Universität des Saarlandes
Max-Planck-Institut für Informatik
AG5

An Embedding-based Approach to Rule Learning from Knowledge Graphs

Masterarbeit im Fach Informatik
Master's Thesis in Computer Science
von / by

Vinh Thinh Ho

angefertigt unter der Leitung von / supervised by

Dr. Daria Stepanova

begutachtet von / reviewers

Dr. Daria Stepanova
Prof. Dr. Gerhard Weikum

Saarbrücken, May 2018

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, May 2018 Vinh Thinh Ho

Abstract

Knowledge Graphs (KGs) play an important role in various information systems and have application in many fields such as Semantic Web Search, Question Answering and Information Retrieval. KGs present information in the form of entities and relationships between them. Modern KGs could contain up to millions of entities and billions of facts, and they are usually built using automatic construction methods. As a result, despite the huge size of KGs, a large number of facts between their entities are still missing. That is the reason why we see the importance of the task of Knowledge Graph Completion (a.k.a. Link Prediction), which concerns the prediction of those missing facts.

Rules over a Knowledge Graph capture interpretable patterns in data and various methods for rule learning have been proposed. Since KGs are inherently incomplete, rules can be used to deduce missing facts. Statistical measures for learned rules such as confidence reflect rule quality well when the KG is reasonably complete; however, these measures might be misleading otherwise. So, it is difficult to learn high-quality rules from the KG alone, and scalability dictates that only a small set of candidate rules is generated. Therefore, the ranking and pruning of candidate rules are major problems. To address this issue, we propose a rule learning method that utilizes probabilistic representations of missing facts. In particular, we iteratively extend rules induced from a KG by relying on feedback from a precomputed embedding model over the KG and optionally external information sources including text corpora. The contributions of this thesis are as follows:

- We introduce a framework for rule learning guided by external sources.
- We propose a concrete instantiation of our framework to show how to learn high-quality rules by utilizing feedback from a pretrained embedding model.
- We conducted experiments on real-world KGs that demonstrate the effectiveness of our novel approach with respect to both the quality of the learned rules and fact predictions that they produce.

Acknowledgements

First of all, I would like to give my special thanks to Dr. Daria Stepanova for guiding me to complete my master thesis. She not only gave me a plenty of ideas for my thesis, but also sent me so much energy and enthusiasm when I was in the hardest time. To me, she is an ideal supervisor, I would not have been able to finish this thesis without her help.

I also want to say thanks to Prof. Dr. Gerhard Weikum, Dr. Evgeny Kharlamov and Mohamed H. Gad-Elrab for cooperating with me to successfully publish a paper in the *ISWC'18* conference.

I am also thankful for the support from Dang during my master life. He is not only my friend, my teammate, my senior, but also my brother, who helped me a lot in my career life.

Lastly but most importantly, I want to send a sincere thanks to my grandparents, my parents, my brother, who are always besides me, providing me with love and encouragement to continually push myself to succeed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Rules are widely used to represent relationships and dependencies between data items in datasets and to capture the underlying patterns in data [2, 36]. Applications of rules include health-care [52], telecommunications [27], manufacturing [3], and commerce [39, 20]. In order to facilitate rule construction, a variety of rule learning methods have been developed, see e.g. [11, 23] for an overview. Moreover, various statistical measures such as confidence, actionability, and unexpectedness to evaluate the quality of the learned rules have been proposed.

Rule learning has recently been adapted to the setting of Knowledge Graphs (KGs) [13, 50, 12, 44] where data is represented as a graph of entities interconnected via relations and labeled with classes, or more formally as a set of grounded binary and unary atoms typically referred to as facts. Examples of large-scale KGs include Wikidata [46], Yago [42], NELL [30], and Google's KG [1]. Since many KGs are constructed from semi-structured knowledge, such as Wikipedia, or harvested from the Web with the combination of statistical and linguistic methods, they are inherently incomplete [35, 13].

Rules over KGs are of the form *head* $\leftarrow$ *body*, where *head* is a binary atom and *body* is a conjunction of, possibly negated, binary or unary atoms. When rules are automatically learned, statistical measures like support and confidence are used to assess the quality of rules. Most notably, the confidence of a rule is the fraction of facts predicted by the rule that are indeed true in the KG. However, this is a meaningful measure for rule quality only when the KG is reasonably complete. For rules learned from largely incomplete KGs, confidence and other measures may be misleading, as they do not reflect the patterns in the missing facts. For example, a KG that knows only (or mostly) male CEOs would yield a heavily biased rule $gender(X, male) \leftarrow isCEO(X, Y), isCompany(Y)$, which does not extend to the entirety of valid facts beyond the KG. Therefore, it is crucial that rules can be ranked by a meaningful quality measure, which accounts for KG incompleteness.

1.2 Example

Consider a KG about people's jobs, residence and spouses as well as office locations and headquarters of companies. Suppose that a rule learning method has computed the following two rules:

$$r_1 : livesIn(X, Y) \leftarrow worksFor(X, Z), hasOfficeIn(Z, Y) \tag{1.1}$$
$$r_2 : livesIn(Y, Z) \leftarrow marriedTo(X, Y), livesIn(X, Z) \tag{1.2}$$

The rule r_1 is quite noisy, as a company may have offices in many cities, but employees live and work in only one of them, while the rule r_2 clearly is of higher quality. However, depending on how the KG is populated with instances, the rule r_2 could nevertheless score higher than r_1 in terms of confidence measures. For example, the KG may contain only a specific subset of company offices and only people who work for specific companies. If we knew the complete KG, then the rule r_2 should presumably be ranked higher than r_1.

Suppose we had a perfect oracle for the true and complete KG. Then we could learn even more sophisticated rules such as

$$r_3 : livesIn(X, Y) \leftarrow worksFor(X, Z), hasHeadquartersIn(Z, Y), not\ locatedIn(Y, USA) \tag{1.3}$$

This rule would capture that most people work in the same city as their employers' headquarters, with the USA being an exception (assuming that people there are used to long commutes). This is an example of a rule that contains a negated atom in the rule body (so it is no longer a Horn rule) and has a partially grounded atom with a variable and a constant as its arguments.

1.3 Problem

The problem of KG incompleteness (i.e. KG completion, link prediction) has been tackled by methods that (learn to) predict missing facts for KGs (or actually missing relational edges between existing entities). A prominent class of approaches is statistics-based and includes tensor factorization, e.g. [34] and neural-embedding-based models, e.g. [5, 33]. Intuitively, these approaches turn a KG, possibly augmented with external sources such as text [51, 53], into a probabilistic representation of its entities and relations, known as *embeddings*, and then predict the likelihood of missing facts by reasoning over the embeddings (see, e.g. [47] for a survey).

These kinds of embeddings can complement the given KG and are a potential asset in overcoming the limitations that arise from incomplete KGs. Consider the following gedankenexperiment: we compute embeddings from the KG and external text sources, that can then be used to predict the complete KG that comprises all valid facts. This would seemingly be a perfect starting point for learning rules, without the bias and quality problems of the incomplete KG. However, this scenario is way oversimplified. The embedding-based fact predictions would themselves be very noisy, yielding also many spurious facts. Moreover, the computation of all fact predictions and the induction of all possible rules would come with a big scalability challenge: in practice, we need to restrict ourselves to computing merely small subsets of likely fact predictions and promising rule candidates.

1.4 Thesis Goals and Research Challenges

The goals of this thesis are as follows:

- Design an abstract framework for learning rules from knowledge graphs that also takes into account the support from external sources to improve the rules' quality.
- Create a rule learning model of the above framework by initializing external sources with an embedding model and design a mining algorithm that establish an *"user-in-the-loop"* inspired interaction between the mining process and the embedding model.
- Build a system that implements the designed model.
- Conduct various experiments to evaluate the proposed approach using the implemented system.

To achieve these goals, there are several challenges that we have to overcome:

- The interaction between the rules and the abstract external sources or the specific embedding models in our approach is not clearly defined yet. We need to express them somehow that can preserve the advantages from both rule-based and embedding-based approaches.

- We need to figure out how to make use of the defined interaction in learning rules. In particular, we need to develop an algorithm to extract high-quality rules from KGs. In addition, it is ideal if the extracted rules are of various forms, i.e. they could contain unary/binary predicates, positive/negative atoms.
- The system we build should be efficient in terms of both running time and memory consumption. Relevant optimization techniques should be established to meet these requirements.
- We need to construct benchmarks and define evaluation metrics relevant for the experiments. Moreover, the experiments should be conducted on different datasets.

1.5 Contribution

In this work we propose an approach for rule learning guided by external sources that allows to learn high-quality rules from incomplete KGs. In particular, our method extends rule learning by exploiting probabilistic representations of missing facts computed by embedding models of KGs and possibly other external information sources. We iteratively construct rules over a KG and collect feedback from a precomputed embedding model, through specific queries issued to the model for assessing the quality of (partially constructed) rule candidates. This way, the rule induction loop is interleaved with the guidance from the embeddings, and we avoid scalability problems. Our machinery is also more expressive than many prior works on rule learning from KGs, by allowing non-monotonic rules with negated atoms as well as partially grounded atoms. Within this framework, we devise confidence measures that capture rule quality better than previous techniques and thus improve the ranking of rules.

The salient contributions of our work are as follows.

- We propose a rule learning approach guided by external sources, and show how to learn high-quality rules by utilizing feedback from embedding models.
- We implement our approach into a system prototype RuLES and present extensive experiments on real-world KGs including Freebase, Wikidata and IMDB, that demonstrate the effectiveness of our approach with respect to both the quality of the learned rules and the fact predictions that they produce.
- Our code and data are made available to the research community at `https://github.com/hovinhthinh/RuLES`

1.6 Thesis Structure

The structure of this thesis is as follows. In chapter 2, we present related work, where various existing approaches for solving the KG completion task are briefly recapped. Chapter 3 provides background knowledge and preliminaries, including: knowledge graph, (non-)monotonic logic program, relational association rule learning and embedding model. In chapter 4, we present our approach in detail to tackle the link prediction task. Chapter 5 describes the usage of our developed system. Conducted experiments, results and their analysis are demonstrated in chapter 6. Finally, chapter 7 concludes this thesis and outlines some relevant future research directions.

Chapter 2

Related Work

There are several existing approaches for solving the problem of KG completion. In this chapter, we give an overview about relevant works from these approaches.

2.1 Knowledge Graph Latent Feature Models

Aiming at the problem of knowledge graph completion, Latent Feature Models (or Embedding Models) have gained attention from research community in the last few years. These models encode each entity and relation of the knowledge graph as a low-dimensional vector (or embedding). Each unknown fact of the knowledge graph is then given a score computed from a mathematical formula based on the embeddings of its subject, predicate and object. These scores reflect the trustfulness of those facts. The process of learning these embedding vectors is a machine learning problem. In particular, Stochastic Gradient Descent is used to train the embeddings to minimize the defined loss function, which indicates the total plausibility of existing facts of the knowledge graph.

A large number of models have been introduced in the literature to learn the representation feature vectors of entities and relations of the knowledge graphs. Some possibilities are RESCAL [34], Neural Tensor Network (NTN) [41], Structured Embedding (SE) [6], HolE [33], ComplEx [45] and TransE [5]. Table 2.1 recaps some models having been introduced in the literature and their memory complexity.

2.1.1 TransE - The Translating Embedding Model

The Translating Embedding Model (TransE [5]) is among the most prominent state-of-the-art embedding models that have been proposed to tackle the link prediction task.

MODEL	NUMBER OF PARAMETERS
RESCAL[34]	$O(n_ek + n_rk^2)$
SE[6]	$O(n_ek + 2n_rk^2)$
SME(Linear)[16]	$O(n_ek + n_rk + 4k^2)$
SME(Bilinear)[16]	$O(n_ek + n_rk + 2k^3)$
LFM [21]	$O(n_ek + n_rk + 10k^2)$
TransE [5]	$O(n_ek + n_rk)$
HolE[33]	$O(n_ek + n_rk)$
SSP [53]	$O(2n_ek + n_rk)$

Table 2.1: Numbers of parameters for some embedding models (adapt from [5]). n_e and n_r are the number of entities and binary predicates; k is the embeddings dimension.

Its main advantages are efficiency and simplicity. TransE encodes each entity as a vector $e \in \mathbb{R}^k$ and each relation as a vector $r \in \mathbb{R}^k$. The motivation behind this model comes from the idea that triples of the knowledge graph could be represented as the translation of embeddings over the latent vector space. In particular, the score of each given triple $\langle s, p, o \rangle$ is computed using the following formula:

$$score_{\langle s,p,o \rangle} = -d(e_s + r_p, e_o) = -||e_s + r_p - e_o||_{L_{1,2}}$$

in which, the distance function d is defined as the $L_{1,2}$-norm function. To learn the embedding representation of entities and relations, TransE model is then trained with the margin-based pairwise loss training [5] to maximize the margin between existing triples of the knowledge graph and some random false triples. This training technique is well-known and is used not only in TransE, but also in many other embedding models. The number of parameters in TransE model is linearly dependent on the number of entities and relations, which witnesses its effectiveness. TransE runs fast and could be potentially applied on large knowledge graphs.

The TransE model is the baseline for a large number of other embedding models. In [28], the authors improve TransE by leveraging additional schema knowledge. In [22], TransE model is enhanced with temporal information about facts of the KG. Meanwhile, STransE [31] is introduced based on the combination of TransE and Structured Embedding model (SE) [6]. TransH [49] improves TransE by projecting the embeddings of subjects and objects into a hyperplane before translating. Additionaly, TransR [25] continues enhancing the performance of TransH by allowing the feature vectors of entities and relations to have different dimensions.

2.1.2 HolE - The Holographic Embedding Model

Another state-of-the-art embedding model for KG completion task that has been recently proposed is the Holographic Embedding Model (HolE [33]). In terms of memory complexity, similarly to TransE model, HolE also stores a vector $e \in \mathbb{R}^k$ for each entity and a vector $r \in \mathbb{R}^k$ for each relation. However, its scoring function is totally different. The HolE model is quite closed to holographic models of associative memory [33], where it takes advantage of *circular correlation* to represent the likelihood scores of facts. In particular, HolE defines the scoring function as follows:

$$score_{\langle s,p,o\rangle} = \sigma(r_p^\top cxcorr(e_s, e_o))$$

where σ is the sigmoid function and $cxcorr(e_s, e_o)$ denotes the *circular correlation* of representation vectors of the subject and the object, which could be computed efficiently by using Fast Fourier Transform (FFT) technique. By employing *circular correlation* operator, the model can capture rich interactions over entities and relations of the KG, and at the same time be fast, easy to compute, and might potentially scale to large KGs.

2.1.3 SSP - The Semantic Space Projection Embedding Model

Unlike TransE model and HolE model, which work only on the given knowledge graph, the Semantic Space Projection (SSP [53]) is a state-of-the-art model among the class of embedding models that support external resources. In particular, apart from the KG itself, each entity of the KG might be given a description text that will also be considered during the training process of the SSP model. The model builds the connection between the symbolic triples of the KG and the external entities description texts, and then facts' likelihood is extracted based on the two sources of information.

