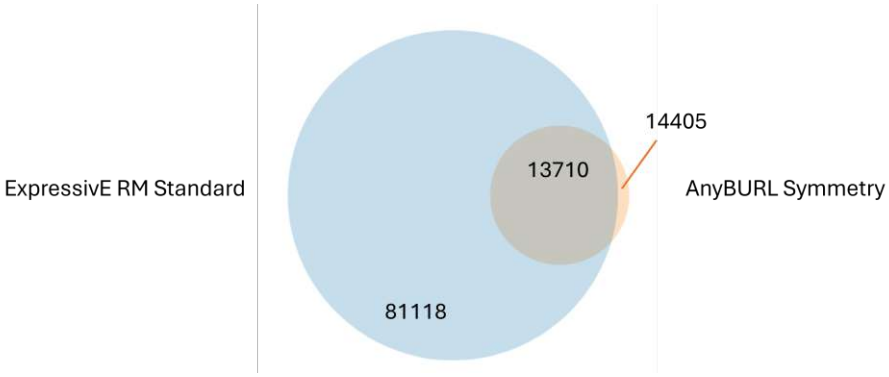

Figure 5.1: Result rule sets of ExpressivE RM standard approach, symmetry rules of AnyBURL and rules both have in common

When analysing the rule sets in detail, it is possible to find 13710 rules both the rule set of ExpressivE RM standard approach and the symmetry rule set of AnyBURL have in common, which accounts for 17% of ExpressivE RMs rules and 95% of AnyBURLs symmetry rules rule set. This means that ExpressivE RM standard covers 95% of the rules of AnyBURL and AnyBURL only 17% of our rules. This raises the question of whether our approach mines a more significant amount of predictive rules or whether

AnyBURL mines higher quality rules. Table 5.7 shows the results for evaluating the exclusively mined rules for both approaches and the performance of the rules both approaches have in common.

Table 5.7: Results for specific rule sets

Rule Set	Rules	h@1	h@3	h@10	MRR
Exclusive ExpressivE RM standard rules	67408	.2773	.2773	.2773	.2773
Exclusive AnyBURL symmetry rules	695	.3465	.3465	.3465	.3465
AnyBURL symmetry rules	14405	.3465	.3465	.3465	.3465
Common rules	13710	.2049	.2049	.2049	.2049

The numbers show that the rules that both approaches have in common reach a hits@1, hits@3, hits@10 and MRR score of 0.2049, which is remarkably lower than the results of both approaches individually. When we exclude these common rules from both rule sets, the remaining 695 rules of AnyBURL have scores respectively of 0.3465, whereas ExpressivE RM standards 67408 rules reach a score of 0.2773 for the used metrics. This indicates that both approaches have exclusive rules with scores in the used hits@k metrics in knowledge graph completion. It might be beneficial to inspect these rules and see how they can be combined with the common rules.

5.6 Combination

The results of the previous sections raise the question of whether a combination of rule sets might be beneficial. This section will use two combinations of rule sets, namely all rules of ExpressivE RM standard and AnyBURLs rules, and in the second combination, ExpressivE RM standards symmetry rules and all other rules from AnyBURL, to create a rule set where AnyBURLs symmetry rules are replaced by ExpressivE RMs.

Table 5.8: Evaluation of combined rule sets of ExpressivE RM standard approach and AnyBURLs rule set.

Rule set	**Metrics**				
	Rules	h@1	h@3	h@10	MRR
All (ExpressivE RM)	81118	.3226	.3226	.3226	.3226
All (AnyBURL)	65214	.4488	.4990	.5570	.4840
Overall	146332	.4488	.4990	.5570	.4840

Putting both rule sets together creates a result set of a total 146332 rules. These rules were evaluated and led to scores as seen in Table 5.8. The numbers indicate that the rules of the rule set of AnyBURL are dominating the knowledge graph completion task,

so a deeper analysis of the rules is necessary to decide whether combining the rule sets is beneficial in regards to better results in knowledge graph completion.

Table 5.9: Combination of rule sets by inference patterns.

Rule set	**Metrics**				
	Rules	h@1	h@3	h@10	MRR
Symmetry (ExpressivE RM)	81118	3226.	.3226	.3226	.3226
Hierarchy, Inversion, Other (AnyBURL)	50809	.1481	.2368	.3414	.2107
Overall	131927	.4221	.4807	.5420	.4618

Table 5.9 shows the evaluation result if ExpressivE RMs rules and all rules not being symmetry rules of AnyBURL are combined. The numbers indicate that the approaches complement each other well and both approaches contribute to high scores of the used metrics.

