TU WIEN Informatics

Mining Rules on Knowledge Graph Embeddings: Reducing Rule Set Sizes and Raising Explainability

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Dominik Gail, BSc
Matrikelnummer 11904661

an der Fakultät für Informatik

der Technischen Universität Wien

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

Wien, 3. November 2024

Dominik Gail	Emanuel Sallinger

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

TU WIEN Informatics

Mining Rules on Knowledge Graph Embeddings: Reducing Rule Set Sizes and Raising Explainability

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Dominik Gail, BSc
Registration Number 11904661

to the Faculty of Informatics

at the TU Wien

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

Vienna, November 3, 2024

Dominik Gail	Emanuel Sallinger

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

Erklärung zur Verfassung der Arbeit

Dominik Gail, 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.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 3. November 2024

Dominik Gail

Acknowledgements

I want to thank everyone who supported me on my journey at the TU Vienna. Especially, I want to thank Aleksandar Pavlovic, who supported me throughout this thesis and answered all my questions with patience and without any hesitation. Also, I want to thank my advisor Emanuel Sallinger for introducing me to the exciting topic of knowledge graphs.

I am grateful for my friends and family, who supported me and kept me motivated throughout the years at the university. Without you, this would not have been possible! Thank You!

Kurzfassung

Viele Datenstrukturen in unserem Alltag können als Graphen dargestellt werden. Zum Beispiel stehen wir alle in Verbindungen zu anderen Menschen. Man kann dies als sozialen Graph betrachten, mit den Menschen als Knotenpunkte, die miteinander auf verschiedenste Weise verbunden sind. In solchen Graphen wird es oft der Fall sein, dass nicht alle Verbindungen zwischen den Knoten bekannt sind und hier kann man mit künstlicher Intelligenz nachhelfen. Die Methoden, um unbekannte Verindungen in Graphen zu finden, können wir in verschiedene Gruppen gliedern: Zwei prominente Gruppen sind Regel-basierte und Embedding-basierte Systeme. Erstere sind sehr gut erklärbar und können auch auf neuen Daten angewandt werden, während zweitere eine bessere Leistung aufweisen, aber nicht immer erklärbar sind und nicht auf ungesehenen Daten angewandt werden können.

Wenn wir uns die Vor- und Nachteile von Regel- und Embedding-basierten Systemen anschauen, sehen wir, dass sich diese gut ergänzen. Könnten wir sie verbinden, hätten wir womöglich ein sehr gut erklärbares, hochperformantes und induktiv anwendbares Modell. Zwei konkrete Modelle, die sich dafür gut eignen, sind AnyBURL und ExpressivE. AnyBURL ist ein Regel-basiertes System, das zahlreiche, oft sogar zu viele, Regeln lernt, während ExpressivE ein Embedding-basiertes System ist, das jedoch für das Lernen von logischen Regeln entworfen wurde.

In unserer Arbeit zeigen wir, dass ExpressivE die logischen Regeln von AnyBURL, im speziellen den U_c Regelsubtyp, lernt. Wir entwickeln zwei Priorisierungsmethoden, den Filtered-MRR und den MRR-Evidence Score, sowie Subtypen von diesen, die sich auf die Regelpriorisierung von ExpressivE gründen. Mit dieser Regelpriorisierung können wir wesentlich kleinere Regelsätze, sowohl auf dem WN18RR als auch auf dem InductiveWN18RR Datensatz, auswählen, ohne merklich an Leistung zu verlieren. Damit schaffen wir ein Modell, das sehr gut erklärbar ist, gute Leistung erbringt und sowohl transduktiv, sprich nur auf bereits im Training gesehenen Daten, als auch induktiv, sprich auf ungesehenen Daten, anwendbar ist.

Abstract

With knowledge graph completion methods we can predict missing links in knowledge graphs. Two prominent subgroups of knowledge graph completion methods are rule-based and embedding-based methods - with AnyBURL as an example for the first group and ExpressivE as an example for the second group. Both groups have complemental features: Rule-based methods are explainable and applicable to unseen data, while embedding-based methods are not or less explainable and can not predict links on unseen data. However, the performance of embedding-based methods tends to be better on some tasks.

Suppose we could combine rule-based and embedding-based knowledge graph completion methods, we could have the best of both worlds: State-of-the-art performance, explainability and the ability to generalize to unseen data. One approach to add external knowledge into rule-based methods, for example, from embedding-based methods, is by reranking the logical rules or by selecting smaller rule sets based on a decision criterion extracted from an embedding-based method. A particularly promising embedding-based method for extracting rule priorities is ExpressivE, which was designed to learn logical rules.

In our thesis, we show that ExpressivE is not only capable of learning logical rules, but indeed learns AnyBURL's U_c rules by introducing the novel concept of Rule Conformance and by evaluating ExpressivE's Rule Conformance on the WN18RR dataset. Then, we propose the Filtered-MRR and MRR-Evidence Score, two methods to extract a rule prioritization from ExpressivE. We show that we can drastically reduce the size of rule sets needed to achieve state-of-the-art performance with selection methods based on those scores - both on the WN18RR and the InductiveWN18RR dataset. Therefore, our selection methods are applicable both in the transductive setting, i.e., when each test entity was part of the training set, as well as in the inductive setting, when the test instances were not part of the training set.

With our proposed selection methods, we combine rule-based methods with embedding-based methods, drastically reduce the rule set sizes and thereby increase the explainability of knowledge graph completion methods. Additionally, we use embedding-based methods in the inductive setting, which was previously not possible.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Suppose you are a bank and want to lend money to a creditor. Then, you would want to know whether your creditor is trustworthy or if the money is used for illegal purposes. However, your customers will not tell you that they use the borrowed money for illegal activities - instead, you will have an incomplete web of information about the person and must decide whether to grant the credit.

We can model this incomplete web of information as a graph: We will depict each person, company or activity related to your potential creditor as a node in the graph and call them entities. Then, we will model each relation between two entities with an edge between their corresponding nodes. We will call this graph a knowledge graph and derive certain rules from it. For example, we could reject all potential creditors who are in a business relationship with a fraudulent company. But there is a catch: Your new customers will not tell you this fact as they would be otherwise immediately disqualified from the loan. Here, knowledge graph completion comes into play. Before you decide whether to approve the loan, you try to predict the missing links in the graph, e.g., a fraudulent business relationship.

As you might expect, the principle of knowledge graph completion is not restricted to money laundering. Instead, most knowledge graphs are highly incomplete [MGW+13], and knowledge graph completion can be applied in various fields. For example, we could develop a web crawler that tries to gather information about all celebrities and their relations on this planet. Also, in this case, your generated knowledge graph will miss important relations and could benefit from knowledge graph completion - often even from applying very simple rules. For example, suppose your crawler finds the information that both Michelle and Barack Obama have a child named Natasha. Then you could, with a very high probability, predict that Michelle and Barack are married.

All of these cases have two goals in common: You want to have highly performant models with as few wrong predictions as possible - as, for example, rejecting a potential customer because of a wrong prediction will cost you both money and reputation. Additionally, you want to have an explainable model that is understandable to human beings. This is especially important as regulators will want to know why you accept certain customers and reject others.

For knowledge graph completion, we have three types of models that do not depend on existing domain knowledge: Graph neural networks [SKB+18][VSNT19], (geometric) knowledge graph embeddings [PS22][ACLS20] and rule miners [MCB+24][LGS20], which learn and apply logical rules. While the first two achieve better results, rule learners are, by design, explainable. Geometric knowledge graph embeddings, such as ExpressivE, can be geometrically interpreted, and we could explain the reasoning for their answers. However, tools for generating explanations still need to be developed. Additionally, when we want to achieve state-of-the-art performance with rule miners, we need several thousand rules, even on small datasets, which makes them difficult for human beings to understand. As we see, achieving both goals of knowledge graph completion - or, generally, machine learning - is a challenging task, and our current models either achieve high performance or good explainability, but not both.

Also, there is one more objective: We want our models to be able to generalize to unseen data. For example, if a new potential creditor is added to our network, we do not want to retrain the entire model. Instead, we want to apply our existing rules to decide quickly whether we can trust the customer. We call this setting of generalizing to unseen entities inductive, while we refer to predicting missing relationships only between known entities as transductive. However, the in real-life applications very important task of inductive knowledge graph completion is not trivial to fulfil. The entire class of knowledge graph embedding models can not work with unseen entities as embedding-based models need to learn embeddings for each entity before operating on them. Therefore, interpretable geometrical knowledge graph embeddings with state-of-the-art performance are not usable here.

1.2 Problem Statement

Knowledge graph completion faces the challenge of being explainable, performant and able to generalize to unseen data. All of those challenges have been solved individually. All methods mentioned in the introduction tick two of the three boxes, as shown in Table 1.1. However, no method has been developed that is explainable, achieves state-of-the-art performance, and is usable in the inductive setting.

This thesis aims to provide a new method to combine the strengths of highly performant methods, such as graph neural networks and geometric embeddings, and of the explainable rule miners. In particular, we will look at how to combine the geometric knowledge graph embedding ExpressivE [PS22] with the rule miner AnyBURL [MCB+24] by extracting the priority, or weighting, which ExpressivE assigns to each rule mined by AnyBURL.

	Explainable	Performant	Inductive
Graph neural networks		$\mathcal{X}$	$\mathcal{X}$
Geometric embeddings	$\mathcal{X}$	$\mathcal{X}$	
Rule miners	$\mathcal{X}$		$\mathcal{X}$

Table 1.1: Capabilities of current methods for knowledge graph completion.

Also, AnyBURL assigns a weight to each rule, which acts as a decision criterion on which rule to apply in case of multiple rules fire simultaneously. Therefore, we can answer the following research questions:

1. Does ExpressivE learn logical rules mined by AnyBURL? If yes, how similiar do ExpressivE and AnyBURL weight the individual rules?
2. Can we achieve similar or better performance, measured by the H@1-, H@3-, H@10- and MRR-Scores, by selecting a subset of logical rules based on the weighting of ExpressivE?
3. Can we use ExpressivE to select better rules in the inductive setting, i.e., when predicting relations between entities not previously seen in the training set?

Notably, these research questions build upon each other. We can only introduce a new selection method if ExpressivE assigns a meaning to the logical rules. Also, we can only progress to the more complicated inductive setting when we succeed in the transductive setting.

1.3 Contribution

The main contribution of this thesis is the proposal of novel rule selection methods, which incorporate the weights that ExpressivE assigns to different logical rules. With those rule selection methods, we can drastically reduce the size of rule sets needed for knowledge graph completion while maintaining state-of-the-art performance. Additionally, our new selection methods lead to better explainable predictions and are applicable to the inductive setting. Therefore, they tick all the boxes required for knowledge graph completion: Explainability, state-of-the-art performance, and applicability to the inductive setting.

In our experiments, we used ExpressivE to select a subset of rules mined with AnyBURL. We show that we can achieve similar performance with 22,9% to 79,3% of the original rules, depending on the underlying dataset. As rule miners provide several thousand rules, this is essential to making them entirely understandable for human experts.

Additionally, we empirically show that AnyBURL's rules are reflected in ExpressivE's weights. Thus, we show that ExpressivE is not only theoretically capable of capturing the logical patterns supported by AnyBURL but also learns them. Moreover, we show that

ExpressivE stops improving once it has learned all logical rules, which is an indicator of the importance of rule learning for ExpressivE's performance.

1.4 Structure of the Thesis

In Chapter 2, we familiarize the reader with the most important concepts of knowledge graph completion. We explain the datasets used in the thesis and the scoring methods. Moreover, we take a look at knowledge graph embeddings and rule miners.

We show in Chapter 3 that ExpressivE learns the same rules as AnyBURL and afterwards propose in Chapter 4 the new Filtered-MRR and MRR-Evidence Score, which we will be the basis for our rule selection methods in the following chapters. Also, we show in Chapter 4 the correlation and the differences between AnyBURL's confidence and our MRR-based scores.

In Chapters 5 and 6, we present the results of rule selection with our newly introduced rule selection methods. We start with the transductive setting in Chapter 5. After successfully reducing the rule set size in the simpler transductive setting, we move on to inductive knowledge graph completion in Chapter 6 and show that our approach works there too.