More specifically, SSP model stores two types of embeddings. First, similarly to TransE and HolE, the SSP model also encodes each entity and relation of the KG with a vector $e \in \mathbb{R}^k$ and $r \in \mathbb{R}^k$, correspondingly, which are the embeddings generated from KG's triples. Second, each entity of KG is also given a semantic vector $t \in \mathbb{R}^k$ computed from the description texts. Finally, SSP defines the likelihood function as follows:

$$score_{\langle s,p,o\rangle} = \lambda||l - c^\top lc||_2^2 - ||l||_2^2$$

where hyper-parameter λ is the balance factor between the two parts, l is the loss vector and c is the semantic composition vector, given by:

$$l \doteq e_s + r_p - e_o \quad \text{and} \quad c \doteq \frac{t_s + t_o}{||t_s + t_o||_2^2}$$

The number of parameters is linearly dependent on the number of entities and relations of the KG, together with the ability to integrate external resources, SSP is a very efficient embedding model in terms of running time, memory complexity as well as prediction quality.

2.2 Relational Rule Learning

Alternative techniques for addressing the KG link prediction task include rule-based approaches. Rule-based systems extract logical rules from the knowledge graph via mining methods and then use these rules to predict missing links. Traditionally learning rules from data has been faced in the context of Inductive Logic Programming (ILP) [38, 37, 9], where given background knowledge and a set of positive and negative examples over target predicates, the task is to learn rules for predicting the existence of missing facts from these predicates. In our setting, the negative examples are not available, which prohibits application of the off-the-shelf ILP rule learners.

To overcome this obstacle, a common approach is to treat the problem as an unsupervised relational learning task, and apply algorithms for relational association rule mining such as [17, 13, 50, 7, 44]. In AMIE [14, 13], rules are constructed by adding atoms one after another in a top-down fashion through the usage of mining operators. While AMIE mines 1 rule at a time, the other state-of-the-art system called RDF2Rules [50] parallelizes this process through the extraction of the so-called Frequent Predicate Cycles (i.e. cycles of predicates with specific directions that appear frequently in the KG), which are then post-processed to cast into rules. Recently proposed algorithm Ontological Pathfinding (OP) [7] exploits various optimization techniques including parallelism, pruning and partitioning to extract rules from KGs. While above systems focus on mining only Horn rules (i.e. rules without negated atoms), RUMIS [44] supports also non-monotonic rules with negated parts in their body. All of these systems use standard metrics to assess the quality of the rules such as support, confidence and conviction.

2.3 Combining Rule-based Approach with Embedding-based Approach

So far we have discussed two different approaches for tackling the link prediction problem. Since logical rules are easily interpretable, rule-based methods could achieve a high accuracy. However, they process only small subsets of the knowledge graph and therefore are unable to capture global relational patterns inside the knowledge graph. In contrast, while embedding-based methods can capture some unseen characteristics of entities and relations, the results they produce are hard to interpret.

A natural idea of combining the embedding-based and rule-based methods has led to a number of proposals. In [48], existing embedding models (in particular RESCAL and TransE) are boosted by imposing some physical and logical rules through an Integer Linear Program (ILP). Introduced rules include:

- *noisy observation*: reflects the fact that some existing triples of the knowledge graph could be false.
- *argument type expectation*: puts a constraint on the type of subject/object connected by each specific predicate.
- *at-most-one restraint*: restricts the cardinality of head and tail of each relation.
- *simple implication*: ensures the correctness of relation implication. For example: *isMale* implies *isPerson*.

The work [18] proposes the embedding model KALE, which also takes TransE model into account, and then enhances it with logical rules. KALE uses t-norm fuzzy logics to modelize rules. Apart from minimizing the total plausibility of existing triples as in TransE model, KALE also minimizes the plausibility of instances of rules in the knowledge graph. In addition, there are other works focusing on enhancing embedding models with (possibly learned) rules and constraints, e.g. [19, 40].

A natural way to improve the quality of rules learned from incomplete KGs that has recently gained attention [54, 55] is to completely rely on embedding models during the rule learning process. However, the existing methods are purely statistics-based, i.e., they reduce the rule learning problem to algebraic operations on neural-embedding-based representations of a given KG. Indeed, [54] constructs rules by modeling relation composition as multiplication or addition of two relation embeddings. The authors of [55] propose a differentiable system for learning models defined by sets of first-order rules that exploits a connection between inference and sparse matrix multiplication [8]. Typically, existing approaches pose strong restrictions on the patterns encoded in rules that they can learn, and therefore these methods do not allow to learn rules that reflect natural and interesting patterns.

Chapter 3

Preliminaries

In this chapter we first give some necessary preliminaries. These background knowledge and notations will then be exploited to introduce our framework, describe our system implementation and present the conducted experiments later in this thesis.

3.1 Knowledge Graphs

On the Web, knowledge graphs (KG) are often encoded using the RDF data model [24], which represents the content of the graph with a collection of triples of the form $\langle \mathit{subject\ predicate\ object} \rangle$, reflecting positive binary facts (e.g., $\langle \mathit{bob}, \mathit{livesIn}, \mathit{rome} \rangle$) and unary facts (where $\mathit{predicate} = \mathit{type}$ e.g., $\langle \mathit{bob}, \mathit{type}, \mathit{scientist} \rangle$) about the world. KGs are naturally treated under the Open World Assumption (OWA), where missing facts are assumed to be unknown rather than false as in the Closed World Assumption (CWA).

We consider KG $\mathcal{G}$ defined over the signature $\Sigma_{\mathcal{G}} = \langle \mathcal{R}_{\mathcal{G}}, \mathcal{C}_{\mathcal{G}} \rangle$, where $\mathcal{R}_{\mathcal{G}}$ and $\mathcal{C}_{\mathcal{G}}$ are sets of predicates (aka. relations) and constants (aka entities) respectively. Relying on [10] we define the gap between the ideal graph $\mathcal{G}^i$ (a graph containing all possible correct facts with constant and relation from $\Sigma_{\mathcal{G}}$) and the available graph $\mathcal{G}$.

Definition 3.1 (Incomplete data source)**.** An incomplete data source is a pair $G = (\mathcal{G}, \mathcal{G}^i)$ of two KGs, where $\mathcal{G} \subseteq \mathcal{G}^i$ and $\Sigma_{\mathcal{G}} = \Sigma_{\mathcal{G}^i}$.

3.2 Non-monotonic Logic Programs

In this work, we rely on the standard definitions of logic programs [26]. In short, a *(non-monotonic) logic program* P is a set of rules r of the form:

$$H \leftarrow B, not\ E \tag{3.1}$$

where H is a standard first-order atom of the form $a(\vec{X})$ known as the rule head and denoted as $head(r)$; $B, not\ E$ is the rule body and denoted as $body(r)$, in which B is a conjunction of positive atoms of the form $b_1(\vec{Y_1}), \ldots, b_k(\vec{Y_k})$ to which we refer as $body^+(r)$. Moreover, $not\ E$, to which we refer as $body^-(r)$, with slight abuse of notation, denotes the conjunction of atoms $not\ \ b_{k+1}(\vec{Y_{k+1}}), \ldots, not\ \ b_n(\vec{Y_n})$, where not is called *negation as failure (NAF)* or *default negation*. Here, $\vec{X}, \vec{Y_1}, \ldots, \vec{Y_n}$ are tuples of either constants or variables whose length corresponds to the arity of the predicates $a, b_1, \ldots, b_n$ respectively. Moreover, the rule r is called Horn, if all head variables appear in the body, and the negated part E is empty. The signature of P is given as $\Sigma_P = \langle \mathbf{P}, \mathbf{C} \rangle$, where $\mathbf{P}$ and $\mathbf{C}$ are sets of predicates and constants occurring in P respectively.

We consider answer set semantics [15] in this work. A logic program P is called *ground* if no variable appear in any rule r of P. A ground instantiation $Gr(P)$ of a non-ground logic program P is obtained by replacing its variables by constants of $\mathbf{C}$ in all possible ways. The *Herbrand Universe* $HU(P)$ of a program P is the set of all constants appearing in P. The *Herbrand Base* $HB(P)$ is the set of all possible ground atoms created from predicates in $\mathbf{P}$ and constants in $\mathbf{C}$. An *Herbrand Interpretation* of P is any subset of $HB(P)$. An interpretation I is a *Model* of a program P iff it satisfies every rule r of P, i.e., for every substitution of variables of r with constants in $\mathbf{C}$, if $I \models body(r)$ then we also have $I \models head(r)$. A model I is called minimal if there does not exist any model $I' \subset I$. The set of all subset-inclusion minimal models of P is denoted by $MM(P)$.

An interpretation I of P called an *Answer Set* (or *Stable Model*) of P iff $I \in MM(P^I)$, where P^I is the Gelfond-Lifschitz (GL) reduct of P constructed from $Gr(P)$ by excluding (i) any rule r that $I \not\models body^-(r)$ and (ii) negated part from the remaining rules. The set consisting of all answer sets of P is denoted as $AS(P)$.

Example 3.1. *Consider the following logic program P:*

$$P = \left\{ \begin{array}{l} \textit{(1) } worksFor(Alice, ITech);\ \textit{(2) } hasHeadquartersIn(ITech, Paris); \\ \textit{(3) } livesIn(X,Y) \leftarrow worksFor(X,Z), hasHeadquartersIn(Z,Y), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad not\ locatedIn(Y, USA); \end{array} \right\}$$

We achieve $Gr(P)$ by replacing X, Y, Z by $Alice, Paris, ITech$, correspondingly. With $I = \{worksFor(Alice, ITech), hasHeadquartersIn(ITech, Paris), livesIn(Alice, Paris)\}$, the Gelfond-Lifschitz reduct P^I consists of facts (1), (2) and the rule $livesIn(Alice, Paris) \leftarrow worksFor(Alice, ITech), hasHeadquartersIn(ITech, Paris)$. Because I is a minimal model of P^I, we conclude that I is also an answer set of P. $\square$

3.3 Relational Association Rule Learning

Association rule learning concerns the discovery of frequent patterns in a data set and the subsequent transformation of these patterns into rules. A *conjunctive query* over $\mathcal{G}$ is of the form $Q(\vec{X}) \text{ :- } p_1(\vec{X_1}), \ldots, p_m(\vec{X_m})$, where $\vec{X_i}$ are binary or unary vectors of variables and/or constants. Its right-hand side (i.e., body) is a finite set of atomic formulas over $\Sigma_{\mathcal{G}}$, while the left-hand side (i.e., head) is a tuple of variables occurring in the body and/or constants. The *answer* of Q on $\mathcal{G}$ is the set $Q(\mathcal{G}) = \{\nu(\vec{X}) \mid \nu$ is a function from variables to $\mathcal{C}$ and $\forall i : p_i(\nu(\vec{X_i})) \in \mathcal{G}\}$. The *support* of Q in $\mathcal{G}$ is the number of distinct tuples in the answer of Q on $\mathcal{G}$.

Example 3.2. *The support of the following query:*

$$Q(X, Y, Z) \text{ :- } worksFor(X, Y), hasOfficeIn(Y, Z)$$

over the KG $\mathcal{G}$ in Fig. 3.1 asking for people, their companies and the companies' locations is 12. $\square$

An *association rule* is of the form $Q_1 \Rightarrow Q_2$, such that Q_1 and Q_2 are both conjunctive queries and $Q_1 \subseteq Q_2$, i.e., $Q_1(\mathcal{G}') \subseteq Q_2(\mathcal{G}')$ for any possible KG $\mathcal{G}'$. In this work we exploit association rules for reasoning purposes, and thus (with some abuse of notation) treat them as logical rules, i.e., for $Q_1 \Rightarrow Q_2$ we write $Q_2 \backslash Q_1 \leftarrow Q_1$, where $Q_2 \backslash Q_1$ refers to the set difference between Q_2 and Q_1 seen as sets of atoms.

Example 3.3. *Consider the query Q from Example 3.2 and also the following query:*

$$Q'(X, Y, Z) \text{ :- } worksFor(X, Y), hasOfficeIn(Y, Z), livesIn(X, Z)$$

the association rule $Q \Rightarrow Q'$ can be written as the following logical rule:

$$livesIn(X, Z) \leftarrow worksFor(X, Y), hasOfficeIn(Y, Z)$$

$\square$

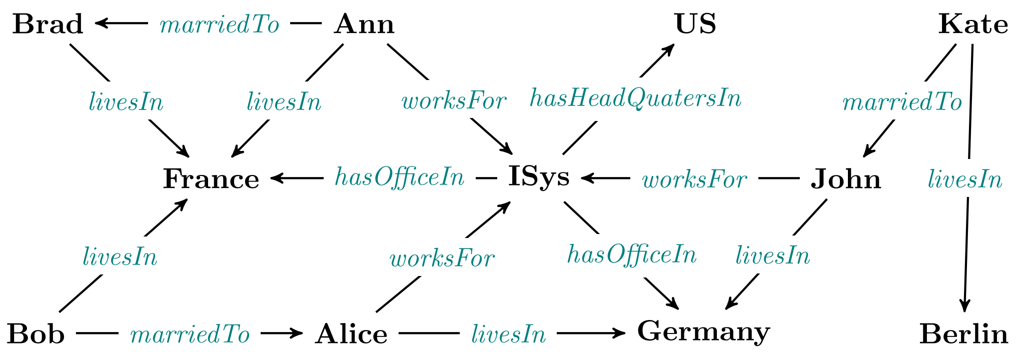


FIGURE 3.1: An example knowledge graph.

3.3.1 Rule Types

Below we recap various rule types.

Definition 3.2 (Connected rule)**.** A rule is *connected* if any two of its variables are connected through a series of atoms, where two atoms are connected if they share a variable or constant.

Example 3.4. *The rule below is connected:*

$$isMusician(X) \leftarrow playsForBand(X, Y)$$

*because variables X and Y are connected through the atom playsForBand(X, Y). In contrast, the following rule is **not** connected:*

$$speaks(X, English) \leftarrow livesIn(X, UK), worksIn(Y, US)$$

This rule seems to be meaningless, since there is no connection between X and Y. If we remove the last atom worksIn(Y, US), then the rule becomes connected. □

Definition 3.3 (Closed rule)**.** A *closed* rule is a connected rule, whose every variable appears at least twice.

Example 3.5. *Rules r_1 and r_2 from Equations 1.1 and 1.2 are closed. An example of a non-closed rule is:*

$$musician(X) \leftarrow playsInstrument(X, Y)$$

□

Definition 3.4 (Recursive rule)**.** A rule is called *recursive* if its body contains the head predicate, since. Applying a recursive rule on the knowledge graph many times might predict new more facts.

Example 3.6. *The rule r_2 is a recursive rule. In contrast, r_1 is* ***not*** *recursive, because the head predicate "livesIn" does not appear in the rule body.* □

Definition 3.5 (Safe rule)**.** A rule is *safe* if every variable in its negated part appears in the positive part of the rule.