A limitation of both combination approaches is that scores of AnyBURL rules and ExpressivE RM rules are not equally scaled because the different scoring functions of the respective approaches were used. Therefore, rules of a certain kind could be preferred from one algorithm over the other.

5.7 Summary

This section describes the experimental setup of the developed algorithm in comparison with other approaches or the combination of rule sets. We assess the performance of the developed algorithm, ExpressivE RM, by training the embedding model and mining rules with it and comparing the algorithm's performance with the state-of-the-art rule miner AnyBURL. First, we introduced the system setup and the parameters used. All experiments are conducted on the same machine to ensure fairness. After that, we presented the experimental results of various combinations of rule sets. To measure performance, the hits@k and MRR scores were used, together with the runtime of the algorithm.

Overall the ExpressivE RM standard approach can mine a rule set in less than 15 minutes. These rules were successfully used in KGC prediction. The algorithm creates 81118 symmetry rules with scores of hits@1, hits@3, hits@10, and MRR of 0.3226 in an explicit format, providing explainability and interpretability. Also, the explicit format allows to add or to remove facts or knowledge from a rule set. Further refinement is needed for the extended approach to realize its potential benefits and extend the search space considerably.

CHAPTER 6

Conclusion

6.1 Discussion

Initially, the implementation started with the model of ExpressivE having 500 dimensions, which quickly raised the question of how to optimize the algorithm due to its high complexity. The ExpressivE RM standard approach was first implemented to show that the geometric interpretation can be used to make the latent knowledge of the KGE explicit by providing a rule set. However, for various reasons, the initial prototype took long to mine these rules.

Firstly, the algorithm was built on Python as the libraries from ExpressivE could be easily used. However, Python quickly showed its extravagant usage of resources, which made several optimizations necessary.

Secondly, the initial algorithm without optimizations did not use multi-threading for its advantage, which was implemented in the development process to drastically decrease processing time (to less than 10% of the initial runtime). This raised another problem: High RAM usage. Since then, RAM usage has been very high, and at first, SWAP space was used to overcome this.

Thirdly, the RAM usage problem was solved by not having shared resources between the threads but by splitting data into independent chunks in the pre-processing step, saving them to the hard disk and then using the worker threads to process chunks. This improvement allowed us to run the whole algorithm on an arbitrarily small system, as small as one core and a few GB of RAM, with the ability to rescale for larger systems automatically and use resources efficiently.

Eventually, the program was refactored and adapted to use 32 dimensions for higher efficiency and to provide the ability to read easily understandable config files instead of adjusting hardcoded values or using extended command line parameters, making implementation of pipelines and parameter variations possible.

A post-processing step was also introduced to provide rules readable by AnyBURL evaluation algorithms written in Java. Finally, a filtering step for collecting statistics as well as improved evaluation of rules was implemented to explore the different kinds of rules and rule sets depending on inference patterns and several entities/variables. This provides insights into the result rule sets and has metrics to compare to state-of-the-art algorithms in a more sophisticated way. All these improvements could minimize the initial runtime of several days to less than 15 minutes on 16 threads and 128GB of RAM for the standard approach.

After promising results, the ExpressivE RM extended approach was implemented to extend the search space, which again led to the necessity of optimization. For example, the data structures were prepared by sorting entities and limiting the search space by considering specific parameters and configurations in the embedding. In future work, we will optimize this extended approach further to make it applicable in prediction tasks.

Ultimately, by theoretically analysing ExpressivE and practically implementing the algorithm to use these theoretical assumptions, we could make the knowledge of the KGE explicit by adding another abstraction layer in the form of rule sets.

Geometrical Interpretation. We used the theoretical foundation of the base model to create simple rules without further aggregations or generalisations of the rules themselves. We see that solely the geometrical representation of the relations and entities provides the information needed to create such explicit rules, dramatically making results' interpretability easier.

Prediction Performance. Furthermore, ExpressivE RM reaches comparable performance as state-of-the-art rule miners, regarding the correctly predicted facts in the test set. At the same time, there is potential to maintain this performance while decreasing the number of rules by generalisation or aggregation. Although the model cannot beat current algorithms, it shows the potential of the approach, which can be extended to capture more advanced inference patterns in the future.

Explainability and Explicity. One main advantage of ExpressivE RM is that it makes implicit knowledge from a KGE explicit and provides quickly understandable rules, making reasoning easier and allowing the possibility to easily remove or add knowledge to a rule set, which is used for prediction tasks.