Finally, we conclude this thesis in Chapter 7 by summarizing and discussing the main results. Also, we provide an outlook on future work, which could build upon this thesis.

CHAPTER 2

Related Work

2.1 Knowledge Graph Completion

The term knowledge graph was coined by Google in 2012 [Sin12]. Back then, Google decided to enhance its search engine by treating user inputs not as a string but as entities in a graph with relations to other entities. Thereby, they could provide additional information, like children or siblings, next to the search results.

While the definition of Google's knowledge graph was rather broad, mainly attributing the term knowledge graph to a graph-based knowledge base, we will refer in this thesis to the following more precise definition [EW16]:

> *A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge.*

This definition clearly distinguishes knowledge graphs from knowledge bases and ontologies. We require a reasoning engine in knowledge graphs, which infers additional missing information. This requirement implies that knowledge graphs are incomplete, i.e., crucial information is missing. The task of identifying the missing information is known as knowledge graph completion [XYC+18], and in the following sections, we will discuss different methods for knowledge graph completion and how we can improve them.

Nevertheless, this definition of a knowledge graph necessitates an underlying knowledge base on which we can apply reasoners. Typically, this knowledge base is represented by a set of triples $(h, r, t) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ denotes the set of all entities in the graph and $\mathcal{R}$ the set of all relations [XYC+18]. We refer to h as the head of the triple and to t as the tail of the triple. Each triple represents a directed edge between the head node and the tail node of the relation's type. Knowledge graph completion can then infer the missing edges in the graph by predicting the missing triples [XYC+18].

2.2 Evaluation Metrics

When we introduce a new algorithmic or machine learning task, the first question is how we will evaluate our models. In the case of knowledge graph completion, it is trickier than one might expect to answer this question, as classical machine learning metrics, like accuracy or precision, make little sense in the knowledge graph completion setting. There are several reasons for this: Firstly, the vast majority of potential triples are not in the graph, so if we randomly choose triples for evaluation, most will be negative instances, and a method classifying all as negative would reach very high accuracy scores. Secondly, we do not know which triples are negative as the knowledge graph is incomplete - therefore, a chosen negative triple could be a positive one. Thirdly, finding a cutoff point where we classify a triple as a positive instance is difficult. Some nodes will have many relations to other nodes, while others only have very few. Would we accept each triple above a chosen score or in the top k-predictions? We already see that we need a smarter way of evaluating knowledge graph completion methods than the standard accuracy or recall metrics.

The solution for evaluating knowledge graph completion methods is the introduction of rank-based metrics. Given a positive triple, we compute its rank among all triples sharing the relation and either the head or the tail entity. Triples ranked high are likely in the graph, while we do not expect low-ranking triples to be in the graph. The advantages of this approach are twofold: We can avoid dealing with negative or unknown triples in the evaluation process. Instead, we can look at the rank of the known positives in a validation or test set. Additionally, it fits our use cases of knowledge graph completion methods very well. In reality, often, e.g., when approving a credit, a human expert will look at the predictions before adding them to the knowledge base [LGS20]. However, they will only look at the top-ranked candidate triples because human expertise is costly.

We have yet to define those rank-based metrics formally. In particular, we will look at two metrics used in this thesis: Hits@k and Mean Reciprocal Rank (MRR). For Hits@k, we calculate for an arbitrary k, typically 1, 3, or 10, the percentage of evaluation triples with a rank lower or equal to k. Formally, we can define Hits@k as follows [ABH+20]:

$$\text{Hits@k} = \frac{|\{t \in \mathcal{K}_{test} \mid rank(t) \leq t\}|}{|\mathcal{K}_{test}|} \tag{2.1}$$

While the Hits@k metric only considers triples with a rank lower or equal to k and disregards triples at ranks higher than k, e.g. at position $k+1$, the MRR incorporates all triples in the validation and test set. As the name says, it is computed by calculating the mean over the reciprocal rank [ABH+20]:

$$\text{MRR} = \frac{1}{|\mathcal{K}_{test}|} \sum_{t \in \mathcal{K}_{test}} \frac{1}{rank(t)} \tag{2.2}$$

Hits@k and MRR have a value range from 0 to 1, with a higher score indicating better performance. Notably, both metrics give little weight to outliers. Hits@k treats each triple ranking higher than k the same, and the reciprocal nature of the MRR reduces the influence of very high ranks in the mean. For both Hits@k and MRR, we compute the rank amongst all triples sharing the head or tail entity and the relation. However, there is one caveat: We must filter out the triples from the train set before computing our ranking, as those are most likely to be ranked first and would negatively influence our metric.

2.3 Benchmarks

Several benchmarks of various sizes and domains exist for knowledge graph completion. For example, we have the small Kinships dataset [KTG+06] modelling the relationships between members of the Australian tribe Alyawarra, which contains only 10 686 triples. Then, we have medium-scale benchmarks, like WN18RR [DMSR18] with its 93 003 triples on the semantic relationships between different words, and large-scale benchmarks like FB15k-237 [TC15] and YAGO3-10 [RSH+16], containing 272 115 triples and 1 179 040 triples of cross-domain knowledge.

In this thesis, we will mainly work with WN18RR [DMSR18], as it is large enough to be a challenging task and small enough to deliver fast results on today's hardware. Additionally, it has the advantage of having inductive variants, like InductiveWN18RR [TDH20], which are necessary to evaluate whether we can transfer knowledge from embedding-based methods to rule-based methods in the inductive setting.

WN18RR is a subset of the larger WN18 [BGWB14], which is again a subset of WordNet [Mil95]. In short, WordNet groups English words into so-called synsets, which represent a collection of words which are synonyms of each other. Then, WordNet models the relations between the different synsets. For example, it connects less specific synsets to more specific synsets via the hyponym and hypernym relation type. More specifically, if we had the synset *clothes* and the synset *shirt*, then we could model their relations via the triples *(clothes, hypernym, shirt)* and *(shirt, hyponym, clothes)*.

WN18 includes only synsets with at least 15 triples. Including only those synsets reduces the size of WN18 to 40 943 entities and 151 442 triples and is necessary because our reasoners need data to learn the meaning of each synset appropriately. However, WN18 has a large test-leakage [TC15], and relatively simple models can achieve good results [DMSR18] due to an abundance of inverse triples. This makes WN18 unsuitable for evaluating knowledge graph completion methods and led to the introduction of WN18RR, which removes the inverse triples and poses a more difficult challenge for knowledge graph completion methods.

Category	Definition
Symmetry	$r_1(X,Y) \implies r_1(Y,X)$
Anti-symmetry	$r_1(X,Y) \implies \neg r_1(Y,X)$
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	$r_1(X,Y) \wedge r_2(Y,Z) \implies r_3(X,Z)$
Hierarchy	$r_1(X,Y) \implies r_2(X,Y)$
Intersection	$r_1(X,Y) \wedge r_2(X,Y) \implies r_3(X,Y)$
Mutual exclusion	$r_1(X,Y) \wedge r_2(X,Y) \implies \bot$

Table 2.1: Inference patterns.

2.4 Rule Learning

The most intuitive way to infer new links in knowledge graphs is via logical rules. For example, if we have two arbitrary entities, X and Y, which are in a relation r, we could propose that Y and X are connected via the same relation. This specific rule would be symmetric, and we could define it as $r(X,Y) \implies r(Y,X)$. If we build our reasoning system only upon logical rules, it would be fully interpretable as each new link follows from one or multiple logical rules.

As we see in Table 2.1, we can group those logical rules into several categories [PS22]. While most rules are trivial, we need to highlight the big difference between compositional definition and general composition, which will be relevant in the subsequent chapters: While compositional definition has an *if and only if* operator, in general composition, the body implies the head relation - but not vice versa. Capturing only the second rule type is not trivial, as we will see in Section 2.5.

As finding a well-performing rule set in a large knowledge graph is a tedious process by hand, we employ rule miners, which automatically find and weigh rules. Two state-of-the-art rule miners are AnyBURL [MCB+24] and AMIE [LGS20]. While the performance of AnyBURL and AMIE are comparable, AnyBURL comes with the advantage of being able to mine rules containing constants in the relations. It will turn out in Chapter 3 that constants within rules are helpful to show that ExpressivE can learn logical rules. We will, therefore, focus our attention on AnyBURL in this thesis.

Similar to other rule miners, the rules which AnyBURL can learn are limited by its language bias. For AnyBURL, we can group the rules into three types, which we will call binary rules (B), unary rules ending with a dangling atom (U_d) and unary rules ending with an atom that includes a constant (U_c) [MCB+24]. Formally, we can define them as follows, where we refer to head relations with h, body relations with b_i, variables with A_i and constants with c and c' [MCB+24]:

$$B : \wedge_{i=1}^{n} b_i(A_{i-1}, A_i) \implies h(A_0, A_n) \tag{2.3}$$

$$U_d : \wedge_{i=1}^{n} b_i(A_{i-1}, A_i) \implies h(A_0, c) \tag{2.4}$$

$$U_c : \left(\wedge_{i=1}^{n-1} b_i(A_{i-1}, A_i)\right) \wedge b_n(A_{n-1}, c') \implies h(A_0, c) \tag{2.5}$$

We will likely find patterns within the training set to derive logical rules. For example, if we find out that most entities, which are related via a relation r, are also in a relation s, we could define the binary rule $r(X, Y) \implies s(X, Y)$. This rule will help us better rank the triples in the test set. AnyBURL works similarly - it tries to find potential rules in the knowledge graph and selects the best ones depending on how strong they are reflected in the training set. More formally, AnyBURL applies the following four steps repeatedly to select an appropriate rule set [MCB+24]:

1. AnyBURL samples a random path from the knowledge graph.
2. AnyBURL converts the path into a bottom rule, i.e., a rule, which contains all the constants from the path.
3. AnyBURL generalizes the extracted rule by replacing constants with variables.
4. AnyBURL evaluates all rules and adds the best ones to the rule set.

While how the rule candidates are generated is less critical for our thesis, we will closely examine the process of selecting and weighting the rules for the final rule set. For both selection and weighting, AnyBURL introduces a confidence measure, which is defined as follows [MCB+24]:

$$conf(r) = \frac{|\{\theta_{XY} | \exists \theta_Z \; r_b\theta_{XYZ} \wedge r_h\theta_{XY}\}|}{|\{\theta_{XY} | \exists \theta_Z \; r_b\theta_{XYZ}\}|} \tag{2.6}$$

Here, θ refers to a substitution of variables with constants. We will call such a substitution a grounding. For example, a grounding $\theta_{X=c}$ would replace all occurrences of a variable X with a constant c. In our confidence formula, X and Y refer to the variables in the head atom, while Z refers to the variables in the body atom, which are not part of the head atom. Intuitively, we can think of confidence as the number of groundings for the head *and* body, divided by the number of body groundings.

Suppose the knowledge graph was be complete, i.e., there would be no true triple, which is not part of our database. Then, the confidence would be a perfect measure of the probability that a head relation is part of the knowledge graph when the body of the rule is fulfilled [MCB+24]. In reality, the confidence is a good estimate and a lower bound for the true probability.

When searching for rules, it is important to define a stopping criterion. In AnyBURL, this stopping criterion is time-based and defined before rule generation. Additionally, the authors of AnyBURL only search for B rules with at most three body atoms and U_d and U_c rules with at most one body atom. This is because longer rules increase runtime but do not improve performance. We will also stick to this approach in our work.