Example 3.7. *The rule r_3 in Equation 1.3 is safe. In contrast, the following rule is unsafe:*

$$livesIn(Y, Z) \leftarrow marriedTo(X, Y), livesIn(X, Z), not\ doResearchAt(Y, T)$$

because variable T does not appear in the positive part of the rule. □

3.3.2 Rule Statistics

To quantify the quality of association rules on the knowledge graph, various rule metrics have been proposed. We list some prominent metrics below. Given a KG $\mathcal{G}$, a rule $r : H \leftarrow B, not\ E$, where $H = h(X, Y)$ and B, E involve variables from $\vec{Z} \supseteq \{X, Y\}$, the following measures can be computed:
- **Rule support** (or *support*): indicates the absolute number of instantiations of a rule that are true in the knowledge graph.

$$supp(r, \mathcal{G}) = \#(X, Y) : H \in \mathcal{G}, \exists \vec{Z} : B \in \mathcal{G}, E \notin \mathcal{G} \tag{3.2}$$

where $\#\beta : \mathcal{B}$ denotes the number of β that fulfill the condition $\mathcal{B}$.
- **Body support**: indicates the absolute number of instantiations of rule's body in the knowledge graph.

$$b\text{-}supp(r, \mathcal{G}) = \#(X, Y) : \exists \vec{Z} : B \in \mathcal{G}, E \notin \mathcal{G} \tag{3.3}$$

- **Head support**: indicates the absolute number of instantiations of rule's head in the knowledge graph. In other words, this is the total number of facts in the KG over the head predicate h.

$$h\text{-}supp(r, \mathcal{G}) = \#(X, Y) : H \in \mathcal{G} \tag{3.4}$$

Example 3.8. *Consider the two rules r_1, r_2 from Equations 1.1 and 1.2 and the KG $\mathcal{G}$ from Fig. 3.1. For the rule r_1, we have $supp(r_1, \mathcal{G}) = 3$ and $b\text{-}supp(r_1, \mathcal{G}) = 6$. Similarly, for r_2, we have $supp(r_2, \mathcal{G}) = 1$ and $b\text{-}supp(r_2, \mathcal{G}) = 3$. Moreover, $h\text{-}supp(r_1, \mathcal{G}) = h\text{-}supp(r_2, \mathcal{G}) = 6$, since there are totally 6 facts in $\mathcal{G}$ with the head predicate "livesIn".* □

- **Head coverage**: While support gives us the absolute number of correct rule instantiations, it does not tell us how much the rule affects the knowledge graph, since it does not take into account the size of the knowledge graph. Head coverage fixes this issue by reflecting the ratio of the rule support to the head support.

$$hc(r, \mathcal{G}) = \frac{supp(r, \mathcal{G})}{h\text{-}supp(r, \mathcal{G})} \tag{3.5}$$

Example 3.9. *We have $hc(r_1, \mathcal{G}) = \frac{3}{6}$ and $hc(r_2, \mathcal{G}) = \frac{1}{6}$.* □

- **Standard Confidence** (or *confidence*): treats the knowledge graph under the Closed World Assumption (CWA). The standard confidence returns a number between $[0, 1]$, which corresponds to the ratio of the rule support to the body support. Formally:

$$conf(r, \mathcal{G}) = \frac{supp(r, \mathcal{G})}{b\text{-}supp(r, \mathcal{G})} \tag{3.6}$$

Example 3.10. *We have $conf(r_1, \mathcal{G}) = \frac{3}{6}$ and $conf(r_2, \mathcal{G}) = \frac{1}{3}$. Moreover, for the following rule with negation:*

$$r_4 : livesIn(Y, Z) \leftarrow marriedTo(X, Y), livesIn(X, Z), not\ researcher(X)$$

states that married people live together unless one is a researcher, and $\mathcal{G}' = \mathcal{G} \cup \{researcher(Bob)\}$, we have $conf(r_4, \mathcal{G}') = \frac{1}{2}$. □

- **PCA Confidence** [13]: is based on the Partial Closed-world Assumption (PCA), which assumes that the data of the knowledge graph is added in batches. More specifically, for each subject s and relation p, if there exists some object o' such that $p(s, o') \in \mathcal{G}$, then all possible true triples $p(s, o)$ are assumed to be presented in $\mathcal{G}$. The *PCA confidence* is defined as follows:

$$conf_{pca}(r, \mathcal{G}) = \frac{supp(r, \mathcal{G})}{\#(X, Y) : \exists \vec{Z} : B \in \mathcal{G}, E \notin \mathcal{G} \wedge \exists Y' : h(X, Y') \in \mathcal{G}} \tag{3.7}$$

Similar to the *standard confidence*, the *PCA confidence* also gives us a value between $[0, 1]$.

Example 3.11. *We have $conf_{pca}(r_1, \mathcal{G}) = \frac{3}{6}$, $conf_{pca}(r_2, \mathcal{G}) = \frac{1}{3}$ and $conf_{pca}(r_4, \mathcal{G}') = \frac{1}{2}$. If Alice was not known to live in Germany in $\mathcal{G}$, then $conf_{pca}(r_2, \mathcal{G}) = \frac{1}{2}$.* □

- **Conviction** [44]: is shown to correlate with the rule predictive power [4]. *Conviction* is defined as follows:

$$conv(r, \mathcal{G}) = \frac{1 - rel\text{-}supp(h, \mathcal{G})}{1 - conf(r, \mathcal{G})} \tag{3.8}$$

where *rel-supp*(h) is the relative support of the head, which is measured by:

$$rel\text{-}supp(h,\mathcal{G}) = \frac{h\text{-}supp(r,\mathcal{G})}{(\#X : \exists Y' : h(X,Y') \in \mathcal{G}) \times (\#Y : \exists X' : h(X',Y) \in \mathcal{G})} \tag{3.9}$$

Example 3.12. *We have* $rel\text{-}supp(livesIn,\mathcal{G}) = \frac{6}{6\times 3} = \frac{1}{3}$*. Thus,* $conv(r_1,\mathcal{G}) = \frac{1-\frac{1}{3}}{1-\frac{3}{6}} = \frac{4}{3}$ *and* $conv(r_2,\mathcal{G}) = \frac{1-\frac{1}{3}}{1-\frac{1}{3}} = 1$. □

Unlike standard confidence and PCA confidence, conviction gives us a value in $[0,\infty]$.

3.3.3 Rule-based KG Completion

Given a rule r and a KG $\mathcal{G}$ the application of r on $\mathcal{G}$ results in a rule-based graph completion defined relying on the answer set semantics as recapped in Section 3.2.

Definition 3.6 (Rule-based KG completion)**.** Let $\mathcal{G}$ be a KG over the signature $\Sigma_{\mathcal{G}} = \langle \mathbf{R}, \mathcal{C} \rangle$ and let r be a rule mined from $\mathcal{G}$, i.e. a rule over $\Sigma_{\mathcal{G}}$. Then the *completion of* $\mathcal{G}$ using r is a graph $\mathcal{G}_r$ constructed from any answer set of $r \cup \mathcal{G}$.

Example 3.13. *The application of* r_1 *and* r_2 *on* $\mathcal{G}$ *from Figure 3.1 results respectively in:*
- $\mathcal{G}_{r_1} = \mathcal{G} \cup \{$*livesIn(Ann,Germany), livesIn(John,France), livesIn(Alice,France)*$\}$
- $\mathcal{G}_{r_2} = \mathcal{G} \cup \{$*livesIn(Alice,France), livesIn(John, Berlin)*$\}$. □

Note that $\mathcal{G}^i$ is the perfect completion of $\mathcal{G}$, i.e., it is supposed to contain all correct facts with entities and relations from $\Sigma_{\mathcal{G}}$ that hold in the current state of the world. The goal of the rule-based KG completion is to extract from $\mathcal{G}$ a set of rules $\mathcal{R}$ such that $\mathcal{G}_{\mathcal{R}} = \cup_{r\in\mathcal{R}}\mathcal{G}_r$ is as close to $\mathcal{G}^i$ as possible.

3.4 Link Prediction Embedding Models

Recently a vast amount of latent-based approaches for statistical relational learning have been proposed (see [32] for overview), which address the KG completion problem by reducing it to a representation learning task, where the main goal is to represent entities and relations of $\mathcal{G}$ in a low-dimensional, say d-dimensional, vector (aka embedding) and relying on these representation estimate the likelihood (not necessary probability) of the potentially missing facts. Some models rely exclusively on the triples in the available graph and estimate the likelihood of the missing ones [5, 33], while others may utilize external resources (*e.g.* text) [53].

Typical embedding models define a scoring function $\xi : \mathcal{R}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}} \rightarrow \mathbb{R}$ that given an $\langle s, p, o \rangle$ triple computes a numerical value reflecting its likelihood. The higher score is, the more plausible the triple is. While in this work we experiment with several selected KG embedding models, our general rule learning approach can conceptually exploit any model. Some prominent embedding models have been recapped in related work of Chapter 2.

Chapter 4

Rule Learning Guided by External Sources

In this chapter, we introduce our framework for rule learning guided by external sources, discuss challenges associated with it, and finally propose its concrete instantiation with embedding models.

4.1 Problem Statement and Proposal of General Solution

Let $\mathcal{G}$ be a KG over the signature $\Sigma_{\mathcal{G}} = \langle \mathcal{R}_{\mathcal{G}}, \mathcal{C}_{\mathcal{G}} \rangle$. A *probabilistic KG* $\mathcal{P}$ is a pair $\mathcal{P} = (\mathcal{G}, f)$ where $f : \mathcal{R}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}} \to [0, 1]$ is a probability function over the facts over $\Sigma_{\mathcal{G}}$ such that for each atom $a \in \mathcal{G}$ it holds that $f(a) = 1$.

The goal of our work is to learn rules that not only describe the available graph $\mathcal{G}$ well, but also predict highly probable facts based on the function f. The key questions now are how to define the quality of a given rule r based on $\mathcal{P}$ and how to exploit this quality during rule learning for pruning out not promising rules.

A quality measure μ for rules over probabilistic KGs is a function $\mu : (r, \mathcal{P}) \mapsto \alpha$, where $\alpha \in [0, 1]$. In order to measure the quality μ of r over $\mathcal{P}$ we propose:

- to measure the quality μ_1 of r over $\mathcal{G}$, where $\mu_1 : (r, \mathcal{G}) \mapsto \alpha \in [0, 1]$,
- to measure the quality μ_2 of $\mathcal{G}_r$ by relying on $\mathcal{P}_r = (\mathcal{G}_r, f)$, where $\mu_2 \colon (\mathcal{G}', (\mathcal{G}, f)) \mapsto \alpha \in [0, 1]$ for $\mathcal{G}' \supseteq \mathcal{G}$ is the quality of extensions $\mathcal{G}'$ of $\mathcal{G}$ over $\Sigma_{\mathcal{G}}$ given f, and
- to combine the result as the weighted sum.

That is, we define our hybrid rule quality function $\mu(r, \mathcal{P})$ as follows:

$$\mu(r, \mathcal{P}) = (1 - \lambda) \times \mu_1(r, \mathcal{G}) + \lambda \times \mu_2(\mathcal{G}_r, \mathcal{P}). \tag{4.1}$$

In this formula μ_1 can be any classical quality measure of rules over complete graphs. Intuitively, $\mu_2(\mathcal{G}_r, \mathcal{P})$ is the quality of $\mathcal{G}_r$ wrt f that allows us to capture the information about facts missing in $\mathcal{G}$ that are relevant for r. The weighting factor λ, we call it *embedding weight*, allows one to choose whether to rely more on the classical measure μ_1 or on the measure μ_2 of the quality of the facts that are predicted by r over $\mathcal{G}$.

4.1.1 Challenges

There are several challenges that one has to face when realizing our approach:

- First, given an incomplete KG $\mathcal{G}$, one has to define f such that $(\mathcal{G}, f)$ satisfies the expectations, i.e., reflects well the probabilities of missing facts.
- Second, one has to define μ_1 and μ_2 that also satisfy the expectations and admit efficient implementation.
- Third, the adaptation of existing rule learning approaches to account for the probabilistic function f without the loss of scalability is not trivial. Indeed, materializing f by augmenting $\mathcal{G}$ with all possible probabilistic facts over $\Sigma_{\mathcal{G}}$ and subsequently applying standard rule learning methods on the obtained graph is not practical. Storing such potentially enormous augmented graph where many probabilistic facts are irrelevant for the extraction of meaningful rules might be simply infeasible.

4.2 Realization of General Solution

We now describe how we addressed the above stated challenges in this thesis. In Section 4.2 we present concrete realizations of f, μ_1 and μ_2, and in Section 4.3 we discuss how we implemented them and adapted within an end-to-end rule learning system.

4.2.1 Realization of the Probabilistic Function f

We propose to define f by relying on embeddings of KGs. Embedding models can be used to estimate the likelihood of potentially missing binary atoms using a scoring function $\xi : \mathcal{R}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}} \to \mathbb{R}$. Examples of concrete scoring functions can be found, e.g., in [47].

Since embeddings per se are not in the focus of this thesis, we will not give further details on them and refer the reader to [47] for an overview. Note that our framework is not dependent on a concrete embedding model. What is important for us is that embeddings can be used to construct probabilistic representations [33] of atoms missing in KGs and we use this to define f.

Consider an auxiliary definition. Given a KG $\mathcal{G}$, and an atom $a = p(s, o)$, the set $\mathcal{G}_s$ consists of a and all atoms a' that are obtained from a by replacing s with a constant from $\Sigma_{\mathcal{G}}$, except for those that are already in $\mathcal{G}$. Then, given a scoring function ξ, $[\mathcal{G}_s]$ is a list of atoms from $\mathcal{G}_s$ ordered in the descending order. Finally, the *subject rank* [16] of a given ξ, $\mathit{subject\_rank}_{\xi}(a)$ is the position of a in $[\mathcal{G}_s]$. Analogously one can define $[\mathcal{G}_o]$ and the corresponding *object rank* [16] of a given ξ, that is, $\mathit{object\_rank}_{\xi}(a)$.

Now we are ready to define the function f for an atom a as the average of its subject and object inverted ranks given ξ [16], i.e.:

$$f_{\xi}(a) = 0.5 \times (1/\mathit{subject\_rank}_{\xi}(a) + 1/\mathit{object\_rank}_{\xi}(a)) \tag{4.2}$$

4.2.2 Realization of μ_1

This measure should reflect the descriptive quality of a given rule r with respect to $\mathcal{G}$. There are many classical data mining measures that can be used as μ_1, see, e.g. [29, 13, 43, 56] for μ_1s proposed specifically for KGs. In principle, we should choose a suitable measure that correctly describes the KG.

In our work we selected the following two measures for μ_1: *standard confidence* and *PCA confidence*. While standard confidence of a rule is the conditional probability of rule's head given its body, computed based on closed world assumption (CWA), PCA confidence is its generalisation to the open world assumption (OWA), which does not penalize rules that predict facts $p(s, o)$, such that $p(s, o') \notin \mathcal{G}$ for any o'.