Scalability and Efficiency. ExpressivE RM follows the principle of divide-and-conquer wherever this is possible. Multi-threading and splitting data for parallelisation make ExpressivE RM scalable and efficient. Due to its design, the algorithm can be run on a personal computer and large computer systems. The standard approach is capable of providing results in a few minutes. Also, it is possible to focus only on specific dimensions to reduce the workload and processing time if only rough data or a first draft of a rule set is needed.

Discussion of results. We have shown that ExpressivE RM is a valid approach for creating rules with comparable performance to state-of-the-art algorithms while providing

results in a short amount of time and making knowledge of a KGE explicit and explainable.

6.2 Discussion of Research Questions

- (RQ1) Can the geometric interpretation of ExpressivE be used to mine rules with prediction performance in knowledge graph completion?
- (RQ2) To what extent can the runtime of ExpressivE RM be improved while maintaining practical applicability?
- (RQ3) In what ways do rule sets generated by ExpressivE RM differ from those produced by state-of-the-art rule miners?
- (RQ4) Can combining rule sets from ExpressivE RM and the state-of-the-art rule miner, boost overall performance in knowledge graph completion?

6.2.1 (RQ1) Can the geometric interpretation of ExpressivE be used to mine rules?

The short answer is yes. We could successfully mine rules on the ExpressivE embedding that lead to meaningful predictions on the test dataset WN18RR. This proves that the geometric interpretation can be used to create meaningful rules. However, some adjustments to the initial assumptions had to be made, such as finding out that rules do not necessarily need to be present in all dimensions. Also, the results show that some dimensions do not contain a single rule candidate. By adjusting these assumptions, the approach ExpressivE RM standard could mine 81118 rules on WN18RR with a hits@1 score of 0.3226, a hits@3 score of 0.3226, a hits@10 score of 0.3226 and an MRR score of 0.3226.

6.2.2 (RQ2) To what extent can the runtime of ExpressivE RM be improved while maintaining practical applicability?

Compared to the original runtime of several days, the final algorithm could be optimized to finish in approximately 13 minutes. This lies within the time range of minutes with AnyBURL. This could be achieved by reducing the number of dimensions, adding the support for processing data in parallel, optimizing RAM usage and organizing data more efficiently in the pre-processing step.

6.2.3 (RQ3) In what ways do rule sets generated by ExpressivE RM differ from those produced by state-of-the-art rule miners?

Compared with our state-of-the-art rule miner AnyBURL and its symmetry rules, the result set of ExpressivE RM standard has 13710 rules in common, which covers 95% of AnyBURLs rules and 17% of ExpressivE RMs rule set. ExpressivE RM is capable of

mining 67408 exclusive rules with hits@1, hits@3, hits@10 and MRR of 0.2773, indicating the ability to correctly predict facts with these exclusively mined rules. However, these rules need to be further inspected, and scoring functions need to be adjusted to reduce the number of rules by removing the ones with low or no ability to predict correct outcomes when used with the data in the test set.

6.2.4 (RQ4) Can combining rule sets from ExpressivE RM and the state-of-the-art rule miner, boost overall performance in knowledge graph completion?

The results show that performance cannot be boosted compared to the state-of-the-art results. A combination of ExpressivE RMs and AnyBURLs rules shows that the metrics do not improve compared to the state-of-the-art approach. Also, replacing the symmetry rules of AnyBURL with ours does not improve the metrics. However, ExpressivE RM showed that its rules are from strong predictive performance in KGC and have the potential to contribute in an improving manner, especially the extended approach, which extends the search space remarkably, promises to provide high-quality rules and uses latent knowledge of an underlying embedding model.

In conclusion, rule mining on the embedding model ExpressivE shows remarkable results and proves that predictive rules can be mined from the geometric interpretation of the model. The rule set of ExpressivE RM standards approach covers 95% of the state-of-the-art rule miner AnyBURL and provides exclusive rules with strong predictive performance in KGC. These findings can be used to limit the search space in a way that provides fewer rules with equal or stronger predictive performance in KGC by adjusting the scoring function of the rules and by extending the search space to mine rules of different types, thus making latent knowledge of underlying embedding models explicit. The assumption that rules must be present in all dimensions should be further inspected in future work since the results show that some dimensions do not even cover one rule candidate. The approximating nature of the underlying model might cause this. However, this knowledge can also be used to reduce the search space and exclude specific dimensions to decrease processing time or get quick first results for new datasets with low computing power.

6.3 Limitations

ExpressivE RM proves that rule mining on embedding models leads to promising results and rules of strong predictive performance in KGC. However, the results are limited by the following factors.

6.3.1 Search Space

Firstly, the search space of ExpressivE RM is currently bound to symmetry rules on datasets. However, the underlying model ExpressivE is capable of capturing a variety of inference patterns. The geometric interpretation of this can be used to extend the

search space and mine more rules that can be used to correctly predict missing facts in a dataset.