AnyBURL uses max aggregation to generate new predictions from the rule set. Max aggregation works by sorting the candidates by the maximum of the confidences of all rules that generate a triple. Suppose there is a tie between two triples, the rule with the second highest confidence acts as a tiebreaker. This approach does not rank triples generated from multiple independent rules higher. However, it also ignores multiple firing rules that are strongly dependent on each other. [MCRS19]

2.5 Knowledge Graph Embeddings

The concept of embedding is familiar to the machine learning community. In several areas, such as in natural language processing [WWC+19] or in genetics [MGH+24], categorical data is embedded into a high-dimensional, numerical space, for example, to use them as input for subsequent neural layers [WWC+19]. In our setting, knowledge graph embedding methods learn a latent vector in a high-dimensional space for each entity and each relation [ABH+20], which should capture their semantic meaning. Knowledge graph embedding allows powerful models but comes at the cost of predicting the links of unseen entities - as their embeddings are not learnt. What makes each embedding method unique is its loss function, which ultimately determines the resulting embedding and the final predictions. We can distinguish between three categories of knowledge graph embedding methods [PS22]:

1. We will call all models capable of capturing compositional patterns **functional models** [PS22]. Generally, those models represent entities as vectors in a high-dimensional field and the relations as functions between those vectors. A triple (h, r, t) is then true, if their embeddings e_h, e_t, and f_r fulfill the equation $e_t = f_r(e_h)$. Two models of this category are TransE [BUGD+13] and RotatE [SDNT19] with their relation functions $f_r(e_h) = e_h + e_r$ and $f_r(e_h) = e_h \circ e_r$, with $\circ$ denoting the Hadamard product.

2. The second embedding model group is **bilinear models** [PS22]. Here, the entity and relation embeddings represent a bilinear factorization of the adjacency matrix of the knowledge graph. The different models, e.g. DistMult [YtYH+14], ComplEx [TWR+16] and TuckER [BAH19], mainly differ in how they perform the matrix factorization, e.g. via the Tucker composition [Tuc66] in the case of TuckER. Notably, all bilinear models cannot capture compositional patterns [PS22].

3. **Spatial models** define semantic regions in the embedding space, which represent relations between entities [PS22]. Defining relations as regions in the embedding

space makes it possible to geometrically interpret those models, which is impossible in functional or bilinear models. BoxE [ACLS20] is an example of a spatial model which embeds each relation as a pair of boxes. A triple evaluates to true if the head and tail embeddings are in the respective boxes. However, also BoxE cannot capture compositional patterns [ACLS20].

One model that stands out from the list above is ExpressivE [PS22]. It combines the advantages of functional and spatial models by defining a hyper-parallelogram in the embedding space for each relation. This embedding method makes it possible to capture compositional patterns and geometrically interpret their predictions. Also, its performance is superior to other functional or spatial models, so we will work in this thesis with ExpressivE.

ExpressivE embeds all its entities via a vector in the embedding space $\mathbb{R}^d$, and assigns to each relation r_i a slope vector $r_i^p \in \mathbb{R}^d$, a center vector $c_i^p \in \mathbb{R}^d$ and a width vector $d_i^p \in (\mathbb{R}_{\geq 0})^d$ for the head and tail positions $p \in \{h, t\}$. A triple then evaluates to true if it fulfills the following inequalities [PS22]:

$$(e_h - c_i^h - r_i^t \circ e_t)^{|\cdot|} \leq d_i^h \tag{2.7}$$

$$(e_t - c_i^t - r_i^h \circ e_h)^{|\cdot|} \leq d_i^t \tag{2.8}$$

In the equalities, $x^{|\cdot|}$ represents the element-wise absolute value of a vector x, $\circ$ the element-wise Hadamard product and $\leq$ the element-wise less or equal operator. As we see in Figure 2.1, these inequalities are fulfilled if and only if the embeddings of the head and tail entities lie within a hyper-parallelogram. Hence, we can interpret ExpressivE geometrically.

In contrast to other embedding models, ExpressivE can capture general compositonal rules [PS22]. Therefore, ExpressivE is a suitable candidate to combine it with rule miners such as AnyBURL, as those mainly generate (general) compositional rules. In our thesis, we will empirically show that this combination is possible, reduces the size of rule sets, and enables embedding-based methods in the inductive setting.

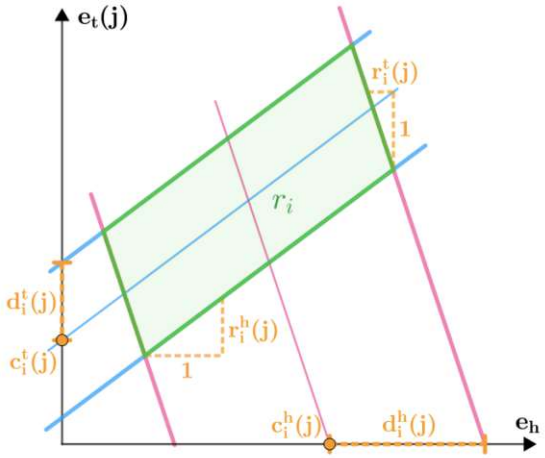

Figure 2.1: Geometric interpretation of the parameters of ExpressivE [PS22].

CHAPTER 3

Rule Conformance

ExpressivE is capable of capturing logical rules of certain types, e.g., it can capture general composition [PS22]. However, it is unclear whether this theoretical ability translates into rules learnt and, more specifically, into rules, which are also captured by rule miners like AnyBURL. In this chapter, we will show that ExpressivE is capable of learning logical rules and, indeed, learns the same logical rules as AnyBURL. We will argue that ExpressivE bases its predictions at least partially on the same logical rules, which ExpressivE potentially weights differently than AnyBURL.

3.1 Rule Conformance

When we want to measure whether ExpressivE learns the same logical rules as AnyBURL, we have two options: Either we look at its predictions and check whether ExpressivE ranks triples derived by AnyBURL's logical rules lower than triples, which can not be deducted from logical rules. Alternatively, we directly analyze ExpressivE's weights, as the logical rules should be encoded within the weights. While we will look at the first option in Chapter 4, in this chapter, we will measure whether the rules mined by AnyBURL are reflected in the weights of ExpressivE.

As discussed in Section 2.5, ExpressivE represents each relation as a hyperparallelogram and two entities are in a relation when their embeddings are within the hyperparallelogram. We can determine whether ExpressivE follows a specific rule by looking at the arrangement of the hyperparallelograms - where the needed arrangement depends on the structure of the logical rule. For example, suppose we have a logical rule $r(X,Y) \wedge s(X,Y) \implies t(X,Y)$, ExpressivE would conform to the rule when the intersection of the hyperparallelograms representing r and s is within the bounds of the hyperparallelogram representing t. In this case, ExpressivE would predict for each possible entity pair, which is in a relation r and a relation s also a relation t.

Our work will focus on one rule type: AnyBURL's unary rules ending with an atom that includes a constant (U_c). In contrast to AnyBURL's other rules, U_c rules contain constants in the body, which enables us to check whether ExpressivE follows the rules by evaluating one or two triples, while we would need a more complex evaluation method for both U_d and B rules. Additionally, we will restrict our experiments to logical rules with a single body atom. While it would be easily possible to translate our approach to rules containing multiple atoms in the body, AnyBURL restricts U_c rules to the single-body atom scenario due to the lack of performance improvement and increased runtime when using more than one body atom [MCB+24].

The following three simple steps can be used to check how strongly U_c rules are reflected within ExpressivE's weights and how strongly ExpressivE conforms to them. We will apply those steps for each dimension separately and compute the mean from all dimensions where a rule is applicable. We will refer to the score resulting from those three steps as rule conformance, as it measures how strongly ExpressivE follows the individual rules.

1. We compute the intersection(s) between the constant in the rule's body and the hyperparallelogram representing the relation in the rule's body. Suppose we have at least one intersection, we move on to Step 2. Otherwise, the rule is not applicable in this dimension, as the body is never fulfilled.
2. We infer new triples from the intersections and the logical rules.
3. For the new triples, we evaluate ExpressivE's loss function (see Equation 2.7 and Equation 2.8).

In our algorithm, a low loss in Step 3 represents conformance to a particular rule. If we want to interpret the rule conformance more closely, we must distinguish the three possible options for the number of intersections in Step 1. Notably, we can have zero, one or two intersections:

- Suppose the constant in the body has no intersections with the hyperparallelogram representing the body's relation. In that case, the body is never fulfilled, and it is impossible to compute any rule conformance.
- A single intersection can only occur at corner points of the hyperparallelogram. Then, the rule is only fulfilled in one point - our intersection - and the loss represents a perfect estimate of how well ExpressivE follows a particular rule.
- Suppose we have two intersections; we evaluate the rule conformance at the boundaries of all entity pairs fulfilling the body of our rule. This case is also depicted in Graphic 3.1. As the maximum loss of the triples inferred from the intersections represents an upper bound for all possible inferred triples, it is enough to evaluate the loss only at the boundaries.

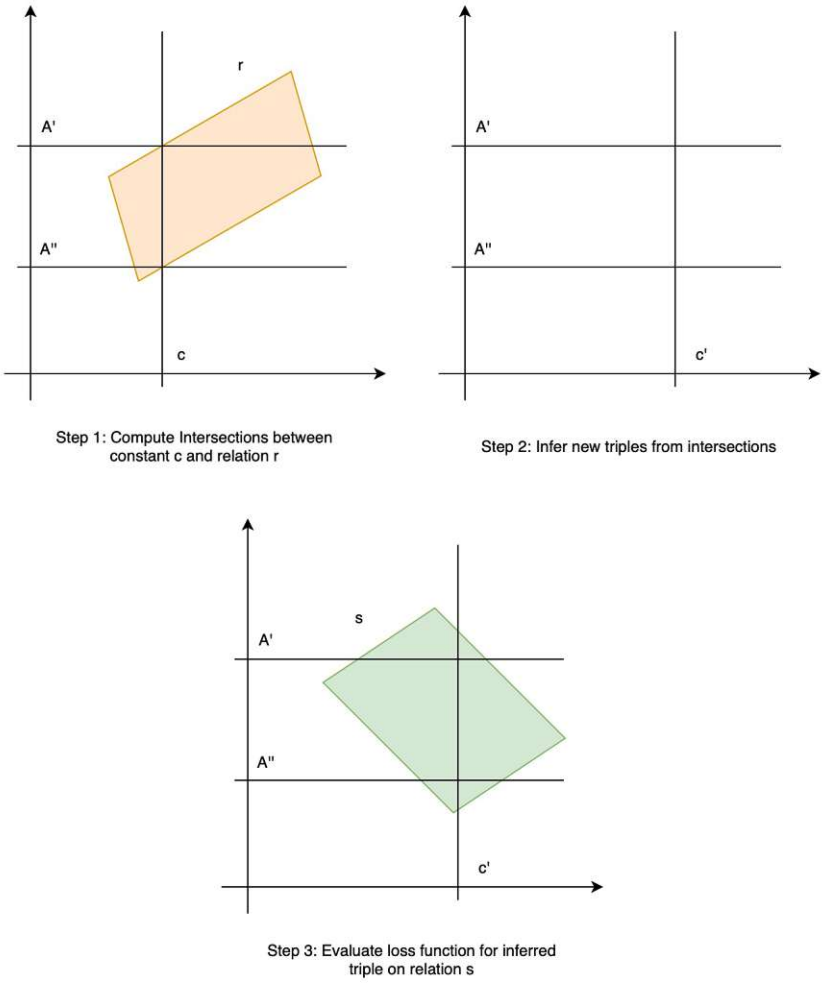

Figure 3.1: Visualization of the process to compute the rule conformance of the rule $r(c, X) \implies s(c', X)$, where the orange parallelogram represents the relation r, and the green parallelogram represents the relation s.

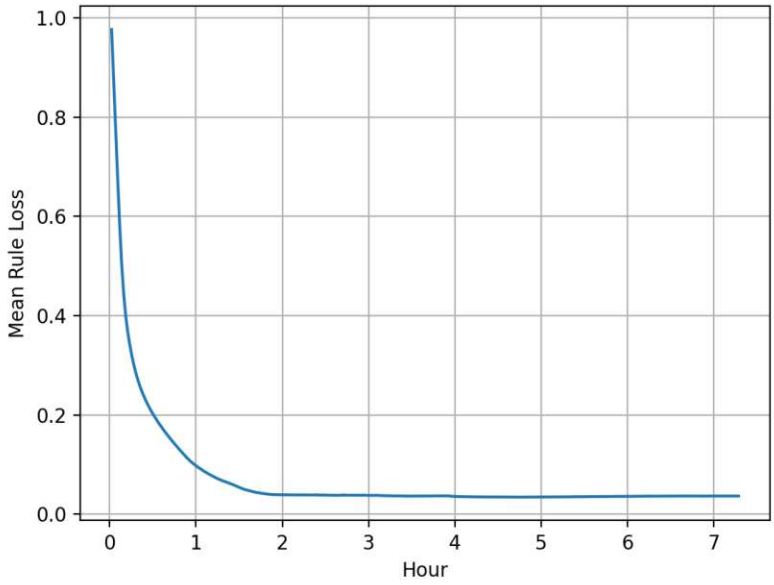

Figure 3.2: Rule conformance respectively ExpressivE's mean loss per rule.