4.2.3 Realization of μ_2

There are various ways how one can define the quality $\mu_2(\mathcal{G}_r, \mathcal{P})$ of $\mathcal{G}_r$. A natural candidate to define the quality of $\mathcal{G}_r$ is the probability of $\mathcal{G}_r$, that is, as $\mu_2(\mathcal{G}_r, \mathcal{P}) = \prod_{a \in \mathcal{G}_r} f(a) \times \prod_{a \in (\mathcal{R}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}} \times \mathcal{C}_{\mathcal{G}}) \setminus \mathcal{G}_r} (1 - f(a))$. A disadvantage of such quality measure is that in practice it will be very low, as the product of many (potentially) small probabilities, and thus Equation 4.1 will be heavily dominated by $\mu_1(r, \mathcal{G})$. Therefore, we advocate to define $\mu_2(\mathcal{G}_r, \mathcal{P})$ as the average probability of newly predicted facts in $\mathcal{G}_r$:

$$\mu_2(\mathcal{G}_r, \mathcal{P}) = (\Sigma_{a \in \mathcal{G}_r \setminus \mathcal{G}} f(a))/|\mathcal{G}_r \setminus \mathcal{G}|. \tag{4.3}$$

Example 4.1. *Consider the KG $\mathcal{G}$ in Figure 3.1, and the rules from Equations 1.1 and 1.2 with their confidence values as presented in Example 3.6. Suppose that a text-enhanced embedding model produced a relatively accurate estimation of the probabilities of facts over livesIn relation. For example, even though within the graph there is no direct connection between Germany and Berlin, relying on the living places of entities similar to John and hidden semantic relations between Germany and Berlin such as co-occurrences in text and other linguistic features, for the fact $a = livesIn(john, berlin)$ we obtained $f(a) = 0.9$, while for $a' = livesIn(john, france)$, a much lower probability $f(a') = 0.09$. These naturally support the predictions of r_2 but not those of r_1.*

Generalising this idea, assume that on the whole dataset we get $\mu_2(\mathcal{G}_{r_1}, \mathcal{P}) = 0.1$ and $\mu_2(\mathcal{G}_{r_2}, \mathcal{P}) = 0.8$, where $\mathcal{P} = (\mathcal{G}, f)$. Thus, for $\lambda = 0.5$ we have $\mu(r_1, \mathcal{P}) = (1 - 0.5) \times 0.5 + 0.5 \times 0.1 = 0.3$, while for $\mu(r_2, \mathcal{P}) = (1 - 0.5) \times \frac{1}{3} + 0.5 \times 0.8 \approx 0.57$, resulting in the desired ranking of r_2 over r_1 based on μ. □

4.3 Approach Overview

In this section, we present the main phases of our approach. Conceptually, our system generalizes the standard relational association rule learners [13, 17] to account for the feedback from the probabilistic function f. Following common practice [13] we restrict ourselves to rules that are *closed* and *safe*.

The input of the system are a KG, possibly a text corpus, and a set of user specified parameters that are used to terminate rule construction. These parameters include an embedding weight λ, a minimum threshold for μ_1, a minimum rule support *r-supp* and other *rule-related* parameters such as a maximum number of positive and negative atoms allowed in $body(r)$. The KG and text corpus are used to train the embedding model that in turn is used to construct the probabilistic function f. The rules r are constructed in the iterative fashion, starting from the head, by adding atoms to its body one after another until at least one of the termination criteria (that depend on f) is met. In parallel with the construction of r the quality $\mu(r)$ is computed.

In Figure 4.1, we present a high level overview of our approach, where arrows depict information flow between blocks. The *Rule Learning* block constructs rules over the input KG, *Rule Evaluation* supplies it with quality scores μ for rules r, using $\mathcal{G}$ and f,

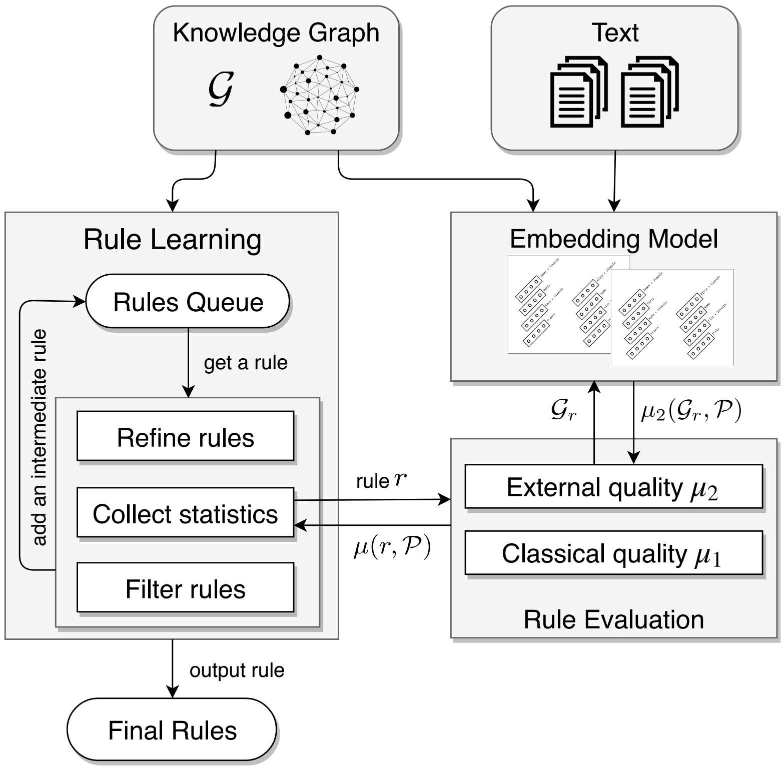


FIGURE 4.1: Overview of our system.

where f is computed by the *Embedding Model* block from $\mathcal{G}$ and text. Below, we describe in more detail each of these components in our system.

4.3.1 Embedding Model

Embedding model is the key component that is used to measure the external quality μ_2 of the rules. For the basic operation, embedding model gives each given a fact $p(s, o)$ a numerical value reflecting its likelihood. This component is pre-trained on the available KG and possibily with external data (e.g. texts), depending on the chosen model. Some prominent embedding models have been briefly described in chapter 2.

4.3.2 Rule Learning

Algorithm 1 demonstrates the mining process behind the *Rule Learning* block in Figure 4.1, in which we will discuss now. Following [13] we model rules as sequences of atoms, where the first atom is the head of the rule and other atoms are its body. The algorithm maintains a (priority) queue of intermediate rules (see the *Rules Queue* component in Figure 4.1). Initially the queue contains only an empty rule. At each iteration, a single rule is selected from the queue for processing. If the rule satisfies the *filtering criteria*

Algorithm 1: (Non-)monotonic Rule Mining

```
1  $queue \leftarrow \langle[]\rangle$
2  Execute in parallel:
3  while $\neg$queue.isEmpty() do
4      rule ← queue.dequeue()
       /* Computes rule statistics and output if necessary.          */
5      if rule.isClosed() then
6          stats ← computeStatistics(rule)
7          if stats is not pruned for outputting then
8              output(rule)
9          else
10             continue while loop
       /* Applies refinement operators to explore more new rules.    */
11     foreach operator o do
12         foreach newRule $\in$ o(rule) do
13             if newRule satisfies language bias then
                   /* Checks if there is some version of rule in queue.   */
14                 if newRule $\notin$ queue then
15                     queue.enqueue(newRule)
```

(*Filter rules* in Fig. 4.1) which we define below, then the system returns it as an output. If the rule is not filtered, then it is processed with one of the *refinement operators* (*Refine rules* in Fig. 4.1) that we define below that expand the rule with one more atom and produce new rule candidates, which are then pushed into the queue (if not being pushed before). The iterative process is repeated until the queue is empty. All the reported rules are finally ranked by the decreasing order of the hybrid measure μ, computed in *Collect statistics* block.

In the remainder of this section, we discuss refinement operators, filtering criteria, and how to check rule duplication efficiently.

4.3.2.1 Refinement Operators

We rely on the following three standard refinement operators [13] that extend rules:

(i) *add a positive dangling atom*: add a new positive atom with one fresh variable and another one appearing in the rule, i.e., *shared*.

(ii) *add a positive instantiated atom*: add a positive atom with one argument being a constant and the other one being a shared variable.

(iii) *add a positive closing atom*: add a positive atom with both of its arguments being shared variables.

Additionally, we introduce two more operators to allow negated atoms in rule bodies:

(iv) *add an exception instantiated atom*: add a negated atom with one of its arguments being a constant, and the other one being a shared variable.

(v) *add an exception closing atom*: add a binary negated atom to the rule with both of its arguments being shared variables or a unary negated atom with a shared variable.

These two operators are only applied to *closed* rules to ensure the *safety* condition. Moreover, we also require the satisfaction of the following condition: for a rule $r : head(r) \leftarrow body^+(r)$ the addition of an exception atom should result in the rule $r' : head(r) \leftarrow body^+(r), body^-(r)$, such that the following condition holds:

$$supp(r, \mathcal{G}) = supp(r', \mathcal{G}) \tag{4.4}$$

Intuitively, the application of exception refinement operators must not lead to the decrease of the rule support, i.e., exceptions should explain the absence of predictions expected to be in the graph rather then their presence.

4.3.2.2 Filtering Criteria

Without any filters and restrictions, we can theoretically explore all possible rules. However, most of them are useless for us. Hence, pruning strategies are needed to extract only meaningful rules as well as to enhance the performance of the algorithm. We classify these filtering criteria into 2 categories: *language-bias-based-filters* and *statistics-based-filters*.

- Language-bias-based Filtering.

After applying one of the refinement operators to a rule, a set of candidate rules is obtained. At this step, we can decide whether we should prune them by directly looking at their format. Following other rule mining systems, we also aim at mining only *closed* rules from the knowledge graphs. For rules having non-empty negated part $body^-(r)$, we allow only rules having closed positive part. Note that while non-closed rules are not returned in the output, they still be pushed into the rule queue to construct new rules, since a closed rule can be generated only by extending some other non-closed rule.

In addition, we also put some thresholds on the form of rules we want to extract from the knowledge graph, including:

- Number of distinct variables appearing in rule.
- Number of all atoms, positive atoms and negated atoms.
- Variable degree, which is the maximum number of atoms using the same variable as their arguments.
- Number of predicate occurrences in a rule.

All of the above restrictions are checked in line 13 of Algorithm 1. This ensures that all rules we push into the queue are exactly the ones we are interested in.

- Statistics-based Filtering.

Filtering by language bias only introduces restrictions on the form of rules to be extracted from the KG, without looking at the their quality. So, rules' statistics are collected by the *Rule Evaluation* block in Figure 4.1 (line 6 of Algorithm 1), and in line 7 of Algorithm 1, we filter the rules based on these statistical metrics as follows:

- *Rule support*: Rules that have low support are likely to be noise, so we filter out all rules having support less than some minimum threshold.

- *Head coverage*: Since we are not interested in rules that cover only a small portion of facts over the head predicate, following [13], we prune out all rules having head coverage below some defined threshold.

- *Classical quality μ_1*: If the classical quality μ_1 of the rule is too low, it is very likely that its external quality μ_2 is likewise low, leading to a very low value of the hybrid quality μ. Hence, we filter out all rules whose classical quality μ_1 is below a defined threshold. Obviously, by filtering rules based on their quality, we not only get rid of nonpromising rules but also restrict the rule search space, thus improving the performance of the whole system. More specifically, since querying the embedding model for μ_2 is a time-consuming operation, we only do this for rules whose μ_1 is above a minimum value.

- *Increasing hybrid quality*: For each candidate rule we check whether the application of refinement operators results in the increase of the hybrid measure μ, and discard the rule otherwise.

- *Exception confidence*: We propose the novel exception confidence measure, which is computed after the addition of an exception atom to a rule. Formally, given a rule of the form $r : \mathit{head}(r) \leftarrow \mathit{body}^+(r), \mathit{not}\ \mathit{body}^-(r)$, its exception confidence is computed as:

$$\mathit{e\text{-}conf}(r, \mathcal{G}) = \mathit{conf}(r', \mathcal{G}) \tag{4.5}$$

where $r' : \mathit{body}^-(r) \leftarrow \mathit{body}^+(r), \mathit{not}\ \mathit{head}(r)$. If the *e-conf* is below some user specified threshold, then the rule is dropped.

Intuitively, exception confidence is the conditional probability of the exception given predictions produced by the Horn part of r, which helps to disregard insignificant exceptions, i.e., those that explain the absence in $\mathcal{G}$ of only a small fraction of predictions made by $\mathit{head}(r) \leftarrow \mathit{body}^+(r)$, as such exceptions likely correspond to noise. Note that by exploiting the embedding feedback, we can now distinguish exceptions from noise. Consider the rule stating that married people live together. This rule can have several possible exceptions, e.g., either one of the spouses is a researcher or he/she works at a company, which has headquarter in the US. Whenever the rule is enriched with an exception, naturally, the support of its body decreases, i.e., the size of $\mathcal{G}_r$ goes down. Ideally, we want to add such negated atoms, that the average quality of $\mathcal{G}_r$ increases, as this will witness that by adding negated atoms to the rule we get rid of unlikely predictions.

Observe that not all of the filtering criteria are relevant for all rule types. For example, exception confidence is relevant only for non-monotonic rules to ensure the quality of the added exceptions. Moreover, increasing hybrid quality is only used as a filtering criterion for closed rules. Indeed, if the rule is non-closed, this filter is not meaningful [13].

4.3.2.3 Checking Rule Duplication

Since one rule could be generated multiple times from applying mining operators in different orders, and it does not make sense to process a rule more than once, we will prune all duplicated versions of the same rule. At line 14 of Algorithm 1, we push a rule into the queue only if none of its versions is already pushed in the queue. Because the queue contains rules in increasing order of the number of atoms, we can ensure that all duplicated versions of the same rule will be checked through this line of code before the first one of them is processed.

To check if twos rule are duplicated, we can check if there exists a bijection mapping variables between the two rules, such that the structure of the two rules with mapped variables are exactly the same. To check if a rule already exists in the queue, theoretically, we need to compare it with all other rules in the queue. However, we could reduce the number of rules to be compared with by using a hash table, where we encode each rule with a hash code in such a way that duplicated rules must have the same hash code. A good hash function should minimize the number of collisions (i.e. total number of different rules having the same hash code). There are many ways to define a hash code function. For example, we could exploit the number of variables, number of atoms of each type, number of atoms of each predicate, variables degree, graph structure of the rule, etc. An optimization to support the rule duplication checking process is to impose a constraint on the order of refinement operators to be applied on the rule.

4.3.3 Rule Evaluation

Another key component of our system is the *Rule Evaluation*. Obviously, the question here is how to compute rule statistics. We now briefly describe how we answer this question.

For each rule $r : H \leftarrow B, not\ E$, where $H = h(X, Y)$ we need to calculate its statistics (e.g. head coverage hc, classical quality μ_1, hybrid quality μ, and possibly exception confidence *e-conf* if the rule is non-monotonic). The statistics computation involves a subtask of finding all instances of the head variables (X, Y) satisfying the body of the rule. Then, head coverage can be derived by computing the ratio of the number of such pairs (X, Y) satisfying the head of the rule (i.e. rule support) to the total head support. Standard confidence is calculated based on the ratio of rule support to the body support. External quality μ_2 is computed by querying the embedding model. Other metrics can be derived accordingly.

Algorithm 2 describes how we find the head instances (X, Y) that satisfy the rule body. The general idea is to try recursively assigning constants $c \in \mathcal{C}$ to variables to satisfy body atoms one by one. Once the current variables assignment cannot satisfy the given atom (line 12), we backtrack to the previous atom and assign different values to variables. For each variables assignment that satisfies all the body atoms, we insert the value of pair (X, Y) into the result (line 5), which is an empty set at the beginning.