6.3.2 Algorithm Efficiency

Secondly, ExpressivE RMs algorithms were written in Python since the base model could be directly imported. However, Python is a very resource-intensive interpreting language, which might not use resources efficiently enough, making many optimizations necessary. By developing the algorithm more efficiently in another programming language, the processing time could be reduced, and it could run on systems with fewer available resources such as CPU cores, RAM and storage.

6.3.3 Scoring

Currently, the result rule set has only rules with the same score. This makes it especially hard to combine the rules with other approaches that have ranked rules. Also, the scoring function might be the reason for the large number of rules, which do not all have the ability to correctly predict missing facts in the data. By adjusting the scoring function, the rules could be ranked and better combined with other approaches, and their number can be reduced, shifting the resulting rule set from a quantity of rules to qualitative rules.

6.4 Future Work

This work shows promising results in rule mining using embedding models and making their latent knowledge explicit. ExpressivE RM can be considered as the foundation for further improvements to mine qualitative rules in a reasonable amount of time and by extending the algorithms and its search space to cover more rule types, which the underlying embedding model provides.

6.4.1 Search Space

The geometric interpretations and scores of rules should be inspected in detail to derive further improvements of the scoring function, limiting the search space so that fewer rules with equal or better ability to correctly predict missing facts are mined.

Also, the number of dimensions a rule candidate is present should be taken into consideration when assigning scores, as the results show that, unexpectedly, this is a metric that influences the ability to correctly predict missing facts. This knowledge can also be used to quickly exclude specific dimensions and gain rules by a trade-off of this capability, which might be reasonable in specific scenarios where speed is more important than a perfect rule set.

Furthermore, the search space can be extended by further optimizing and adjusting the extended approach, providing rules of different types and covering more inference patterns of the underlying embedding model. By optimizing the rule mining process,

the extended approach might follow the schema of the standard approach, which was inefficient in its first draft and needed to be extensively optimized to mine rules efficiently in a short time.

6.4.2 Datasets

To ensure robustness and generalisability, the algorithm should be tested on different datasets. This will help to understand the performance and potential strengths and weaknesses in different contexts and identify limitations or biases that might have been present in WN18RR.

Testing the algorithm on various datasets could determine potential over- or under-fitting. This will also give a better insight into the generalisability of our results. Topics that can be addressed by validating the algorithm on other datasets include the ability to test transfer learning capabilities between datasets and identify similar datasets. Also, a wide variety of data for testing will ensure robustness to changes in the data distribution or other factors. This will allow us to make more informed decisions and adoptions for future versions of ExpressivE RM.

Overall, by using the algorithm on various datasets, its robustness, generalisability, and applicability to real-world problems can be further evaluated.

6.4.3 Optimization

Optimization strategies such as introducing parallel processing, adjusting data chunks according to system resources and preparing data in a pre-processing step to optimize rule mining should be considered for further research. The resource-intense extended approach can cleverly organize entities, relations, and dimensions to save resources and improve processing speed.

Future work should also cover whether the algorithm can benefit from more efficient programming languages and advanced programming techniques to decrease processing time by efficiently allocating available resources.

Also, assigning scores beforehand, possibly due to the geometrical interpretation and the derived scores, might help to optimize the rule mining process to limit the search space reasonably.

6.4.4 State-of-the-Art Embedding Models

Future work can benefit from newer embedding models and optimized variants such as SpeedE [PS24]. A parallel improvement of underlying embedding models and rule miners, such as the proposed one, can benefit both sides, namely the embedding model, by providing insights into the structure of information and type of inference patterns being able to be captured and providing insights in how dimensions, entity, and relation configuration might be further optimized. The rule miners profit by using the increased

power of embedding models to more accurately capture rules and using nuances in geometric compositions to adjust the scoring of rules and the search space overall.

List of Figures

Bibliography

[ABH19] Carl Allen, Ivana Balažević, and Timothy M Hospedales. On understanding knowledge graph representation. *arXiv preprint arXiv:1909.11611*, 2019.

[ACLS20] Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[BAH19a] Ivana Balažević, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. *Advances in Neural Information Processing Systems*, 32, 2019.

[BAH19b] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

[BUG+13] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795, 2013.

[BYRL21] Yushi Bai, Zhitao Ying, Hongyu Ren, and Jure Leskovec. Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. *Advances in Neural Information Processing Systems*, 34:12316–12327, 2021.

[CWJ+20] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*, 2020.

[DMSR18] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings, July 2018. arXiv:1707.01476 [cs].