In the next Section, we will utilize our algorithm to determine how strongly logical rules mined by AnyBURL are reflected in the weights of ExpressivE.

3.2 Experiments

To show whether AnyBURL's U_c rules are reflected within ExpressivE's weights, we conducted experiments on the WN18RR dataset [DMSR18]. For our experiments, we mined 42 893 U_c rules over a period of 1000 seconds on a MacBook Pro, 2017, having a 2,8 GHz Quad-Core Intel Core i7. For mining, we will keep those parameters for the entire thesis. We evaluated and logged ExpressivE's rule conformance for each rule after each training epoch, as described in Section 3.1. The results are shown in Graphic 3.2 and show a fast decrease of the mean rule loss, corresponding to a high rule conformance.

However, rule conformance, as described in Section 3.1, is insufficient to show that ExpressivE learns U_c rules. Even with a very low rule loss - representing high rule conformance - it could be the case that our algorithm is never applicable as we do not have any intersections between parallelograms representing the body relation and the constant in the rule's body. Therefore, we additionally tracked how often we have intersections between the rule's body constant and the rule's body relation. The mean percentage of dimensions in which the rule conformance is applicable is shown in Graphic 3.3. Additionally, we added in Graphic 3.4 the validation score (Hits@10) of ExpressivE to show whether it correlates with the rule conformance and the rule conformance

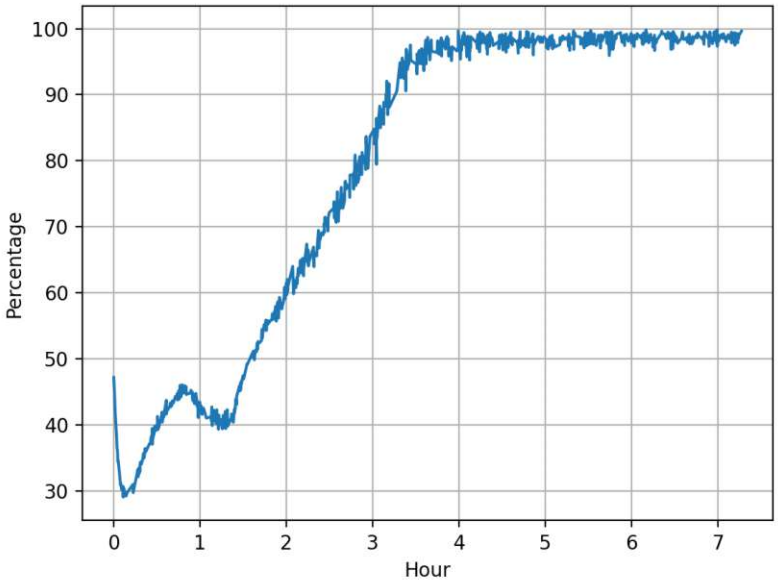

Figure 3.3: Percentage of dimensions in which the rule conformance is applicable, i.e., the constant in the body has an intersection with the rule's body relation.

applicability.

Our results show that within the first hour of training, the U_c rules are reflected within the weights of ExpressivE, at least in the applicable dimensions. However, at initialization, the rules are applicable in only around half of the dimensions, and shortly after training started, the rule conformance is applicable in only roughly a third of the dimensions. While the first fact is easily explainable from the random rule initialization, we expect that the drop in rule applicability stems from an initial trimming phase, where ExpressivE pushes out entities from relations with high rule loss.

After this short trimming phase, and especially after ExpressivE has a very low rule loss in the applicable dimensions, the percentage of applicable dimensions increases continuously, as seen in Graphic 3.3. This trend continues until the rules are applicable and fulfilled in almost all dimensions. Then, no further changes occur in rule conformance and the percentage of applicable dimensions.

Looking at the validation scores, we can recognize a correlation between the validation score and the rule conformance and applicability. The validation score starts to rise when the rule conformance loss plummets around one and a half hours into the training process. Then, it stops rising shortly after the rule conformance is applicable in nearly all dimensions, around the end of the fourth hour.

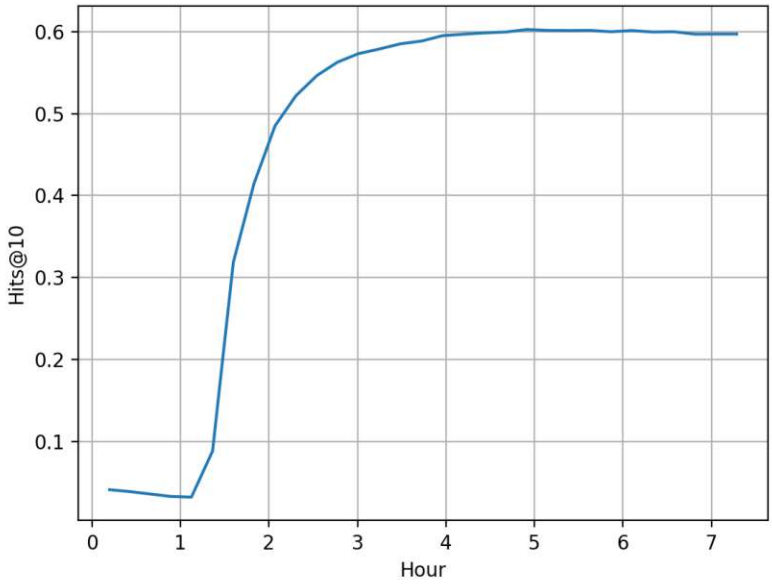

Figure 3.4: ExpressivE's Hits@10 score on WN18RR.

3.3 Discussion

Our experiments show that ExpressivE is not only theoretically capable of learning U_c rules but indeed incorporates AnyBURL's U_c rules within its weights and, thereby, bases its decision at least partially on logical rules.

A critical aspect of basing decisions on logical rules is their weighting. We currently can not say whether ExpressivE weights the rules differently than AnyBURL. However, as ExpressivE has slightly better validation and test scores than AnyBURL, we expect a slightly different weighting of the rules. If this was the case and we can extract the weights of ExpressivE, we expect to be able to improve predictions based on logical rules by utilizing the superior weights of ExpressivE. We will follow this hypothesis in the following chapters.

The second important finding in this chapter is the correlation between the validation score and the rule conformance. We consider this finding unexpected as we would have thought that logical rules are only partially responsible for ExpressivE's success. However, it still leads to three possible explanations:

1. Most of WN18RR's validation and test triples can be predicted from simple logical rules.

2. ExpressivE bases its decision mostly on learnt logical rules.

3. We have a random correlation on the WN18RR dataset, which is neither caused by the simplicity of WN18RR nor by ExpressivE's learning process.

While it is unclear which of these explanations is correct, it would be interesting to check in the future whether the validation score and the rule conformance also correlate on larger, more complex datasets. If this were the case, it would indicate that ExpressivE strongly incorporates logical rules into its weights. If not, it would open a new research direction to find other factors contributing to ExpressivE's predictive success.

Additionally, it will be important to show whether ExpressivE also follows other logical rules in the future. In this chapter, we only showed that ExpressivE learns U_c rules. Theoretically, ExpressivE can learn all general composition rules [PS22], to which U_c, U_d, and B rules belong. While we would expect that ExpressivE also follows U_d and B rules, we leave it to future researchers to show whether this hypothesis is true.

CHAPTER 4

Weight Correlation

In this chapter, we will introduce a novel approach, the Filtered-MRR Score, to evaluate whether knowledge graph embedding methods like ExpressivE base their predictions on logical rules, specifically AnyBURL's B rules. B rules differ from U_c and U_d rules by containing only variables and no constants. The missing constants make extracting the rule conformance directly from the weights more difficult, as we did in Chapter 3. However, it opens the path of extracting the rule conformance from its predictions as B rules generally have more predictions than U_c and U_d rules. Therefore, we will evaluate AnyBURL's B rules on how ExpressivE ranks triples following from logical rules, while we took the other route in the last chapter.

Our method returns how ExpressivE weights different rules and shows that ExpressivE's weight correlates with AnyBURL's weight. We will combine both weights in the novel MRR-Evidence Score, thereby merging the best aspects of AnyBURL's and ExpressivE's weight. The new MRR-Evidence Score will then be the base building block for the MRR-Evidence-Selection, introduced in the next chapter.

4.1 Filtered-MRR Score

We can measure how embedding-based methods weight different logical rules by looking at how they, on average, rank triples following from logical rules. We introduced one scoring method, the mean reciprocal rank, short MRR, in Section 2.2. This scoring method returns high values when triples are ranked low, low values when most of the triples are ranked high and is resistant to outliers - thereby making it a good measure to evaluate knowledge graph completion methods in general and a good measure to evaluate whether triples following from specific logical rules are ranked lower than triples following from other logical rules. Therefore, we will utilize the MRR to introduce the Filtered-MRR Score, which evaluates the MRR for all triples following from logical rules.

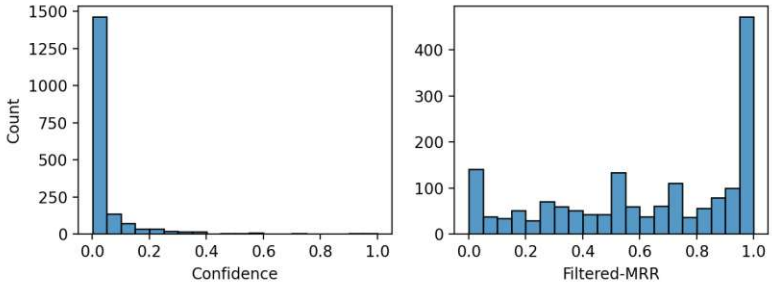

Figure 4.1: Distribution of the Confidence and Filtered-MRR Scores for AnyBURL's B rules on the WN18RR dataset.

In our case, before we evaluate the MRR, we filter all triples already contained in the training set - leading to the Filtered prefix.

When we generate the Filtered-MRR Score for our rules, the first question is which triples to compute it with. We want to compute it on as many triples as possible, and of course, we can not compute the Filtered-MRR on the validation or test set due to potential leakage. In our work, we randomly split the training set into five equally sized parts. For each split, the Filtered-MRR is computed on the triples in the split, which follow from the remaining four splits and the logical rule. Also, for training ExpressivE, we will use the remaining four splits. With this approach, each triple contributes to the Filtered-MRR, and ExpressivE is always well-trained when the Filtered-MRR is generated. We consider five splits as a reasonable tradeoff for keeping a relatively large training set for training ExpressivE, while keeping the number of training iterations small.

The last question to address is how to combine the MRR scores of the different splits and how to handle splits in which not a single triple is predicted from a logical rule. For combining, we propose to use max-aggregation, keeping the highest MRR score of all five splits. Max-aggregation also automatically handles splits without predictions - however, we will come back to the number of splits containing predictions from logical rules in Section 4.3 as it will turn out to be helpful information for determining the strength of a rule.

4.2 Correlation between Confidence and Filtered-MRR

For our experiments, we mined 1810 B rules on the WN18RR dataset, as in the previous chapter, on a MacBook Pro, 2017, with a 2,8 GHz Quad-Core Intel Core i7 over a period of 1000 seconds. When we compare AnyBURL's confidence with the Filtered-MRR score, we immediately spot one difference: While we have an abundance of very low confidence rules, with confidences close to 0, the Filtered-MRR is more evenly distributed, but the distribution is still spiking at values close to 1, which is visible in Graphic 4.1.