4.3.3.1 Optimized Computation of External Rule Quality

Clearly, computing the external rule quality μ_2 is a heavy operation. Indeed, to compute μ_2, we need to compute the probabilistic value f for each fact predicted by the rule and take the average of them. Moreover, to calculate f for a fact, one needs to calculate its *subject rank*[16] and *object rank*[16] by looping through all substitutions of subject

Algorithm 2: Find Body Instances

```
Input  : Rule body $[B_1, B_2, ..., B_n]$ (may contains exceptions) with head variables
         $(X, Y)$
Output : Instances of $(X, Y)$
1  $ins \leftarrow \{\}$ // Stores instances of $(X, Y)$.
2  $val \leftarrow$ array of null values // Stores variables assignments while
   backtracking.
3  Function FillAtom(bodyIndex)
4      if $bodyIndex = n + 1$ then
5          $ins \leftarrow ins \cup \{(val[X], val[Y])\}$
6          return
7      $atom \leftarrow B_{bodyIndex}$
8      $null\_atom\_variables \leftarrow \{variables\ v \in atom \mid val[v]\ is\ null\}$
9      if null_atom_variables is empty then
10         if val satisfies atom then
11             FillAtom(bodyIndex + 1)
12         else
13             return
14     else
15         foreach values of null_atom_variables that satisfy $[B_1, B_2, ..., B_{bodyIndex}]$ do
16             $val[null\_atom\_variables] \leftarrow$ assigned values
17             FillAtom(bodyIndex + 1)
18             $val[null\_atom\_variables] \leftarrow null$
19 FillAtom(1)
20 return ins
```

and object with other entities and then querying the embedding model for the likelihood scores ξ.

If the number entities and/or the number of facts predicted by the rule is high, which is very likely for modern KGs, then computing the value of μ_2 is too time-consuming. So, a full implementation to compute μ_2 will lead to a very poor performance of the mining algorithm. Below, we discuss possible optimizations to fix this issue.

- *Facts sampling*: First, notice that the external quality function μ_2 is defined as the average probability f of rule predictions. Hence, the first thing we can do is to take a sample of the rule predictions and estimate their average probability f based on this set. This solution would speed up the calculation of μ_2 in the situation when a rule predicts a very large number of facts, which usually happens for low confidence rules on large knowledge graphs.

- *Subject/Object ranks early stopping*: To compute the probability f of a fact based on the Equation 4.2, we need to compute its *subject/object ranks*. It is easy to see that the value of f does not change significantly when the subject/object ranks rise to a large enough value. Based on this observation, we can set thresholds on the maximum values of the subject and object ranks, and when looping through the substitutions of the subject/object, we will stop whenever the current subject/object ranks touch these thresholds.

- *Caches for Embedding model*: Another relevant optimization for computing μ_2 efficiently is to use caching for querying the likelihood scores ξ. While the time required for retrieving an answer of a query for computing ξ depends on the choice of the embedding model as well as its hyper parameters, using a Least Recently Used (LRU) cache might be in any case helpful. In principle, we can even use multiple caches, i.e. a separate one for each KG relation.

- *Optimized Embedding model*: The final optimization is at the calculation of the likelihood function ξ alone. This advantageous optimization highly depends on the chosen embedding model and its likelihood formula. Not all models can be improved at this step, and to achieve reasonable enhancements, a deep understanding of the model is required. In our work, since we rely on available implementations of embedding models, this optimization is not applied.

Chapter 5

RuLES - Rule Learning with Embedding Support

Within this thesis, we have developed the Rule Learning with Embedding Support (RuLES[1]) system that implements our hybrid rule learning approach. In this chapter, we provide details about the system implementation and its usage.

5.1 System Implementation

The Algorithms 1 and 2 require the following basic operations:

- Check whether a given fact is true or unknown in the given KG.
- Retrieve all triples having a specific predicate.
- Get all outgoing edges from a particular subject.
- Get all incoming edges to a particular object.
- Get all edges between a pair of a subject and an object.

To meet these requirements, following [13], we store the knowledge graph as an in-memory database and index it using various data structures and techniques including arrays, lists, hashtables, maps, sets. Our system RuLES is implemented in Java 8 and reuses the available implementations of embedding models in different languages. Currently, the system runs on Linux and in the future we may extend it to support Windows. In what follows, we describe in details the usage of our system.

[1]https://github.com/hovinhthinh/RuLES

5.2 Prerequisites

As the prerequisites, the following software needs to be installed:

- For the mining system: `jdk (v1.8.0), ant (v1.9.4)`
- For the embedding models, we currently support TransE, HolE and SSP models. These models require the following software:
  - TransE, HolE: `python (v2.7.9), numpy (v1.13.1), scipy (v0.19.1), scikit-learn (v0.19.0)`
  - SSP: `icc (v18.0.1), boost (v1.55.0.2), armadillo (v4)`

5.3 Installation

After installing the necessary software, one needs to download the source code of the RuLES system and build it using the following commands:

```
$ cd mining/ && ant build && cd ../
```

For the embedding models, we reuse their existing implementations. Since the implementations of TransE and HolE are in Python, there is no need to compile their source code. However, in case we want to use SSP model, which is implemented in C++, we need to run the following command to compile it:

```
$ icc -std=c++11 -O3 -qopenmp -larmadillo -xHost embedding/ssp/ssp_main.cpp
-o embedding/ssp_main
```

5.4 Data Preparation

Prior to running the system, one needs to prepare a `<workspace>` directory containing the file `ideal.data.txt`, which stores all facts in the input KG. Each line of this file describes a triple of the input KG in the RDF form:

```
subject[tab]predicate[tab]object
```

where `subject`, `predicate` and `object` are strings without spaces. For representing unary facts, the predicate `<type>` (with the brackets) should be used, and `object` should present the `subject`'s class as usual:

```
entity[tab]<type>[tab]class
```

We provide an example of the input file below:

```
He_Would_a_Hunting_Go      directedBy      George_Nichols
Tommy_Lee_Jones      actedIn      The_Missing
Where_Eskimos_Live      producedIn      Germany
Too_Beautiful_for_You      <type>      French-language_films
```

Additional external data sources are required depending on the chosen embedding model. If one uses TransE or HolE models, no external data is needed. However, for the usage of SSP model, the file `entities_description.txt` should be provided in the `<workspace>` directory. Each line of this file contains the description of an entity in the following form:

```
entity[tab]description
```

Here, `description` is space-separated string and should be preprocessed (e.g. trim, to lower case, remove special characters). Below, we provide an example of the description file:

```
Oklahoma      state of the united states of america
Falkland_Islands      archipelago in the south atlantic ocean
London      capital of england and the united kingdom
```

5.5 Data Sampling

In the next step, the following command should be run to sample the training KG:

```
$ bash gen_data.sh <workspace> <training_ratio>
```

where `<workspace>` is the workspace folder as described before, and `<training_ratio>` denotes the ratio of the size of the training KG to the size of the input KG. Intuitively, if we want to run the system on the whole input KG, a `<training_ratio>` of 1 should be used. In the experiments described in Section 6, the `<training_ratio>` is set to 0.8.

5.6 Embedding Model Training

Depending on the desired embedding model, we run the corresponding commands to train the models.

- TransE model: `$ bash run_transe.sh --workspace <workspace> --margin <margin> --lr <learning_rate> --ncomp <embedding_dimension>`
- HolE model: `$ bash run_hole.sh --workspace <workspace> --margin <margin> --lr <learning_rate> --ncomp <embedding_dimension>`
- SSP model: `$ ./embedding/ssp_main <workspace> <embedding_dimension> <learning_rate> <margin> <balance_factor> <joint_weight>`

where `<workspace>` is the prepared workspace folder; `<embedding_dimension>` is the dimension of the representation vectors of subjects, predicates, objects used in these models; `<margin>` is the margin exploited in the margin-based pairwise loss training technique; `<learning_rate>` indicates the learning rate used by the Stochastic Gradient Descent training algorithm. Moreover, `<balance_factor>` and `<joint_weight>` are the two parameters used by the SSP model, denoting the balance factor in the likelihood scoring function (see 2.1.3) and correspondingly the joint weight between the embedding-specific and topic-specific objective training functions. We refer to [5, 33, 53] for a better understanding of the parameters used by these models.

5.7 End-to-End System Execution

The following command runs the mining system in its most basic setting:

```
$ java -jar mining/build.jar -w <workspace> -em <embedding_model>
```

where `<workspace>` is the prepared data folder, and `<embedding_model>` is equal to either `transe`, `hole` or `ssp`, corresponding to the embedding model being used. The system outputs mined rules to the file `<workspace>/rules.txt` on the fly. After the mining process is finished, all extracted rules are ranked in the decreasing order of the hybrid quality μ and then written to the file `<workspace>/rules.txt.sorted`.

In the rest of this chapter, we list the command line options supported by RuLES, describe them in details and also discuss the extendability of our system.

5.8 Command Line Options

Our system supports the following command line options:

- `-w,--workspace <arg>`: Mandatory option denoting the working folder.
- `-em,--embedding_model <arg>`: Mandatory option denoting the embedding model being used (i.e. `transe`, `hole` or `ssp`).
- `-ew,--embedding_weight <arg>`: Embedding weight λ of the hybrid scoring function (default: 0.3).
- `-nw,--num_workers <arg>`: Number of parallel workers (default: 8).
- `-o,--output <arg>` : Output file path (default: `<workspace>/rules.txt`). Mined rules are written to this file on the fly. They will finally be ranked and written to the file `<output>.sorted`.

Language-bias-based Options:

- `-nv,--max_num_var <arg>`: Maximum number of variables of extracted rules (default: 3).
- `-vd,--max_var_deg <arg>`: Maximum variables degree (number of atoms sharing a variable) in the extracted rules (default: 3).
- `-nupo,--max_num_uniq_pred_occur <arg>`: Maximum number of predicate occurrences in the extracted rules (default: 2).
- `-na,--max_num_atom <arg>`: Maximum number of atoms in the extracted rules (default: 3).
- `-nna,--max_num_neg_atom <arg>`: Maximum number of negated atoms in the extracted rules (default: 0).

Statistics-based Options:

- `-pca,--use_pca_conf`: Use PCA confidence instead of standard confidence as the measure μ_1.
- `-ms,--min_support <arg>`: Minimum support of extracted rules (default: 10).
- `-hc,--min_hc <arg>`: Minimum head coverage of extracted rules (default: 0.01).
- `-mc,--min_conf <arg>`: Minimum confidence of extracted rules (default: 0.1).

- `-ec,--min_ec <arg>`: Minimum exception confidence of extracted non-monotonic rules (default: 0.05).

5.9 System Extendability

Our system is flexible for plugging in an arbitrary embedding model. Below, we briefly discuss how to do this in Java.

- **Step 1**: After sampling the data as described in Section 5.5, our system generates several files in the `<workspace>` directory. Among the generated files, the file `<workspace>/meta.txt` is crucial for exploiting additional embedding models. It stores the signature of the given KG in the following format:
  - The first two numbers of the first line are the number of entities e and relations r appearing in the KG, respectively. We index the entities with IDs from 0 to $(e-1)$ and correspondingly the relations with IDs from 0 to $(r-1)$.
  - Each line of the next e lines stores the string value of one entity of the KG, from 0^{th} entity to $(e-1)^{th}$ entity.
  - Each line of the next r lines stores the string value of one relation of the KG, from 0^{th} relation to $(r-1)^{th}$ relation.
- **Step 2**: Train the custom embedding model on the given KG. In this step, we can reuse existing implementations of the embedding model in any language.
- **Step 3**: Create a subclass of the abstract class `de.mpii.embedding.EmbeddingClient` that works as a bridge between our mining system and the pre-trained embedding model. This class must implement the following methods:
  - A constructor that has a `String workspace` as one parameter (denoting the `<workspace>` folder) and calls the following constructor of the superclass: `super(workspace);`
  - A function that overrides the abstract method `public abstract double getScore(int subject, int predicate, int object);` returning the likelihood score of a fact given its subject, predicate and object ids, computed based on the pre-trained embedding model. The mapping of these ids to their real values is described in the signature file `meta.txt` above. To interact with the embedding model within this function, a naive strategy is to write all optimized parameters of the trained model from step 2 into a file, and then reload them in the constructor of this class.

We refer to 3 classes `de.mpii.embedding.TransEClient`, `de.mpii.embedding.HolEClient`, and `de.mpii.embedding.SSPClient` for examples.

- **Step 4**: Edit the constructor of the class `de.mpii.mining.Miner` to add a short name (e.g. `transe`,`hole` or `ssp` as we currently have) and a class instantiation for the custom embedding model.
- **Step 5**: Simply use the chosen short name as the value for the parameter `-em,--embedding_model <arg>` when executing the mining system.

Chapter 6

Evaluation

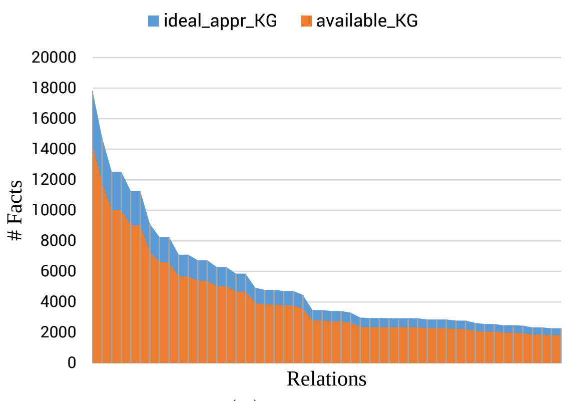
We have conducted experiments on a Linux machine with 80 cores and 500GB RAM to examine our system RuLES. In this section we report the results of our experimental evaluation, which focuses on *(i)* the benefits of our hybrid embedding-based rule quality measure over traditional rule measures; *(ii)* the effectiveness of RuLES against the state-of-art Horn rule learning systems; and *(iii)* the quality of non-monotonic rules learned by RuLES compared to existing methods.

6.1 Experimental Setup

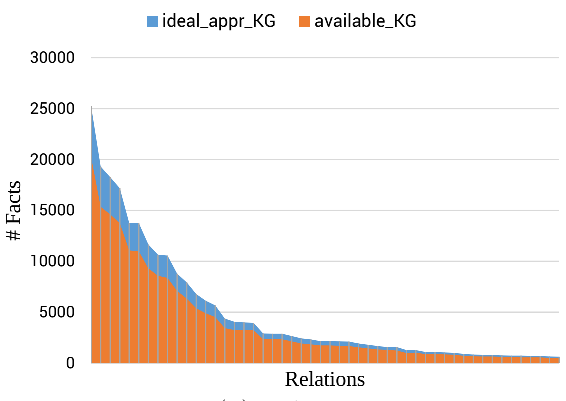
6.1.1 Datasets

We performed experiments on the following real world datasets:

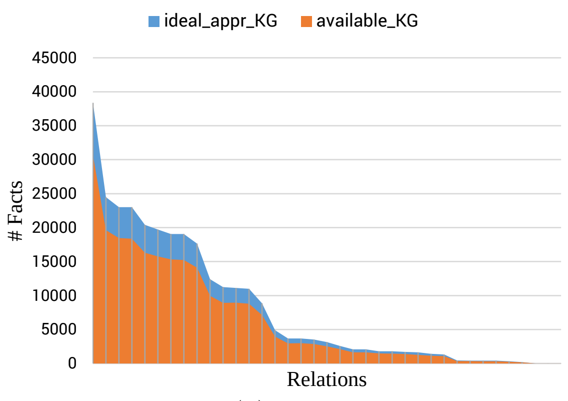
- *Freebase*: This is a huge knowledge graph consisting of general facts. To meet the requirement of running both rule mining and embedding, we adopt *FB15K* [5], a subset of Freebase containing 15K entities, 1345 binary predicates and 592K binary facts, commonly used for evaluating KG embedding models [47].
- *Wikidata*: This is a free, community-based knowledge base maintained by the Wikimedia Foundation. In our experiments, we created a subset of Wikidata dump from December 2014 (also used in [13]) by choosing triples that have entities appearing at least 20 times in the whole dataset, and then selecting top 100 predicates that have most number of facts. This results in a dataset with 250K binary facts over 44K entities and 100 relations, which we refer to as *Wiki44K*.