[DT01] Luc Dehaspe and Hannu Toivonen. Discovery of relational association rules. In *Relational data mining*, pages 189–212. Springer, 2001.

[EI19] Takuma Ebisu and Ryutaro Ichise. Graph pattern entity ranking model for knowledge graph completion. *arXiv preprint arXiv:1904.02856*, 2019.

[EKN+17] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[FHY19] Hironobu Fujiyoshi, Tsubasa Hirakawa, and Takayoshi Yamashita. Deep learning-based image recognition for autonomous driving. *IATSS Research*, 43(4):244–252, 2019.

[Fra20] Alexander L. Fradkov. Early history of machine learning. *IFAC-PapersOnLine*, 53(2):1385–1390, 2020.

[GBP17] Scoff R. Granter, Andrew H. Beck, and David J. Papke. AlphaGo, deep learning, and the future of the human microscopist. *Archives of Pathology and Laboratory Medicine*, 141(5):619–621, 2017.

[GTHS13] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 413–422, New York, NY, USA, 2013. Association for Computing Machinery.

[GTHS15] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730, dec 2015.

[Han02] David J Hand. Pattern detection and discovery. In *Pattern Detection and Discovery: ESF Exploratory Workshop London, UK, September 16–19, 2002 Proceedings*, pages 1–12. Springer, 2002.

[JGO22] Johanna Jøsang, Ricardo Guimarães, and Ana Ozaki. On the Effectiveness of Knowledge Graph Embeddings: a Rule Mining Approach, June 2022. arXiv:2206.00983 [cs].

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kon01] Igor Kononenko. Machine learning for medical diagnosis: History, state of the art and perspective. *Artificial Intelligence in Medicine*, 23(1):89–109, 2001.

[KP18] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):4284–4295, 2018.

[lai16] lain. Freebase is dead, long live Freebase. `https://medium.com/@iainsproat/freebase-is-dead-long-live-freebase-6c1daff44d19`, May 2016. [Online; accessed 28-November-2023].

[LGS20] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. Fast and exact rule mining with amie 3. In *European Semantic Web Conference*, pages 36–52. Springer, 2020.

[Liu13] Xiaozhong Liu. Full-Text Citation Analysis : A New Method to Enhance. *Journal of the American Society for Information Science and Technology*, 64(July):1852–1863, 2013.

[LVZ+19] Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the geometry of probabilistic box embeddings. In *International Conference on Learning Representations*, 2019.

[MCFS20] Christian Meilicke, Melisachew Wudage Chekol, Manuel Fink, and Heiner Stuckenschmidt. Reinforced Anytime Bottom Up Rule Learning for Knowledge Graph Completion, April 2020. arXiv:2004.04412 [cs].

[MCRS19] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3137–3143, Macao, China, August 2019. International Joint Conferences on Artificial Intelligence Organization.

[MDR94] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[NTK+11] Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584, 2011.

[OMP18] Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. Robust discovery of positive and negative rules in knowledge bases. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1168–1179. IEEE, 2018.

[PS23] Aleksandar Pavlović and Emanuel Sallinger. Expressive: A spatio-functional embedding for knowledge graph completion. In *The Eleventh International Conference on Learning Representations*, 2023.

[PS24] Aleksandar Pavlović and Emanuel Sallinger. Speede: Euclidean geometric knowledge graph embedding strikes back. In *Findings of the North American Chapter of the Association for Computational Linguistics*, 2024.

[Qui90] J. Ross Quinlan. Learning logical definitions from relations. *Machine learning*, 5:239–266, 1990.

[RBF+21] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. *ACM Transactions on Knowledge Discovery from Data*, 15(2):1–49, April 2021.

[SC18] Sandeep Subramanian and Soumen Chakrabarti. New Embedded Representations and Evaluation Protocols for Inferring Transitive Relations. 2018. Publisher: arXiv Version Number: 1.

[SDNT19] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[SEM+94] Giovanni Semeraro, Floriana Esposito, Donato Malerba, Clifford Brunk, and Michael Pazzani. Avoiding non-termination when learning logic programs: A case study with foil and focl. In *Logic Program Synthesis and Transformation - Meta-Programming in Logic*, pages 183–198. Springer, 1994.

[SGT+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[TWR+16] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[Uni] Princeton University. What is WordNet? `https://wordnet.princeton.edu`. [Online; accessed 28-November-2023].

[VLMM18] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 263–272, Melbourne, Australia, 2018. Association for Computational Linguistics.

[WGM+14] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526, 2014.

[WMWG17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE transactions on knowledge and data engineering*, 29(12):2724–2743, 2017.

[YYH+14] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[ZCZW20] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3065–3072, 2020.