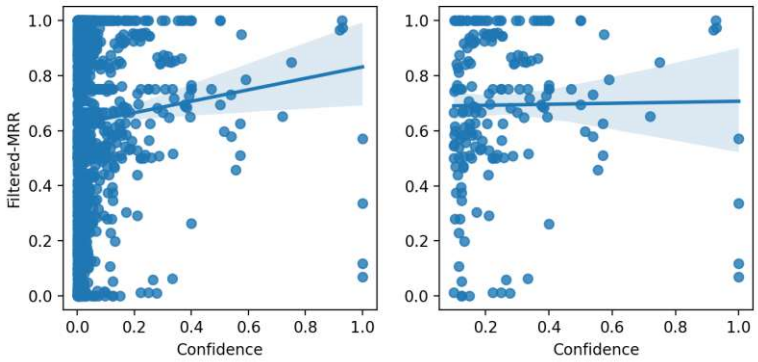

Figure 4.2: Correlation between Confidence and Filtered-MRR Scores for AnyBURL's B rules on the WN18RR dataset.

The different distribution does not come by surprise as the nature of the confidence and the Filtered-MRR is different: While the confidence divides the number of supporting triples by all possible predicted triples, often many thousands, the Filtered-MRR bases its scoring on triples, which should normally be learnt implicitly by ExpressivE. Consequently, we expect relatively high Filtered-MRRs and relatively low confidences.

In our analysis, we will consider this huge difference in distribution by plotting all rules and rules with a confidence higher than 0.1 in separate graphs. This has two advantages: First, the additional graph is better readable, as we have many low-confidence rules, which overshadow the few high-confidence rules. Second, we can look more closely at the high-confidence, high-impact rules responsible for many predictions, which should give us insights into whether ExpressivE agrees with AnyBURL on the weights of the most used rules.

In Graphic 4.2, we can see the results of our comparison between AnyBURL's confidence and our new Filtered-MRR Score. In the left graph, we see that, on average, ExpressivE ranks triples following from rules with very low confidence higher than triples following from rules with high confidence. However, there is a huge variety in the Filtered-MRR scores, and even several rules with confidence close to 0 have Filtered-MRR scores close to 1. However, this is not entirely unexpected. Suppose a low-confidence rule produces a single triple, which, by chance, ranks low as other high-confident rules also predict it. Then, the low-confidence rule would have a high Filtered-MRR score, although ExpressivE did not necessarily learn the rule.

The correlation completely disappears when we compare only high-confidence rules, as in the right graph in Graphic 4.2. Many rules with confidences around 0.1 to 0.3 still have very high Filtered-MRR scores, while a few with very high confidences have Filtered-MRR scores around 0.1. Therefore, we would expect ExpressivE to weigh the higher-confidence rules differently than AnyBURL. While these experiments are only

about the correlation of both scores and empiric evaluation will follow in Chapter 5, we already expect that both confidence scoring and Filtered-MRR scoring miss crucial parts from the other scoring method and could benefit strongly from each other. For example, the Filtered-MRR scoring could benefit from filtering out low-confidence rules, while the confidence scoring could benefit from adjusted weights of high-confidence rules. Therefore, we will try to combine both scoring methods in the next section.

4.3 MRR-Evidence Score

We have seen that confidence and Filtered-MRR have advantages: While the first can detect rules with little impact, the second can incorporate the weights of embedding-based methods, like ExpressivE. Here, we will combine both in the MRR-Evidence Score, which will be used in the following chapters to select the best logical rules to predict new triples.

The straightforward approach to combine AnyBURL's confidence and the Filtered-MRR is to multiply them. However, for the MRR-Evidence Score, we add one additional tweak: As discussed in Section 4.1, not all splits used in the Filtered-MRR computation contain triples following from specific logical rules. Therefore, the Filtered-MRR is not applicable to those splits. In the MRR-Evidence Score, we additionally divide by the number of non-applicable splits, i.e., the number of splits where we can not compute the Filtered-MRR, which leads to the following formula for the MRR-Evidence Score:

$$\text{MRR-Evidence Score} = \frac{\text{Confidence} \times \text{Filtered-MRR}}{1 + |\text{Non-applicable splits}|} \tag{4.1}$$

The factor $\frac{1}{1+|\text{Non-applicable splits}|}$ represents an estimation of the likelihood that the rule is applicable in the validation and test set. Although this formula adjustment does not incorporate any weights of ExpressivE, we consider it important as we will show in the next chapter that it improves our ability to select the best rules from a given rule set.

4.4 Discussion

In this chapter, we introduced the novel Filtered-MRR Score, a rule weighting purely based on the priority an embedding-based method like ExpressivE puts on the individual rules. While the distribution of the Filtered-MRR Score extracted from ExpressivE and the rule weighting generated via AnyBURL differ starkly, there is still a correlation between them. Therefore, we expect that both rule weightings can differ between important and less important rules.

When we look only at AnyBURL's high-confidence rules, the correlation between the rule weightings disappears. Therefore, we expect that the Filtered-MRR Score captures additional information to which AnyBURL has no access. To combine both weightings, we introduced the MRR-Evidence Score, which we will test in the following chapters. With this scoring method, we combine the strengths of both methods: We add additional

information to AnyBURL's high-confidence rules and do not prioritize all rules with a high Filered-MRR Score, which we have in abundance.

CHAPTER 5

Rule Selection

When we perform knowledge graph completion with rule miners, we face two challenges: We want high validation and test scores, and we want to achieve these scores with small rule sets to keep the model explainable. However, both goals are contrary to each other. More rules normally lead to better test scores, and therefore, we need to sacrifice the additional goal of small rule sets to match the test scores of embedding-based models.

In this chapter, we will introduce several rule selection methods, which select smaller rule sets while keeping high test scores. Specifically, we will introduce the MRR-Selection and the MRR-Evidence-Selection, which build upon the scores developed in the last chapter. Additionally, we will add extensions to those selection methods and find out which method performs best. We will show that our new selection methods allow us to drastically reduce the rule set size without sacrificing the performance of the model. Thereby, we will create an explainable and well-performing knowledge graph completion model.

5.1 Reranking Rules

When two different logical rules predict two different triples, we face a decision: Which triple should we rank first, and based on which criterion should we prefer one triple over the other? AnyBURL's way of dealing with such problems is by using the confidence score. Each rule is associated with a confidence score, the rule's priority. If two rules predict the same triple, then one rule will have a higher confidence and the triple predicted from the high-confidence rule is ranked before the triple following from the low-confidence rule. So, the rule set has a total order, and each rule has a rank in this set, with higher-ranked rules being more important than lower-ranked rules.

Suppose we had a different decision criterion than the confidence, for example, the MRR-Evidence Score introduced in Section 4.3, we could consider this a reranking of the

	Hits@1	Hits@3	Hits@10	MRR
Confidence	0,4164	0,4703	0,5226	0,454
MRR-Evidence Score	0,4164	0,4708	0,5216	0,4538
Equal Rank	0,4164	0,4703	0,5226	0,454
Random Rank	0,4164	0,4703	0,5226	0,454

Table 5.1: Validation scores for different rule rankings of AnyBURL's B rules on the WN18RR dataset.

rules. If this reranking better reflects the true priorities of the rules, we would expect that predictions based on the reranked rule set have higher validation and test scores. Finding better rule rankings, potentially incorporating information from other knowledge graph completion methods like embedding-based methods, therefore, promises to improve rule-based methods.

We tested this hypothesis on the WN18RR dataset. We reranked the 1810 B rules, which we mined in the previous chapter with AnyBURL, with the MRR-Evidence Score. We restricted our reranking experiments to B rules as we can not compute the MRR-Evidence Score for U_c and U_d rules. In addition to reranking by the MRR-Evidence Score, we also tested how ranking all rules equally or applying random weights affects the scores. With this, we want to show whether the rank influences the WN18RR dataset.

The results are shown in Table 5.1. We see that reranking by the MRR-Evidence Score does not improve our results. However, neither do random or equal weights worsen the scores. Instead, our results show that the rule prediction step ignores the weights on the WN18RR dataset - or at least, they do not influence the returned triples.

AnyBURL's ignorance of the rule ranking comes by surprise. We would have expected that ranking rules with the MRR-Evidence Score improves and ranking them randomly hurts the scores. Also, the AnyBURL's authors express concerns about finding the correct confidence scores [MCB+24]. Therefore, we would have expected that getting them wrong - or applying random weights - harms the model. In the future, the influence of the rule rank should, therefore, be retested on larger datasets, like FB15k-237 [TC15] and YAGO3-10 [RSH+16]. We expect that the small dataset is responsible for the complete ignorance of the rule ranking. However, as we use WN18RR in our work, we will shift to rule selection, which we consider a better choice for comparing the different rule scores.

5.2 Rule Selection Methods

One of the main reasons to use rule-based methods for knowledge graph completion is their explainability. However, the explainability sinks when the rule set size increases, and rule-based methods need large rule sets to achieve accurate predictions on current knowledge graph completion datasets. For example, in our experiments, we mined 1810 B and a combined 64961 U_c and U_d rules with AnyBURL to achieve state-of-the-art

	Selection Criterion
Confidence-Selection	Confidence
Extended-Confidence-Selection	$\frac{\text{Confidence}}{1+\lvert\text{Non-applicable splits}\rvert}$
MRR-Selection	Filtered-MRR
Extended-MRR-Selection	$\frac{\text{Filtered-MRR}}{1+\lvert\text{Non-applicable splits}\rvert}$
MRR-Confidence-Selection	Confidence $\times$ Filtered-MRR
MRR-Evidence-Selection	$\frac{\text{Confidence}\times\text{Filtered-MRR}}{1+\lvert\text{Non-applicable splits}\rvert}$

Table 5.2: Rule selection methods.

performance. While it is already hardly possible to evaluate 1810 rules by hand, checking tens of thousands of rules is practically impossible and renders AnyBURL, even for small datasets, almost unexplainable. Therefore, we would like to find rule selection methods that reduce the rule set size without sacrificing the test scores.

In our work, we introduce new rule selection methods and evaluate them on two tasks: Firstly, we want to find the smallest possible rule set size while keeping the scores at an acceptable range, i.e., we will choose the smallest rule set size, where the MRR drops by at most 1% on the validation set. Additionally, we want to find a rule set that reduces the rule set sizes but keeps almost identical scores. Therefore, we will evaluate a second rule set, where the MRR drops by at most 0,5% on the validation set. Thereby, we will show how many rules we can drop and still achieve acceptable or even close-to-perfect scores.

Table 5.2 shows the different rule selection methods. Each selection method is based on a different score, which orders the rules differently and picks the top rules based on the decision criterion. In our experiments, we will compare both simple methods, such as the Confidence- and MRR-Selection, as well as the MRR-Confidence- and MRR-Evidence-Selection, which combine AnyBURL's confidence score and the Filtered-MRR within one score. Additionally, we evaluate extensions of the Confidence- and MRR-Selection. Those extensions are built similarly to the MRR-Evidence-Selection. We randomly split the training set into five parts. Then, we divide our original scores, i.e. the confidence and the Filtered-MRR, by the number of splits without triples following from a logical rule and the remaining four splits. Thereby, we extend the Confidence- and MRR-Selection by an estimation of the likelihood that a rule will be applicable to the validation and test set.