(A) FB15K



(B) Wiki44K



(C) IMDB

FIGURE 6.1: Distribution of facts over relations of KGs $\mathcal{G}^i_{appr}$ and $\mathcal{G}$.

- *IMDB*[1]: We construct a domain-specific KG from *IMDB* dataset, which is also used in [44]. The KG consists of 118K entities, 37 predicates and 301K binary facts.

In the experiments, for each incomplete KG $\mathcal{G}$, we need its *ideal* completion $\mathcal{G}^i$ that would give us a gold standard for evaluating our approach and comparing it to others. Since obtaining a real life $\mathcal{G}^i$ is hard, we used the KGs *FB15K*, *Wiki44K*, and *IMDB* as reference graphs $\mathcal{G}^i_{appr}$ that approximate $\mathcal{G}^i$. We then constructed $\mathcal{G}$ by randomly selecting 80% of its facts while preserving the distribution of facts over predicates. Figure 6.1 demonstrates the distribution of facts over top 50 predicates for the 3 datasets.

6.1.2 Embedding Models

TransE [5] and HolE [33] are two state-of-the-art embedding models that do not account for external data. In contract, SSP [53] is a state-of-the-art model which accounts for

[1]http://www.imdb.com/

DATASET	FB15K				Wiki44k				IMDB			
MODEL	MRR		Hits@10(%)		MRR		Hits@10(%)		MRR		Hits@10(%)	
Eval. setting	*Raw*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Filt.*
TransE	0.23	0.33	47.48	59.64	0.22	0.26	39.23	43.58	0.26	**0.31**	36.63	**40.39**
HolE	0.24	0.36	47.54	60.45	0.14	0.18	24.54	28.38	0.21	0.26	28.27	32.08
SSP	0.29	**0.45**	55.73	**70.35**	0.26	**0.31**	45.15	**51.05**	–	–	–	–

Table 6.1: Performance of embedding models on the three data sets.

short textual descriptions attached to KG entities. We focus on these 3 embedding models in this work, since they are efficient and relatively simple.

We reuse existing implementations of TransE, HolE[2], and SSP[3] and trained these embedding models on the available KGs until convergence using Stochastic Gradient Descent. While we train embedding models on *IMDB* without external text, these embedding models are trained on *FB15K* and *Wiki44K* with 2 different settings: with and without the external textual data. In particular, each entity of *FB15K* and *Wiki44K* links to a small piece of description text, which is extracted from the corresponding Wikidata page. Furthermore, we discard all entities having empty description text for both experimental settings.

For evaluation protocol of the embedding models, we followed the same method of HolE [33]. To validate and test the models, we sampled 2000 facts from $\mathcal{G}^i_{appr} \setminus \mathcal{G}$ into the validation set, and other 2000 facts into the test set. We report the Mean Reciprocal Rank and the percentage of Hit@10 in Table 6.1. The optimal hyperparameters of each model and dataset are computed via grid search. We select the hyperparameters, which result in the highest MRR score with *filtered* setting on the validation set. We refer to [33] for details.

According to Table 6.1, we compared the effectiveness of the models and selected for each KG the best one. Apart from SSP, which showed the best performance on both FB15K and Wiki44K, we also selected HolE for FB15K and TransE for Wiki44K and IMDB. Note that in this work as a proof of concept we considered some of the most popular embedding models, but conceptually any model (see [47] for overview) can be used in our system.

6.1.3 Evaluation Metrics

To evaluate the learned rules we use the quality of predictions that they produce when applied on $\mathcal{G}$, i.e., the more correct facts beyond $\mathcal{G}$ a ruleset produces, the better it is.

[2]urlhttps://github.com/mnick/scikit-kge
[3]urlhttps://github.com/bookmanhan/Embedding

We consider two evaluation settings: *closed world* setting (CW) and *open world* setting (OW). In the CW setting, we define the prediction precision of a rule r and a set of rules R as:

$$pred\_prec_{CW}(r) = \frac{|\mathcal{G}_r \cap \mathcal{G}^i_{appr} \setminus \mathcal{G}|}{|\mathcal{G}_r \setminus \mathcal{G}|}, \quad pred\_prec_{CW}(R) = \frac{\sum_{r \in R} pred\_prec_{CW}(r)}{|R|}$$

In the OW setting, we also take into account the incompleteness of $\mathcal{G}^i_{appr}$ and consider the quality of predictions outside it by performing a random sampling and manually annotating the sampled facts relying on Web resources such as Wikipedia. Thus, we define the OW prediction precision $pred\_prec_{OW}$ for a set of rules R as follows:

$$pred\_prec_{OW}(R) = \frac{|\mathcal{G}' \cap \mathcal{G}^i_{appr}| + |\mathcal{G}' \backslash \mathcal{G}^i_{appr}| \times accuracy(\mathcal{G}' \backslash \mathcal{G}^i_{appr})}{|\mathcal{G}'|}$$

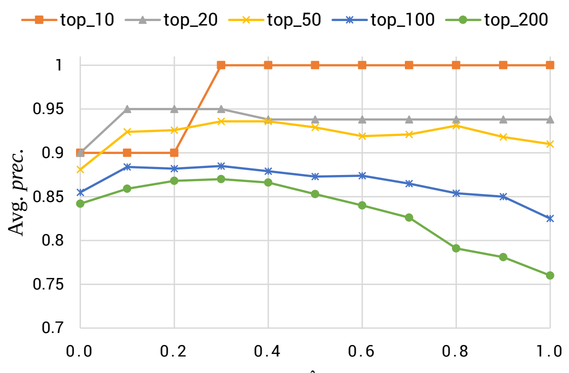
where $\mathcal{G}' = \bigcup_{r \in R} \mathcal{G}_r \backslash \mathcal{G}$ is the union of predictions generated by rules in R, and $accuracy(S)$ is the approximated ratio of true facts inside S computed via manual checking of facts sampled from S. Finally, to evaluate the meaningfulness of exceptions in a rule (i.e., negated atoms) we compute the *revision precision*, which according to [44] is defined as the ratio of incorrect facts in the difference between predictions produced by the Horn part of a rule and its non-monotonic version over the total number of predictions in this difference (the higher the revision precision, the better the rule exceptions) computed per ruleset. Formally,

$$rev\_prec_{OW}(R) = 1 - \frac{|\mathcal{G}'' \cap \mathcal{G}^i_{appr}| + |\mathcal{G}'' \backslash \mathcal{G}^i_{appr}| \times accuracy(\mathcal{G}'' \backslash \mathcal{G}^i_{appr})}{|\mathcal{G}''|}$$

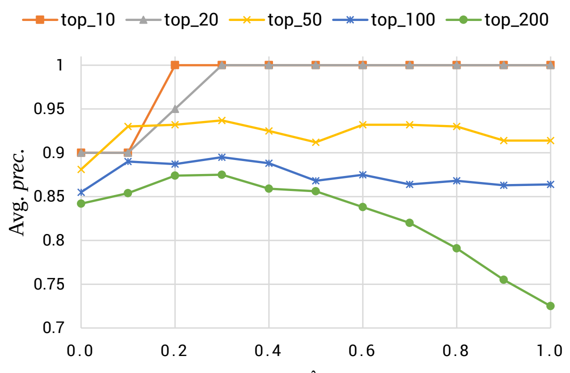
where $\mathcal{G}'' = \mathcal{G}_H \backslash \mathcal{G}_R$ and H is the set of Horn parts of rules in R. Intuitively, $\mathcal{G}''$ contains facts not predicted by the rules in R but predicted by their Horn versions.

6.1.4 RuLES Configuration

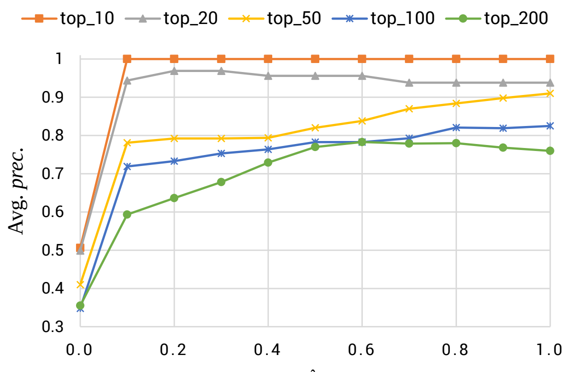
We run RuLES in several configurations where μ_1 is set to either *standard confidence (Conf)* or *PCA confidence (PCA)*. Furthermore, while we require $\mu_1 \in [0, 1]$, we also consider *non*-$[0, 1]$ metrics, which does not hold this condition, to realize μ_1 (e.g. *conviction*). Meanwhile, μ_2 is computed based on either TransE, HolE, or SSP models. Throughout the experiments, the configurations are named as μ_1-μ_2 (e.g. Conf-HolE).



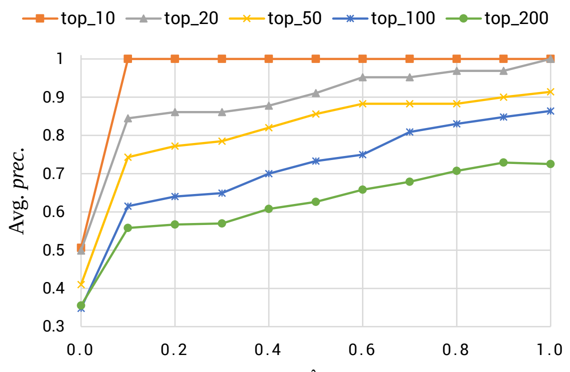
(A) Conf-HoLE on FB15K



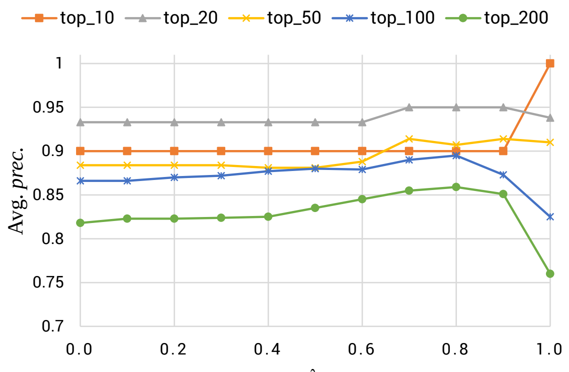
(B) Conf-SSP on FB15K



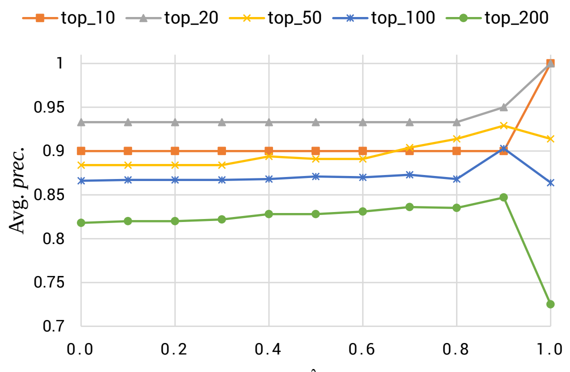
(C) PCA-HolE on FB15K



(D) PCA-SSP on FB15K



(E) Conv-HolE on FB15K



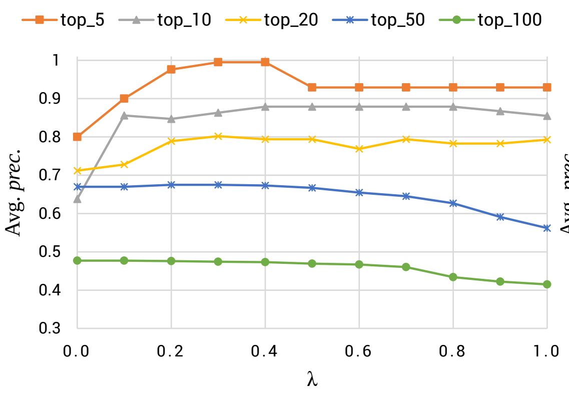
(F) Conv-SSP on FB15K

FIGURE 6.2: Avg. prediction precision of the *top-k* rules with various *embedding weights* on FB15K dataset.

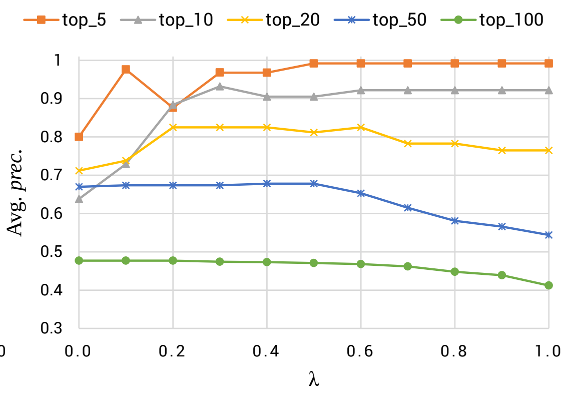
6.2 Embedding-Based Hybrid Quality Function

6.2.1 Hybrid quality function versus standard rule metrics

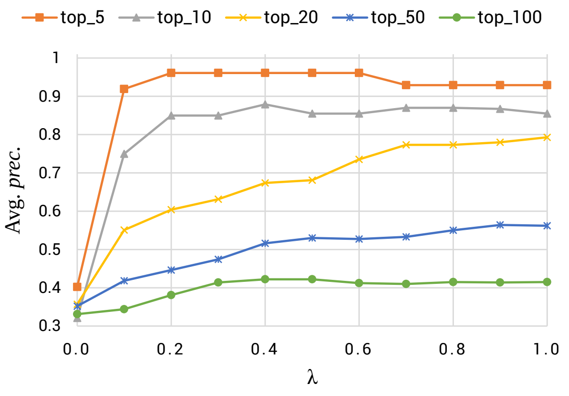
In this experiment we study the effect of using our hybrid embedding-based rule measure μ from Equation 4.1 on the rule ranking compared to traditional measures. We do it by first learning rules of the form $r: \ h(X,Z) \leftarrow p(X,Y), q(Y,Z)$ from $\mathcal{G}$ where



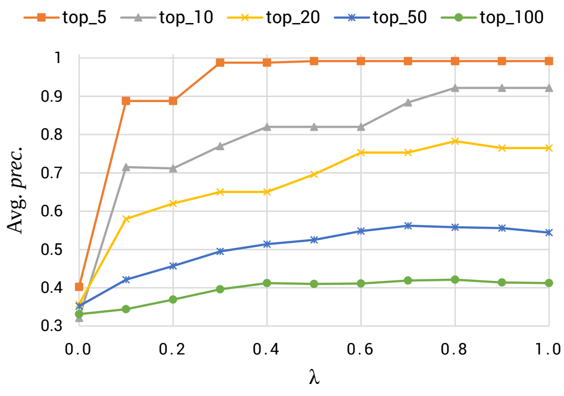
(A) Conf-TransE on Wiki44K



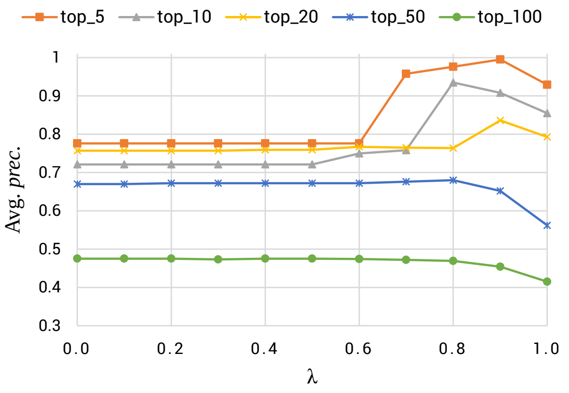
(B) Conf-SSP on Wiki44K



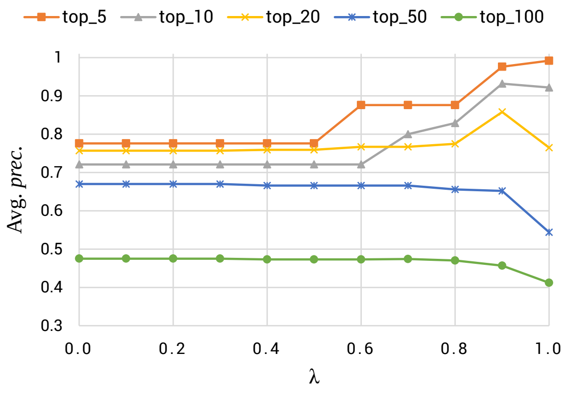
(C) PCA-TransE on Wiki44K



(D) PCA-SSP on Wiki44K



(E) Conv-TransE on Wiki44K

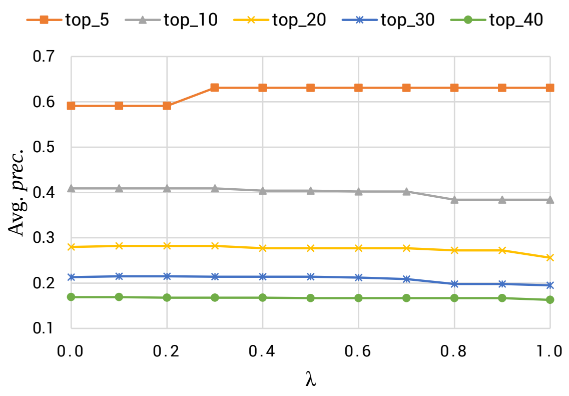


(F) Conv-SSP on Wiki44K

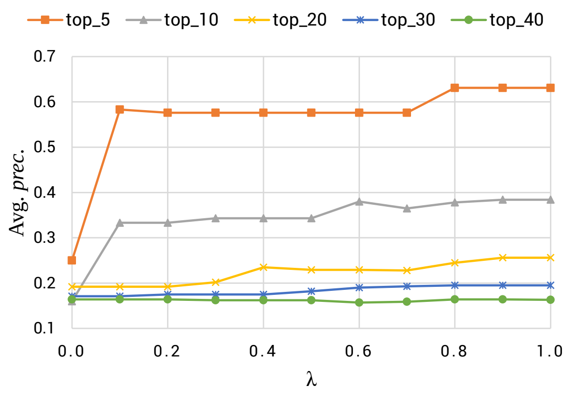
FIGURE 6.3: Avg. prediction precision of the *top-k* rules with various *embedding weights* on Wiki44K dataset.

r-supp$(r, \mathcal{G}) \geq$ *10*, *conf*$(r, \mathcal{G}) \in [$*0.1*, *1*$)$ and *h-cover*$(r, \mathcal{G}) \geq$ *0.01*. Then, we rank these rules using Equation 4.1 with $\lambda \in \{0, 0.1, 0.2, \ldots, 1\}$, $\mu_1 \in \{conf, conf_{pca}, conv\}$ and with μ_2 that is computed by relying on TransE, HolE and SSP. Note that $\lambda = 0$ corresponds to the standard rule measure μ_1 alone.

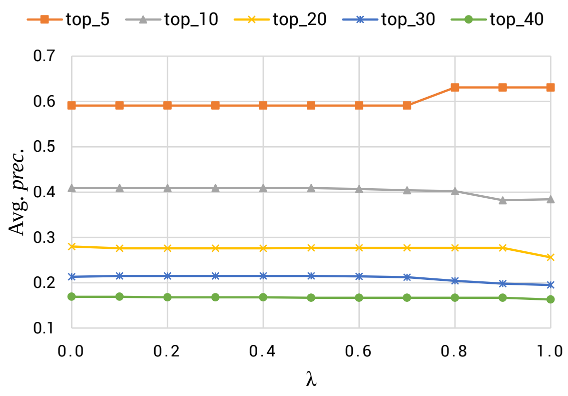
Figures 6.2, 6.3 and 6.4 show the average prediction precision *pred_prec*$_{CW}$ of the *top-k* rules ranked using our measure μ for different embedding weights λ (*x-axis*). In Figures 6.2a, 6.2b, 6.3a, 6.3b and 6.4a we observe that combining standard confidence



(A) Conf-TransE on IMDB



(B) PCA-TransE on IMDB



(C) Conv-TransE on IMDB

FIGURE 6.4: Avg. prediction precision of the *top-k* rules with various *embedding weights* on IMDB dataset.

($\mu_1 = conf$)with any embedding model increases the average prediction precision for $0 \leq \lambda \leq 0.3$. Moreover, we observe the decrease of prediction precision for $0.4 \leq \lambda \leq 1$ and *top-k* rules learned from FB15K when $k \geq 20$ and from Wiki44K when $k \geq 10$. This shows that the combination of μ_1 and μ_2 gives noticeable positive effect on the prediction results. On the other hand, for $\mu_1 = conf_{pca}$ the precision increases significantly when combined with embedding models and only decreases slightly for $\lambda = 1$. Utilizing $conf_{pca}$ instead of $conf$ as μ_1 in our hybrid measure is less effective, since our training data $\mathcal{G}$ is randomly sampled breaking the partial completeness assumption adopted by the PCA confidence. Meanwhile, with $\mu_1 = conv$, the best value of embedding weight λ is close to 1. This is reasonable, since conviction could give us a value greater than 1.

On IMDB dataset, the usage of our hybrid quality measure does improve the quality of result. However, the improvement is not easily noticeable. One explanation is that this dataset is very sparse, which could lead to the ineffectiveness of the embedding models.

Table 6.2 compactly summarizes the average prediction precision of *top-k* rules ranked by the standard rule measures and our μ for the best value of $\lambda = 0.3$ and highlights the effect of using the better embedding model (text-enhanced vs standard). On *FB15K*

top-k	FB15K				
	Conf ($\lambda = 0$)	**PCA** ($\lambda = 0$)	**Conv** ($\lambda = 0$)	**Conf-HolE** ($\lambda = 0.3$)	**Conf-SSP** ($\lambda = 0.3$)
5	0.800	0.638	**1.000**	**1.000**	**1.000**
10	0.900	0.506	0.900	**1.000**	**1.000**
20	0.900	0.499	0.933	0.950	**1.000**
50	0.881	0.410	0.884	0.936	**0.937**
100	0.855	0.348	0.866	0.885	**0.895**
200	0.842	0.355	0.818	0.870	**0.875**

top-k	Wiki44K				
	Conf ($\lambda = 0$)	**PCA** ($\lambda = 0$)	**Conv** ($\lambda = 0$)	**Conf-Tr.E** ($\lambda = 0.3$)	**Conf-SSP** ($\lambda = 0.3$)
5	0.800	0.402	0.776	**0.995**	0.968
10	0.638	0.321	0.721	0.863	**0.932**
20	0.712	0.357	0.757	0.802	**0.825**
50	0.670	0.352	0.670	**0.675**	0.674
100	**0.477**	0.331	0.475	0.474	0.474

top-k	IMDB			
	Conf ($\lambda = 0$)	**PCA** ($\lambda = 0$)	**Conv** ($\lambda = 0$)	**Conf-Tr.E** ($\lambda = 0.3$)
5	0.591	0.250	0.591	**0.631**
10	**0.409**	0.160	**0.409**	**0.409**
20	0.280	0.192	0.280	**0.282**
30	0.213	0.171	0.213	**0.214**
40	**0.169**	0.164	**0.169**	0.168

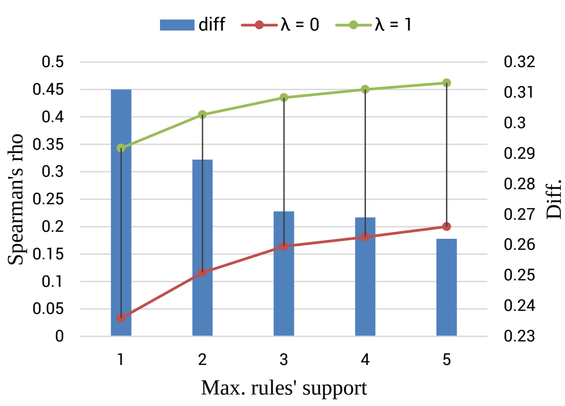
Table 6.2: Avg. prediction precision of the rules learned using different measures.

and *Wiki44K*, we observe that the accuracy of a utilized embedding model is naturally propagated to the accuracy of the rules that we obtain using our hybrid ranking measure μ. This demonstrates that the use of a better embedding model positively effects the quality of learned rules.

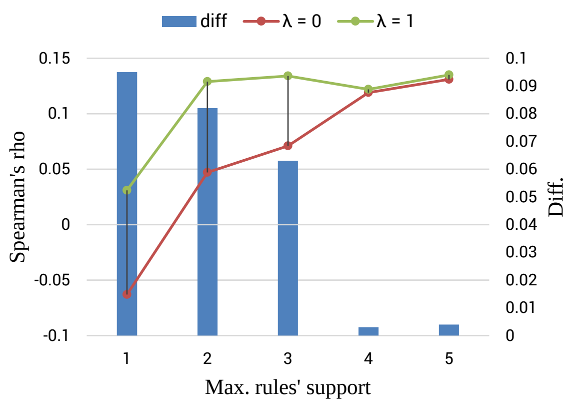
6.2.2 Handling rules with low support

Our hybrid quality function is especially beneficial for assessing rules with low support. Indeed, standard confidence is unreliable for such rules, as it estimates rules' quality by relying only on a small number of rules' instantiations. For example, ignoring all rules having confidence 1 since they do not infer any new facts, when a rule has support 1, its confidence is never greater than 0.5, even though the predicted facts may be true.

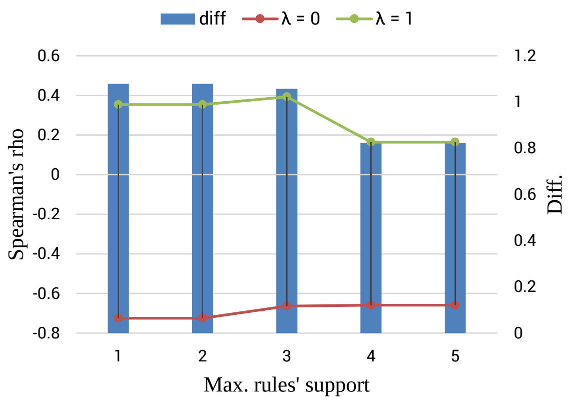
Hybrid quality function μ fixes this issues by directly looking at the quality of predicted facts, which is achieved by feedback from the embedding models. We verify this hypothesis on *FB15K*, *Wiki44K* and *IMDB* datasets. We also extract rules of form the $r: \ h(X,Z) \leftarrow p(X,Y), q(Y,Z)$ from $\mathcal{G}$, with $conf(r,\mathcal{G}) \geq 0.1$ and $r\text{-}supp(r,\mathcal{G}) \leq k$,



(A) FB15K



(B) Wiki44K



(C) IMDB

FIGURE 6.5: Spearman's rho of top rules ranked by confidence and the hybrid quality function, with limit on rules' support.

where k indicates the maximum support of rules to be extracted, we try different values of $k \in \{1, 2, 3, 4, 5\}$. These rules are then ranked using standard confidence and our hybrid quality function μ. In addition, since we hypothesize that the confidence does not give strong information about the rules in this case, we use $\mu = \mu_2(\lambda = 1)$, where the hybrid quality function relies only on feedback from the embedding models. To compute μ_2, we use the best embedding models for each dataset: SSP for *FB15K*, *Wiki44K* and TransE for *IMDB*.

To evaluate the quality of the 2 ranked rule lists, we measure the Spearman's rank correlation coefficient (*Spearman's rho*, i.e. Pearson correlation applied to rank) of rules' confidence or hybrid quality with their $pred\_prec_{CW}$. Figure 6.5 shows the Spearman's rho of the 2 ranked lists and the difference between them with different values of rules' support limit k. The hybrid quality function outperforms standard confidence on 3 datasets, and the results are more visible for lower-support rules.

top-k	**FB15K**						**Wiki44K**					
	AMIE-PCA		**AMIE-Conf**		**RuLES**		**AMIE-PCA**		**AMIE-Conf**		**RuLES**	
	Facts	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*
20	1029	0.28	82	0.63	44	1.00	185	0.73	91	0.95	3291	0.98
50	1716	0.43	190	0.74	186	0.92	47099	0.10	3594	0.95	6154	0.88
100	3085	0.65	255	0.78	539	0.80	56831	0.20	13870	0.83	13253	0.82
200	10586	0.62	1210	0.83	1205	0.88	82288	0.39	19538	0.72	20408	0.73
500	40050	0.51	2702	0.75	7124	0.95	219264	0.35	124836	0.23	128256	0.48

TABLE 6.3: Prediction precision of the *top-k* rules generated by RuLES and AMIE.

top-k	**Family**			
	NeuralLP		**Conf-TransE**	
	Facts	*Prec.*	*Facts*	*Prec.*
10	3709	0.72	4201	0.68
20	8821	0.53	6957	0.72
30	11337	0.49	9368	0.71
40	14662	0.46	11502	0.72
50	18768	0.40	14547	0.62

TABLE 6.4: Prediction precision of the *top-k* rules generated by NeuralLP and RuLES.

6.3 Horn Rule Learning

In this experiment, we compare the Conf-SSP configuration of RuLES (with embedding weight $\lambda = 0.3$) against the state-of-art Horn rule learning system AMIE. We used the default AMIE-PCA configuration with $conf_{pca}$ and AMIE-Conf with *conf* measures respectively. For a fair comparison, we set the two configurations of AMIE and our system to generate rules with at most three positive atoms in the body and filtered them based on minimum confidence of 0.1, head coverage of 0.01 and rule support of 10 in case of FB15K and 2 in case of Wiki44K. We then filtered out all rules with $conf(r, \mathcal{G}) = 1$, as they do not produce any predictions.