5.3 Experiments

In our experiments, we worked with 1810 B rules for the WN18RR dataset, which we mined with AnyBURL. We refrained from adding U_c and U_d rules as not all rule selection methods listed in Table 5.2 are usable on U_c and U_d rules. We selected subsets of different sizes from our rule set and evaluated the Hits@1, Hits@3, Hits@10, and MRR scores on

#Rules	Selection Method	Hits@1	Hits@3	Hits@10	MRR
500	Confidence-Selection	0,41	0,4638	0,5096	0,4452
	Extended-Confidence-Selection	**0,4102**	**0,4681**	**0,5187**	**0,4481**
	MRR-Selection	0,3409	0,3539	0,3574	0,3479
	Extended-MRR-Selection	0,3757	0,4052	0,4194	0,3926
	MRR-Confidence-Selection	0,4075	0,4558	0,4923	0,4379
	MRR-Evidence-Selection	0,41	0,4633	0,5085	0,4447
700	Confidence-Selection	**0,4108**	0,4684	0,5182	0,4486
	Extended-Confidence-Selection	0,4107	**0,4698**	**0,5227**	**0,4496**
	MRR-Selection	0,3435	0,3588	0,3625	0,3516
	Extended-MRR-Selection	0,4046	0,4564	0,4876	0,4354
	MRR-Confidence-Selection	0,4092	0,4646	0,5105	0,445
	MRR-Evidence-Selection	0,4105	0,4692	0,5201	0,4488
1810	Not applicable	0,4107	0,4703	0,5281	0,4513

Table 5.3: Test scores of the different rule selection methods on the WN18RR dataset.

the validation set for each subset size. The results for all selection methods are shown in Graphic 5.1.

On the validation set, the Extended-Confidence-Selection outperforms all other selection methods, and we expect that we can drastically reduce the rule set size without losing much performance. While the MRR-Evidence-, Confidence, and MRR-Confidence-Selection also achieve good results, the MRR- and Extended-MRR-Selection achieve inferior results and need almost all rules to achieve high validation scores.

For testing, we have two cutoff points: Once, we will select the best 500 rules with each selection method, and once, we will select the best 700 rules with each selection method, as those are the points where the best-performing selection method, the Extended-Confidence-Selection, achieves MRRs of 99% and 99,5% compared to the MRR of the entire rule set.

Table 5.3 shows the test scores for all selection methods. There, we see that, as in the validation set, the Extended-Confidence-Selection performs best, closely followed by the MRR-Evidence-Selection. On the test set, we achieve both goals of a 0,5% and a 1% deviation of the MRR. We achieve a MRR of 0,4481 with 500 rules, equaling 99,29% of the MRR achieved on the whole rule set. With 700 rules, we achieve a MRR of 0,4496, equaling 99,6%. So, we can reduce our rule set on the WN18RR by 72,38% and 61,33% without losing more than 1% and 0,5% on the MRR.

5.4 Discussion

In this chapter, we compared different rule-scoring methods. The obvious choice for comparing them is to use them for reranking rules. We tried to rerank rules with the

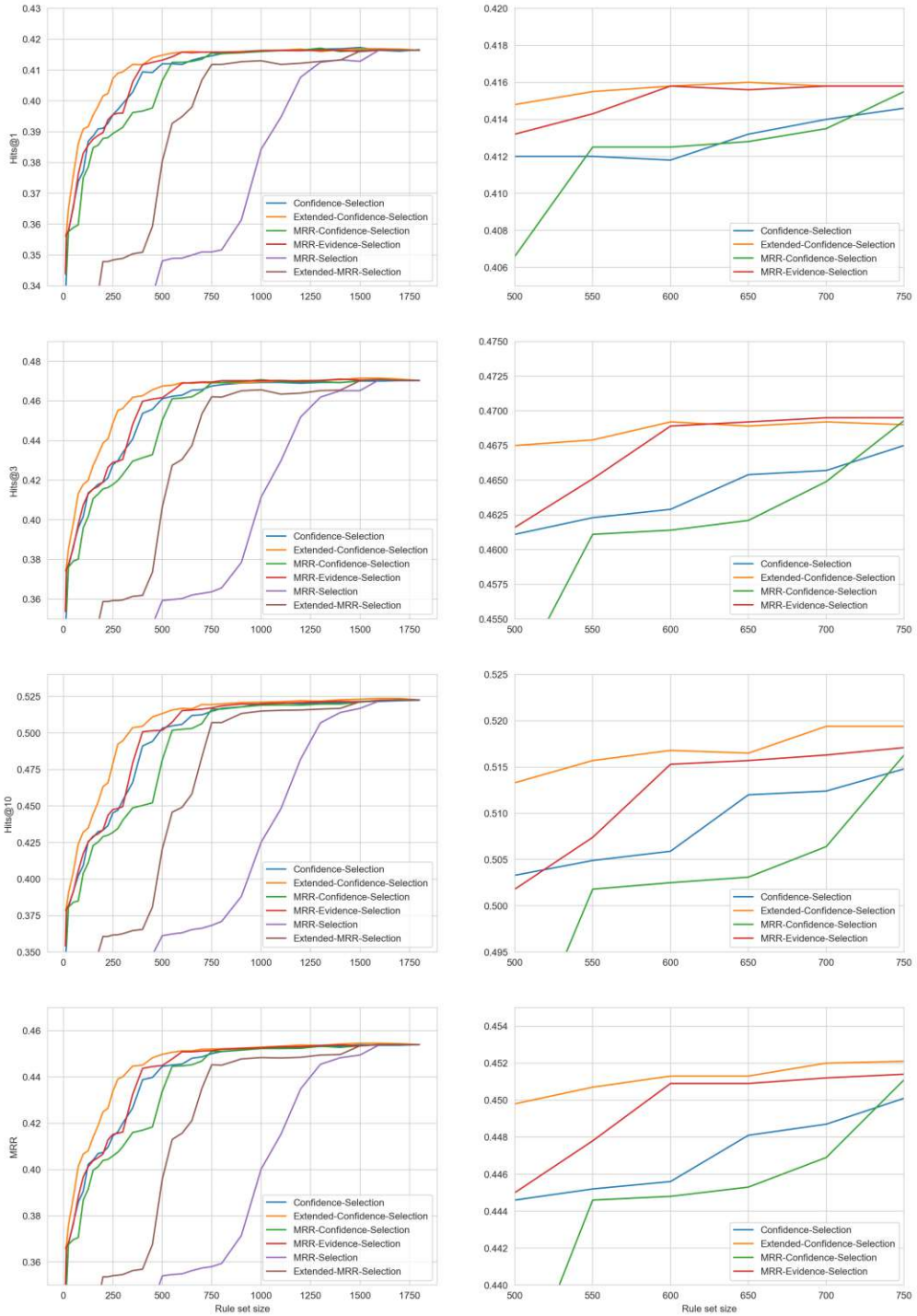

Figure 5.1: Validation scores for the different rule selection methods on the WN18RR dataset.

MRR-Evidence Score in Section 5.1. However, it turned out that reranking rules with the MRR-Evidence Score does not change the scores on the WN18RR dataset - and neither does a random reranking. As our results in this section are unexpected, further research will be necessary. We expect this phenomenon to be visible only on small datasets like WN18RR, and reranking will influence larger datasets like FB15k-237 [TC15] and YAGO3-10 [RSH+16].

While improving the scores by reranking the rules was impossible, we drastically reduced the rule set sizes with our newly proposed rule selection methods. By reducing the rule set size, we introduced better explainable and less computationally expensive rule sets. Further, some of our new selection methods are computationally inexpensive. For example, we can perform the Extended-Confidence-Selection with our given confidence scores and by computing in how many splits - out of five - triples follow from a logical rule and the remaining for splits.

When we compare our two cutoff points for selecting the rule set sizes, we see that we can significantly reduce the rule set size from 700 to 500 by reducing our MRR target by only 0,5%. However, also the reduction from 1810 to 700 rules with a target MRR drop of at most 0,5% and a test drop of 0,4% represents a rule set reduction of 61,33%, which shows us that we can achieve almost the same results with only a minority of the rules.

Comparing the different selection methods, we see that the Extended-Confidence-Selection works best on the WN18RR dataset, both on the validation and test set. However, it is closely followed by the MRR-Evidence-Selection and future research will need to show which method performs best. The best selection method will probably depend on the underlying task, and we will check this assumption in the next chapter.

Filtered-MRR-based scores, which do not consider AnyBURL's confidence, perform worse than confidence-based scores on the WN18RR dataset. We expect this is caused by the failure to select a few high-confidence rules responsible for many predictions. This assumption is backed by the results of the validation scores when we select very small subsets of rules with high confidence. With only the ten rules with the highest confidence, we achieve validation scores only 20-34% lower than with the complete rule set, and the scores climb rapidly when adding a few additional rules. Filtered-MRR-based selection methods would miss several high-confidence, high-impact rules and perform weakly. In this chapter, we conclude that confidence is a measure which should generally be included when selecting rules. However, in the next chapter, we will test whether this is true for all datasets or if the dependence on the confidence is dataset-specific.

CHAPTER 6

Rule Selection - Inductive Setting

In machine learning, we often want to make predictions on previously unseen data. However, this is impossible with embedding-based methods as those need to learn an embedding for each instance in the training phase. Therefore, embedding-based methods, like ExpressivE, are not usable in the inductive knowledge graph completion setting, where the training and test graphs contain different entities. Rule-based methods, on the other hand, can easily learn logical rules on a training graph and transfer those rules to test graphs, as long as those rules do not contain constants. For example, we can use AnyBURL's B rules in the inductive setting.

As embedding-based methods typically perform better than rule-based methods, we would like to transfer the knowledge learnt by embedding-based methods in the training phase onto the rules learnt by rule miners. Then, we could also benefit from the superior performance of embedding-based methods in the inductive setting. In Chapter 5, we introduced four rule selection methods which perform precisely this task. The MRR-, Extended-MRR-, MRR-Confidence-, and MRR-Evidence-Selection all include rule weights extracted from ExpressivE to select smaller rule sets. Here, we will show that those methods can also drastically reduce the rule set sizes in the inductive setting. For this, we will train AnyBURL and ExpressivE on the InductiveWN18RR [TDH20] datasets, an inductive variant of the previously used WN18RR. Then, we will test how far we can reduce the rule sets while keeping the test scores in an acceptable range, i.e., at 99% and 99,5% of the entire rule set's scores.

6.1 Datasets

The InductiveWN18RR dataset provides four different knowledge graphs (v1-v4), which have a similar structure as WN18RR [TDH20]. Each variant consists of three disjoint subgraphs of WN18RR, a train, validation and test graph, which do not share any entities. However, the train, validation and test graphs contain the same relations within one

		#relations	#entities	#triples
v1	train	9	922	1618
	valid	9	290	185
	test	9	286	188
v2	train	10	2757	4011
	valid	10	661	411
	test	10	710	441
v3	train	11	5084	6327
	valid	11	882	538
	test	11	975	605
v4	train	9	7084	12334
	valid	9	2205	1394
	test	9	1429	2270

Table 6.1: Statistics of the different knowledge graph variants in the InductiveWN18RR dataset.

variant. The subgraphs were created by sampling different entities as roots and by taking the union of the k-hop neighbourhoods around the roots [TDH20]. Each subgraph has a different size and was created with different hyperparameters to test how the knowledge graph completion methods perform under different circumstances. The properties of the different variants are listed in Table 6.1.

In our work, we evaluate the different rule selection methods on all variants contained in the InductiveWN18RR dataset to evaluate whether they work equally on graphs of different sizes.

6.2 Experiments

In our experiments, we tested the rule selection methods listed in Table 5.2 in the inductive setting by performing the same experiments as in Section 5.3 on each variant contained in the InductiveWN18RR dataset. For each variant, we mined rules with AnyBURL on the training graph, with the same machinery and time constraints as in the previous chapters. We restricted the learning to B rules as only those are applicable in the inductive setting. Table 6.2 shows the number of mined rules for each variant.

Then, we randomly split each training graph into five parts and computed the Filtered-MRR scores for each rule, as we did in the previous experiments. However, while we took the hyperparameters for ExpressivE in the previous experiments from the authors of ExpressivE, we finetuned ExpressivE ourselves for each variant. The hyperparameters for the different datasets are shown in Table 6.3.

We computed the validation scores for the different variants to select rule set sizes that target a maximum MRR drop of 0,5% and 1%. The validation scores are shown in

Variant	#rules
v1	483
v2	490
v3	387
v4	843

Table 6.2: Number of mined rules for the different variants in the InductiveWN18RR dataset.

	Batch Size	Learning Rate	Adversarial Temperature	Number of Negatives	Margin
v1	256	10^{-3}	1	100	2
v2	512	10^{-3}	1	200	2
v3	512	10^{-3}	1	100	1
v4	512	2×10^{-3}	1	150	1

Table 6.3: Hyperparameters used for training ExpressivE on the different variants of the InductiveWN18RR dataset.

Graphic 6.1 and Graphic 6.2. Here, we already see differences between the different variants. While we can easily match and exceed the validation scores of the entire rule set with around a third and a half of the rules for v1 and v2, we need more than half of the rules in v3 and v4 to achieve an acceptable MRR on the validation set.

When we look at the performance of the different rule selection methods, we see that they perform differently on the different datasets. For example, the Extended-MRR-Selection performs far better than all other selection methods on v1, outperforming the original model with 100 rules less than all other selection methods. However, the Extended-MRR-Selection does not outperform the other selection methods on the other three variants. All variants have in common that the Confidence-Selection and the MRR-Selection, the only selection methods without a second information component, perform worst. In contrast, the other four variants perform similarly well - except the Extended-MRR-Selection in v1. Therefore, we do not have a clear winner from the validation scores, and the best selection method depends on the variant and the targeted rule size.

In Table 6.4 to 6.7, we show the test results for a MRR score target of 99% and 99,5% compared to the entire rule set for each dataset variant. For v2 and v3, we test only one rule set size as rule sets with 225 and 275 rules are the last ones above the 0,5% and 1% threshold. The results show that we are above our target at the 0,5% threshold in v1 and above both targets in v4. For v4, we even match the results of the complete rule set at the 0,5% target. However, we fail to reach the 1% target in the test scores in v1 and v2 with drops of 1,75%, 1,63% and the 0,5% target in v3 with a drop of 1%.

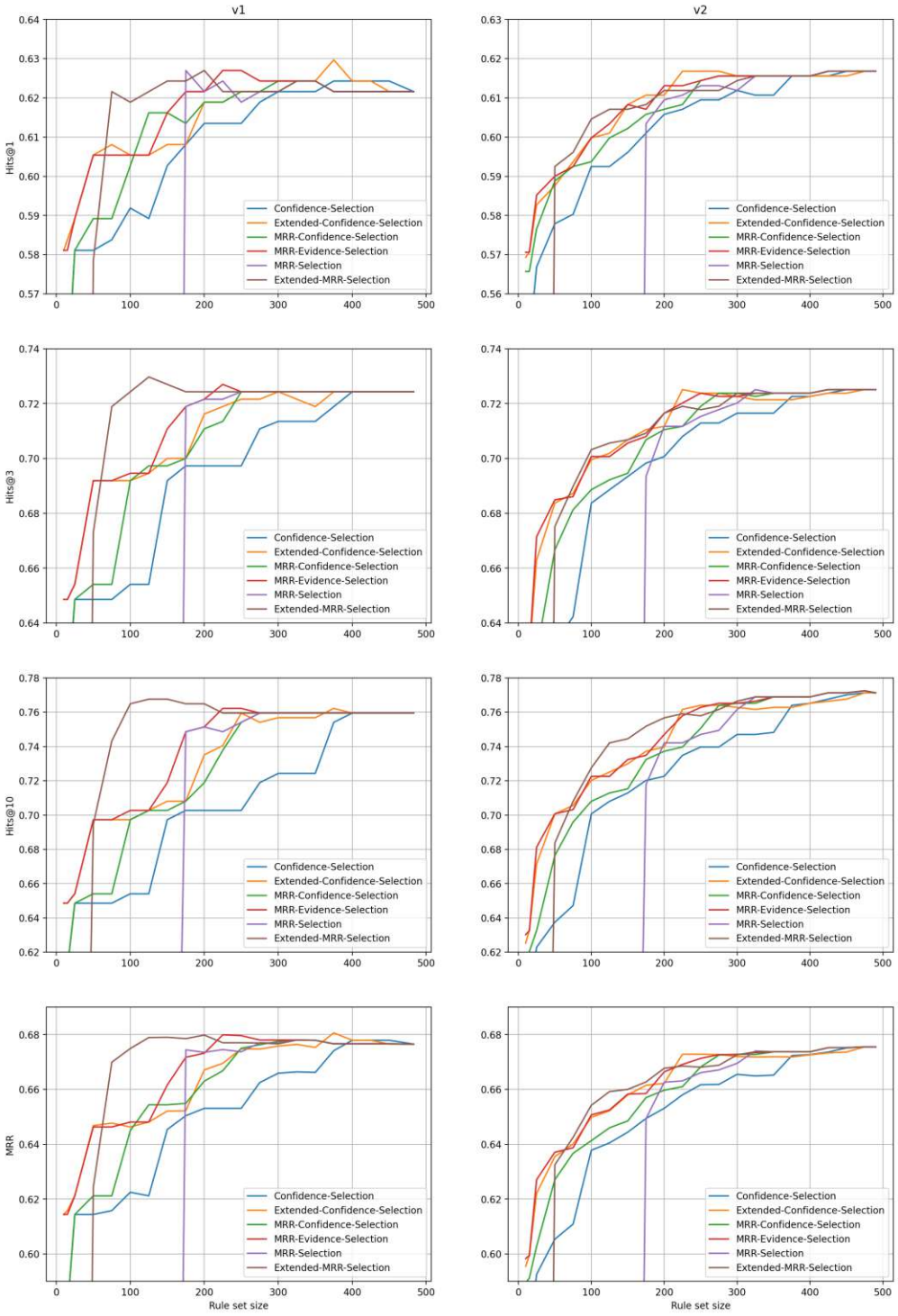

Figure 6.1: Validation scores for the different rule selection methods on InductiveWN18RR v1 and v2.

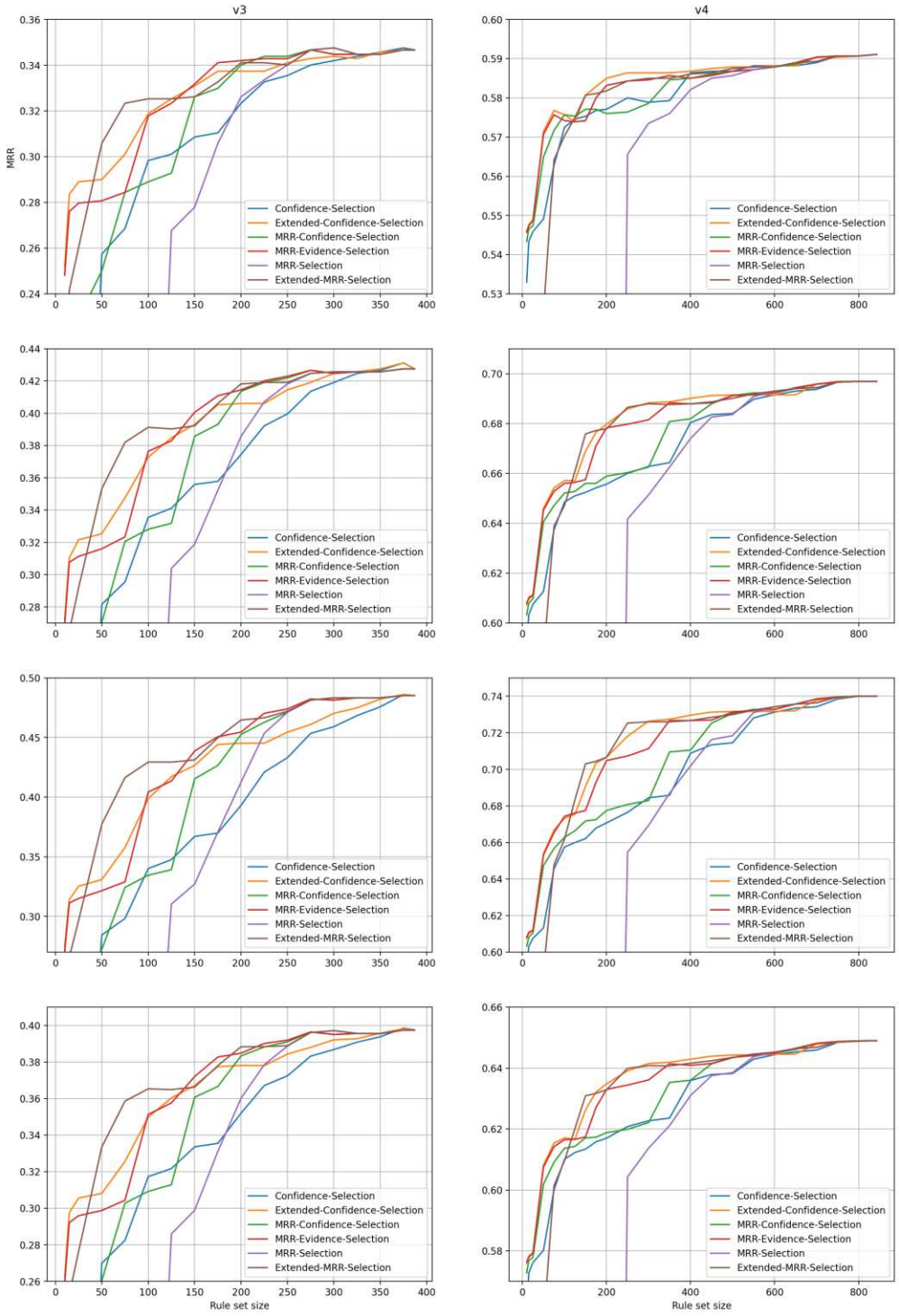

Figure 6.2: Validation scores for the different rule selection methods on InductiveWN18RR v3 and v4.

#Rules	Selection Method	Hits@1	Hits@3	Hits@10	MRR
75	Confidence-Selection	0,5585	0,617	0,617	0,5869
	Extended-Confidence-Selection	**0,6011**	0,6915	0,7207	0,6497
	MRR-Selection	0,0452	0,0851	0,1489	0,0744
	Extended-MRR-Selection	0,5984	**0,7048**	**0,75**	**0,6571**
	MRR-Confidence-Selection	0,5718	0,633	0,633	0,602
	MRR-Evidence-Selection	0,5984	0,6941	0,7181	0,6487
100	Confidence-Selection	0,5691	0,633	0,633	0.6002
	Extended-Confidence-Selection	0,6011	0,6995	0,734	0.6538
	MRR-Selection	0,0612	0,1037	0,1755	0,0958
	Extended-MRR-Selection	**0,6037**	**0,7074**	**0,7766**	**0,6681**
	MRR-Confidence-Selection	**0,6037**	0,6915	0,7128	0,6492
	MRR-Evidence-Selection	0,6011	0,6968	0,7367	0,6533
483	Not applicable	0,6037	0,7128	0,7766	0,6688

Table 6.4: Test scores of the different rule selection methods on InductiveWN18RR v1.

#Rules	Selection Method	Hits@1	Hits@3	Hits@10	MRR
225	Confidence-Selection	0,5884	0,6814	0,7063	0,6384
	Extended-Confidence-Selection	0,6043	0,7029	0,7347	0,6581
	MRR-Selection	0,5975	0,7018	0,7256	0,6525
	Extended-MRR-Selection	0,6043	0,7063	0,7392	0,6601
	MRR-Confidence-Selection	0,6066	0,7063	0,737	0,661
	MRR-Evidence-Selection	**0,6088**	**0,7086**	**0,7415**	**0,6637**
490	Not applicable	0,6168	0,7211	0,7585	0,6747

Table 6.5: Test scores of the different rule selection methods on InductiveWN18RR v2.

#Rules	Selection Method	Hits@1	Hits@3	Hits@10	MRR
275	Confidence-Selection	0,3545	0,4198	0,4595	0,3947
	Extended-Confidence-Selection	0,3562	0,4223	0,4612	0,3969
	MRR-Selection	**0,3587**	**0,4322**	**0,4793**	**0,4041**
	Extended-MRR-Selection	**0,3587**	**0,4322**	**0,4793**	**0,4041**
	MRR-Confidence-Selection	0,3554	0,4273	0,4752	0.4003
	MRR-Evidence-Selection	0,3562	0,4289	0,4769	0,4015
387	Not applicable	0,3628	0,4364	0,4843	0,4082

Table 6.6: Test scores of the different rule selection methods on InductiveWN18RR v3.

#Rules	Selection Method	Hits@1	Hits@3	Hits@10	MRR
500	Confidence-Selection	0,5623	0,6554	0,684	0,6113
	Extended-Confidence-Selection	0,5651	0,6634	0,7019	0,6182
	MRR-Selection	0,5602	0,6536	0,6872	0,6107
	Extended-MRR-Selection	0,5644	0,6624	0,7029	0,6181
	MRR-Confidence-Selection	**0,5654**	**0,6638**	0,7026	**0,6191**
	MRR-Evidence-Selection	0,5647	0,6634	**0,704**	0,6188
650	Confidence-Selection	**0,5665**	0,6672	0,7061	0,6208
	Extended-Confidence-Selection	0,5661	0,6666	0,7057	0,6204
	MRR-Selection	0,5602	0,6536	0,6872	0,6107
	Extended-MRR-Selection	**0,5665**	**0,6679**	**0,7075**	**0,6215**
	MRR-Confidence-Selection	0,5661	0,6666	0,7064	0,6207
	MRR-Evidence-Selection	0,5661	0,6662	0,7061	0,6206
843	Not applicable	0,5665	0,6679	0,7075	0,6215

Table 6.7: Test scores of the different rule selection methods on InductiveWN18RR v4.

6.3 Discussion

In this chapter, we showed how embedding-based rule selection methods can improve inductive knowledge graph completion. Rule selection methods based on the Filtered-MRR performed best in all four datasets. This is in stark contrast to the transductive setting, which we examined in the last chapter and where the MRR-Selection and the Extended-MRR-Selection performed worst. In the future, research will need to show whether embedding-based selection methods generally perform better in the inductive setting than in the transductive setting or if this is a speciality of the WN18RR dataset.

With our rule selection methods, we drastically reduced the rule set sizes in all four datasets while keeping the MRR at acceptable levels. However, the reduction depended on the variant of the dataset. For example, it was possible to reduce the rule set size of v1 by 79,3% while reducing the MRR only by 0,1% compared to the full rule set, while for v4 only a reduction of 22,9% was possible to keep a similar MRR score.

Our results also show we do not know the best rule selection method a priori. Instead, we must find the best rule selection method in the validation set. Taking the last chapter into account, the Extended-MRR-Selection performed best on most tasks - being the clear winner on the v1 dataset, tying with the MRR-Selection on v3, and performing best at the 0,5% target for v4. However, the Extended-Confidence-, MRR-Evidence-, MRR-, and MRR-Confidence-Selection also perform best on at least one task.

While we can not select the best selection method, we can certainly say that the Confidence-Selection method performs worst. Only for the Hits@1 metric can it sporadically, e.g., in the transductive setting and for v4, match the scores of the other selection methods. Therefore, we conclude that both the Filtered-MRR and our extensions are valuable for

selecting good rule sets.

CHAPTER 7

Conclusion

In our work, we studied whether ExpressivE learns logical rules, how it weights those rules and whether we can extract the weighting to improve rule-based knowledge graph completion methods. We showed that logical rules are indeed reflected in ExpressivE's weights and that we can drastically reduce the rule set size of rule-based methods by employing ExpressivE's rule weights as decision criteria to select smaller rule sets - in both the transductive and inductive setting. Thereby, we developed better explainable models, which have only marginally worse test scores than the larger models. In this chapter, we will summarize our main results and look at open questions which should be answered in future research.

7.1 Main Contribution

We can measure whether ExpressivE learns logical rules in two ways: Either we measure whether ExpressivE's weights conform to logical rules or check how ExpressivE ranks triples following from logical rules. While the second option is easier, it is less accurate and more error-prone, as we have small sample sizes and triples following from multiple logical rules. In our thesis, we studied both methods for measuring ExpressivE's rule conformance, and they both show that ExpressivE learns the same logical rules that we can mine with rule miners like AnyBURL.

In Chapter 3, we showed that AnyBURL's U_c rules are reflected within ExpressivE's weights. For this, we developed the new **Rule Conformance**, which is a geometric interpretation of how strongly U_c rules are reflected within the weights of ExpressivE. Our experiments showed that ExpressivE learns the logical rules right at the beginning of the training. However, the logical rules are not immediately applicable in all dimensions. For example, the body of a logical rule can be unsatisfied in several dimensions of ExpressivE. Our experiments show that during ExpressivE's training process, the number of applicable dimensions grows continuously and reaches its peak - with almost all dimensions being

applicable - when the validation scores stop improving. Therefore, we conclude that learning logical rules is important for ExpressivE's performance.

Next, we developed the **Filtered-MRR-Score** in Chapter 4, a measure of how ExpressivE weights AnyBURL's B rules. The Filtered-MRR-Score is particularly interesting for B rules as it is based on how ExpressivE ranks triples and, therefore, needs several samples for each rule to provide valid results. This is the case for B rules, while U_c and U_d rules generally predict less triples as they contain constants. With the Filtered-MRR-Score, we confirmed that ExpressivE prioritizes triples following from high-confidence rules compared to triples following from low-confidence rules. However, we also showed that ExpressivE prioritizes rules differently than AnyBURL, as there is no correlation between AnyBURL's confidence and the Filtered-MRR Score for high-confidence rules.

We extended the Filtered-MRR Score to the MRR-Evidence Score, which unites AnyBURL's confidence, the Filtered-MRR and an estimation of the probability that a rule will be applicable to the test set. As we wanted to use our newly developed scores to achieve better results, or in our case, to achieve smaller rule sets, we deduced the following six rule selection methods from the confidence, the Filtered-MRR Score and the MRR-Evidence Score:

- Confidence-Selection
- Extended-Confidence-Selection
- MRR-Selection
- Extended-MRR-Selection
- MRR-Confidence-Selection
- MRR-Evidence-Selection

In Chapter 5 and 6, we showed that we can drastically reduce the rule set sizes with those selection methods, both in the transductive as well as in the inductive setting, while keeping acceptable test scores. In particular, we showed that we can reduce the rule set size by 22,9% to 79,3% while keeping the MRR at 99,5% compared to the entire rule set size. However, we also showed that we do not know a priori which selection method will perform best. Instead, the best selection method is task-dependent and needs to be selected while training the model.

Our experiments showed that we can create smaller rule sets and, therefore, better explainable models. Additionally, we showed that we can use ExpressivE in the inductive setting, which was previously impossible as ExpressivE needs to learn an embedding for each triple in the training phase. Also, our selection methods provide little overhead. Especially, our extensions, incorporating the probability that rules will be applicable in the test set, are lightweight and provide an easy mechanism to select smaller rule sets in the future.

7.2 Future Work

While our work provides several answers on whether ExpressivE learns logical rules and how we can select better rule sets, it also poses several questions. Here, we will summarize the most important open questions that future research should answer.

In Chapter 3, we introduced the rule conformance for U_c rules. In the future, the rule conformance could be extended to U_d and B rules. We expect that ExpressivE also learns those rule types.

While the rule conformance is specific to ExpressivE, the Filtered-MRR-Score is applicable to all embedding-based methods - as long as they learn logical rules. While ExpressivE is undoubtedly an interesting example for extracting rule weights out of embedding-based methods - as it is designed to capture logical rules during the training phase - it would be interesting to compute the Filtered-MRR score for other embedding-based methods. Thereby, we could show whether selection methods based on the Filtered-MRR score of other models work better or worse, and we could show whether those models learn logical rules.

In Section 5.1, we tried to improve the predictions of AnyBURL by reranking the rules. However, we could not change the predictions by reranking the rules - neither to the positive by reranking via the MRR-Evidence Score nor to the negative by applying random weights. This result is surprising and should be further investigated. We would expect rerankings to influence the outcome, and we would expect a positive influence on our new scores, as those performed well when selecting rule sets. Therefore, the reranking experiments should be repeated on larger datasets to find out whether the rule rank has either no influence or if we can improve the results by incorporating our scores.

When we look at the validation scores for different rule set sizes, we see that the scores do not improve continuously but stepwise. This stepwise improvement is visible in Graphic 6.1 and 6.2, where we have periods with few improvements and sudden jumps in the validation scores. Therefore, we would expect a small set of high-impact rules to exist, far more critical than the rest. Additionally, as the validation scores drop in some cases when more rules are added, we expect that rules with negative impact will exist. Therefore, we would expect that better rule selection methods, which better identify those high-impact rules, could achieve similar or better results with smaller rule sets. While it will be challenging to find a suitable scoring system that identifies all high-impact rules, and while it is computationally infeasible to find the best rule set by brute-forcing all possible combinations, heuristic methods could improve our ability to select better rule sets. For this, we suggest genetic algorithms as it has been shown that they perform well on the task of subset-selection [TFZB08].

The most important and easiest addition to our work is validating our results on other datasets. In our work, we restricted our experiments to the WN18RR dataset as it is a small dataset on which we can train knowledge graph completion models quickly and with relatively little computational power. In the future, extending the experiments to larger

datasets, for example, FB15k-237, would be interesting to show whether ExpressivE generally learns logical rules or whether rule learning occurs only on specific datasets. Also, on larger datasets, the ability to reduce the size of rule sets is of much greater importance, as the needed rule sets are much larger. Therefore, we could significantly improve both the explainability and the runtime of rule-based methods.

Overview of Generative AI Tools Used

To ensure grammatical correctness, the premium version of Grammarly was used. No generative AI tools, neither from Grammarly nor from any other vendor, were used throughout this thesis.

List of Figures

List of Tables

Bibliography

[ABH+20] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:8825–8845, 2020.

[ACLS20] Ralph Abboud, undefinedsmail undefinedlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: a box embedding model for knowledge base completion. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[BAH19] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Tucker: Tensor factorization for knowledge graph completion. pages 5188–5197, 01 2019.

[BGWB14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94:233–259, 2014.

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[DMSR18] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[EW16] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. 09 2016.

[KTG+06] Charles Kemp, Joshua Tenenbaum, Thomas Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. *Cognitive Science*, 21, 01 2006.

[LGS20] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. Fast and exact rule mining with amie 3. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *The Semantic Web*, pages 36–52, Cham, 2020. Springer International Publishing.

[MCB+24] Christian Meilicke, Melisachew Wudage Chekol, Patrick Betz, Manuel Fink, and Heiner Stuckeschmidt. Anytime bottom-up rule learning for large-scale knowledge graph completion. *The VLDB Journal*, 33:131–161, 2024.

[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, IJCAI-19*, pages 3137–3143. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[MGH+24] S. M. Hasan Mahmud, Kah Ong Michael Goh, Md. Faruk Hosen, Dip Nandi, and Watshara Shoombuatong. Deep-wet: a deep learning-based approach for predicting dna-binding proteins using word embedding techniques with weighted features. *Scientific Reports*, 14, 2024.

[MGW+13] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[Mil95] George A. Miller. Wordnet: A lexical database for english. 38(11):39–41, 1995.

[PS22] Aleksandar Pavlović and Emanuel Sallinger. Expressive: A spatio-functional embedding for knowledge graph completion, 06 2022.

[RSH+16] Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II*, page 177–185, Berlin, Heidelberg, 2016. Springer-Verlag.

[SDNT19] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

[Sin12] Amit Singhal. Introducing the knowledge graph: things, not strings, 2012.

[SKB+18] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 593–607. Springer International Publishing, 2018.

[TC15] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In Alexandre Allauzen, Edward Grefenstette, Karl Moritz Hermann, Hugo Larochelle, and Scott Wen-tau Yih, editors, *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

[TDH20] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9448–9457. PMLR, 13–18 Jul 2020.

[TFZB08] Feng Tan, Xuezheng Fu, Yanqing Zhang, and Anu G. Bourgeois. A genetic algorithm-based method for feature subset selection. *Soft Computing*, 120:111–120, 2008.

[Tuc66] Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31, 1966.

[TWR+16] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org, 2016.

[VSNT19] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. 2019.

[WWC+19] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C.-C. Jay Kuo. Evaluating word embedding models: methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8:e19, 2019.

[XYC+18] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990, October-November 2018.

[YtYH+14] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2014.