Table 6.3 shows the number of facts (see the *Facts* column) predicted by the set R of *top-k* rules in the described settings and their prediction precision $pred\_prec_{OW}(R)$ (see the *Prec.* column). The size of the random sample outside $\mathcal{G}^i_{appr}$ is 20. We can observe that on FB15K, RuLES consistently outperforms both AMIE configurations. The *top-20* rules have the highest precision difference (outperforming AMIE-PCA and AMIE-Conf by 72% and 37% respectively). This is explained by the fact that the hybrid embedding quality penalizes rules with higher number of false predictions. For Wiki44K, RuLES is capable of achieving better precision in most of the cases. Notably, for the *top-20* rules RuLES predicted significantly more facts then competitors yet with a high precision.

In table 6.4, we compare RuLES with the recently developed NeuralLP system [55].

r^1:	$\mathit{nationality}(X, Y) \leftarrow \mathit{graduated\_from}(X, Z), \mathit{in\_country}(Z, Y), \textbf{not } \mathit{research\_uni}(Z)$
r^2:	$\mathit{scriptwriter\_of}(X, Y) \leftarrow \mathit{preceded\_by}(X, Z), \mathit{scriptwriter\_of}(Z, Y), \textbf{not } \mathit{tv\_series}(Z)$
r^3:	$\mathit{noble\_family}(X, Y) \leftarrow \mathit{spouse}(X, Z), \mathit{noble\_family}(Z, Y), \textbf{not } \mathit{chinese\_dynasties}(Y)$

Table 6.5: Example rules with exception generated by RuLES.

For this we used the Family dataset offered by the authors[4] with 28K facts over 3K entities and 12 relations. Starting from the *top-20* rules RuLES is capable of achieving significantly better precision. For the *top-10* rules the precision of NeuralLP is slightly better, but RuLES predicts many more facts.

6.4 RuLES for Exception-Aware Rule Learning

In this experiment, we aim at evaluating the effectiveness of RuLES for learning exception-aware rules. First, consider in Table 6.5 examples of such rules learned by RuLES over Wiki44K dataset. The first rule r^1 says that a person is a citizen of the country where his alma mater is located, unless it is a research institution, since most researchers in universities are foreigners. The second rule r^2 states that the scriptwriter of some artistic work is also the scriptwriter of its sequel unless it is a TV series, which actually reflects the common practice of having several screenwriters for different seasons. Additionally, r^3 encodes that someone belonged to a noble family if his/her spouse is also from the same noble family, excluding the Chinese dynasties.

To quantify the quality of RuLES in learning non-monotonic rules, we compare the Conf-SSP configuration of RuLES (with embedding weight $\lambda = 0.3$) with RUMIS [44], which is a non-monotonic rule revision system which finds exceptions by minimizing the conflicts between the induced rules. RUMIS learns rules of the form $r:\ h(X, Z) \leftarrow p(X, Y), q(Y, Z), \mathit{not}\ E$, where E is either $e(X, Z)$ or $\mathit{type}(X, t)$ with $t \in \mathcal{C}_{\mathcal{G}}$. For a fair comparison we restricted RuLES to learn rules of the same form. We configured both systems setting the minimum rule support threshold to 10 and exception confidence for RuLES to 0.05. To enable the systems to learn rules with exceptions of the form $\mathit{type}(X, t)$, we enriched the KGs with *type* information from the original Freebase and Wikidata graphs.

From the list of mined rules from our system, we keep only rules having some exceptions. If there exist more than 1 such rule having the same positive part, we keep only the one with the highest hybrid quality. In addition, to ensure that the added exception is meaningful, from the list of generated rules from both systems, we keep only rules, whose positive part has standard confidence less than 0.8.

[4]https://github.com/fanyangxyz/Neural-LP

	FB15K				**Wiki44K**			
top-k	**RUMIS**		**RuLES**		**RUMIS**		**RuLES**	
	Facts	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*
20	672	0.95	34	0.97	5844	0.93	5640	0.93
50	1797	0.94	158	0.99	8585	0.83	13333	0.84
100	2672	0.94	434	0.99	21081	0.76	25265	0.81
200	4103	0.87	1155	0.96	50957	0.51	43677	0.67
500	13439	0.76	5466	0.90	–	–	–	–

Table 6.6: Prediction precision for the *top-k* rules learned by RUMIS and RuLES.

	FB15K				**Wiki44K**			
top-k	**RUMIS**		**RuLES**		**RUMIS**		**RuLES**	
	Facts	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*	*Facts*	*Prec.*
20	76	0.70	111	0.68	63	0.47	81	0.94
50	126	0.51	435	0.74	191	0.28	611	0.69
100	183	0.43	680	0.76	543	0.49	1698	0.79
200	310	0.30	1112	0.87	4861	0.40	3175	0.80
500	1155	0.53	3760	0.59	–	–	–	–

Table 6.7: Revision precision for the *top-k* rules learned by RUMIS and RuLES.

Table 6.6 reports the number of predictions produced by a rule set R of *top-k* non-monotonic rules learned by both systems as well as their precision $pred\_prec_{OW}(R)$ with a sample of 20 prediction outside $\mathcal{G}^i_{appr}$. The results show that RuLES consistently outperforms RUMIS on both datasets. For Wiki44K, and $k \in \{50, 100\}$, the *top-k* rules produced by RuLES predicted more facts than those induced by the competitor achieving higher overall precision. Regarding the number of predictions, the converse holds for the FB15K KG; however, the rules learned by RuLES are still more accurate.

To evaluate the quality of the chosen exceptions, we compare the $rev\_prec_{OW}(R)$ with a sample of 20 predictions. Observe that in Table 6.7, rules induced by RuLES prevented the generation of more facts than RUMIS. In all of the cases apart from *top-20* for FB15K, our system managed to remove a larger fraction of erroneous predictions. For Wiki44K, RuLES consistently performs twice as good as RUMIS. In conclusion, the guidance from the embedding model exploited in our system gives us hints on which among the possible exception candidates likely correspond to noise.

6.5 Summary

Conducted experiments demonstrate a highly positive impact of the integration of an embedding model on the quality of learned rules. By using an good embedding model, our mining algorithm can extract rules with better predictions' quality, and also capture more meaningful exceptions in comparison with standard methods.

Chapter 7

Conclusion

7.1 Thesis Summary

Knowledge graphs capture information about the real world and play an important role in various information systems. However, due to the automatic construction, KGs are usually incomplete so that a large amount of research effort has been committed to address this problem.

Solutions for the KGs completion problem often fall into the two categories: *embedding-based* approach and *rule-based* approach. In this thesis, we have presented a hybrid approach for learning non-monotonic rules from KGs that dynamically exploits feedback from a precomputed embedding model. Our method is general in that any embedding model can be utilized including text-enhanced ones, which indirectly allows us to harness unstructured web sources for rule learning. We evaluated our approach with various configurations on real-world datasets and observed significant improvements over state-of-the-art rule learning systems.

7.2 Future Research Directions

Regarding our approach, several prominent research directions could be considered to further advance this work. In what follows, we discuss the most important ones.

7.2.1 Incorporation of other embedding models

Currently, we are supporting only 3 embedding models namely TransE, HolE and SSP. While TransE and HolE models work based only on the given KG, SSP model provides the integration of entity descriptions. However, a variety of other embedding models exist that can even support different types of external data such as temporal knowledge and general Web-texts. These embedding models are more advanced and might produce better predictions than the ones that we exploited. Hence, injecting better embedding models is a promising direction for further facilitation of practicability of our approach.

7.2.2 Incorporation of multiple heterogeneous external sources

In this work, we currently rely on only a single external source, which is an embedding model. However, other types of external sources could be likewise taken into account to improve the quality of our approach. Technically, our external quality function μ_2 is computed based on facts predicted by a given rule. One idea regarding the extension of our rule learning method is to exploit different external sources depending on the rule head predicate to quantify the rule quality. Here, several possibilities exist:

- First, since the performance of different embedding models varies for different predicates [5], we can use the embedding models to assess only selected relations on which they perform well.

- Second, by separating relations, we can even combine the unsupervised and supervised learning settings. For instance, besides using embedding models for the relations on which they do perform well, we can also involve human judgement for other relations that are problematic for embedding models. Note that, naturally when using human inspection, the probabilistic fact assessment function f would return binary values $\{0, 1\}$.

- Third, different external sources could be incorporated for learning rules from general KGs, which contain relations from various domains. In particular, when learning rules from a KG with different predicates, the external quality function μ_2 could be computed per predicate independently by using models trained with different domain-specific external data. For example, for movie-related predicates, IMDB could be invoked, whereas for sport-related ones, we can use data from dedicated sport websites. However, the challenge here is to figure out how to integrate these external sources to extend our hybrid probabilistic function f.

Incorporating many external sources can certainly improve the quality of the learned rules. However, this likely comes with the price of the increased running time and possibly memory overload. So, finding the tradeoff between these two aspects is an important point for further research.

7.2.3 Auto-tuning embedding weight value

As our experiments demonstrate, the value of the embedding weight λ is crucial for achieving good results with our approach. Within this thesis, the best value of λ has been empirically determined for every considered embedding model. Manually choosing a suitable value of λ for an arbitrary dataset and embedding model requires a lot of human efforts. Hence, a relevant extension of our work concerns the development of an automatic procedure for computing the optimal weight λ. Naive strategy here would be to automate the procedure executed in Experiment 6.2.1. However, other advanced and less costly solutions should exist.

Additionally, following the idea of treating every rule predicate individually as mentioned before, one can even automatically learn different values of λ per predicate.

7.2.4 Incorporation of various classical rule measures

In this thesis, we considered *standard confidence*, *PCA confidence* and *conviction* as classical rule measures. However, there are more sophisticated rule measures μ_1 that could be also studied empirically, e.g., *soft confidence* [50], *completeness-aware* measures [43], *RC confidence* [56], etc. However, the challenge here is to choose the metric which is the most suitable for the dataset. For instance, *PCA confidence* only performs well if the KG follows the Partial Closed-world Assumption, *soft confidence* requires *type* information to be available in the KG, and *completeness-aware* measures ask for extra (in-)completeness meta-data about the KG in form of cardinality statements.

7.2.5 Iterative improvement of rules and embedding model

As mentioned in Section 2.3, a variety of approaches introducing embedding models that make use of prespecified rules have been provided in the literature. So, if we plug these embedding models in our approach, an iterative improvement of rules and embedding models could be developed as sketched follows. We start with some rules extracted from a rule mining system (e.g., our system with $\lambda = 0$). Then, these initial rules are used to train the embedding model. Repeatedly, rules are then revised using our approach and rely on the trained model, which results in an iterative procedure.

7.2.6 Inference of facts from intersection of rules and external source

Currently, the main output of our approach is rules. In highly incomplete KGs, the precision of rules is normally rather low, which leads to the incorrectness of inferred facts. To tackle this issue, one could learn more complex rules (e.g., with more atoms), which could improve the likelihood of predicted facts by pruning low quality branches. However, overcoming the barrier of language bias is not easy in this context.

An alternative solution is to infer facts based on the intersection of predictions between the rules and the external source. For example, we could predict a fact as being true only if the external source gives it a probability value f above a pre-defined threshold, and at the same time, the fact is binded with some rule that is confident enough. The main drawback of this method is that, while precision of predicted facts obviously increases, their recall could decrease significantly. Hence, computing the optimal thresholds is the main challenge of this research direction, which is quite interesting to study.

Bibliography

[1] Google's KG. http://www.google.co.uk/insidesearch/features/search/knowledge.html.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.

[3] Eric Auriol, Michel Manago, and Jérôme Guiot-Dorel. Cassiopée: Fehlerdiagnose von CFM 56-3 triebwerken für boing 737 flugzeuge. *KI*, 10(1):47–53, 1996.

[4] Paulo J. Azevedo and Alípio M. Jorge. *Comparing Rule Measures for Predictive Association Rules*, pages 510–517. 2007.

[5] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of NIPS*, pages 2787–2795, 2013.

[6] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 301–306, 2011.

[7] Yang Chen, Sean Goldberg, Daisy Zhe Wang, and Soumitra Siddharth Johri. Ontological pathfinding. In *SIGMOD*, 2016.

[8] William W. Cohen. Tensorlog: A differentiable deductive database. *CoRR*, abs/1605.06523, 2016.

[9] Domenico Corapi, Daniel Sykes, Katsumi Inoue, and Alessandra Russo. Probabilistic rule leadblp:conf/nips/yangyc17rning in nonmonotonic domains. In *CLIMA*, 2011.

[10] Fariz Darari, Werner Nutt, Giuseppe Pirrò, and Simon Razniewski. Completeness statements about RDF data sources and their use for query answering. In *ISWC*, 2013.

[11] Johannes Fürnkranz and Tomás Kliegr. A brief overview of rule learning. In *RuleML*, pages 54–69, 2015.

[12] Mohamed H Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. Exception-enriched rule learning from knowledge graphs. In *Proceedings of ISWC*, pages 234–251, 2016.

[13] Luis Galarraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730, 2015.

[14] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 413–422, 2013.

[15] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080. MIT Press, 1988.

[16] Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *CoRR*, abs/1301.3485, 2013.

[17] Bart Goethals and Jan Van den Bussche. Relational association rules: Getting warmer. In *PDD*, 2002.

[18] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *EMNLP*, 2016.

[19] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Knowledge graph embedding with iterative guidance from soft rules. *CoRR*, abs/1711.11231, 2017.

[20] Seunghyun Im, Zbigniew W. Ras, and Hanna Wasyluk. Action rule discovery from incomplete data. *Knowl. Inf. Syst.*, 25(1):21–33, 2010.

[21] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. A latent factor model for highly multi-relational data. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, pages 3167–3175, 2012.

[22] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. Towards time-aware knowledge graph completion. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1715–1724, 2016.

[23] Sotiris Kotsiantis and Dimitris Kanellopoulos. Association rules mining: A recent overview. *GESTS Int. Trans. on CS and Eng.*, 32(1):71–82, 2006.

[24] Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. 1999.

[25] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2181–2187, 2015.

[26] John W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.

[27] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovering frequent episodes in sequences. In *KDD-95*, 1995.

[28] Pasquale Minervini, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Leveraging the schema in latent factor models for knowledge graph completion. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 327–332, 2016.

[29] Tran Duc Minh, Claudia d'Amato, Binh Thanh Nguyen, and Andrea G. B. Tettamanzi. Comparing rule evaluation metrics for the evolutionary discovery of multi-relational association rules in the semantic web. In *EuroGP*, 2018.

[30] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-Ending Learning. In *AAAI*, 2015.

[31] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. Stranse: a novel embedding model of entities and relationships in knowledge bases. *CoRR*, abs/1606.08140, 2016.

[32] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[33] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, 2016.

[34] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[35] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[36] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248. 1991.

[37] Luc De Raedt, Anton Dries, Ingo Thon, Guy Van den Broeck, and Mathias Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *IJCAI*, pages 1835–1843. AAAI Press, 2015.

[38] Luc De Raedt and Ingo Thon. Probabilistic rule learning. In *ILP*, 2010.

[39] Zbigniew W. Ras and Alicja Wieczorkowska. Action-rules: How to increase profit of a company. In *PKDD*, 2000.

[40] Pushpendre Rastogi, Adam Poliak, and Benjamin Van Durme. Training relation embeddings under logical constraints. In *KG4IR*, 2017.

[41] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934. 2013.

[42] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *Proceedings of WWW*, pages 697–706, 2007.

[43] Thomas Pellissier Tanon, Daria Stepanova, Simon Razniewski, Paramita Mirza, and Gerhard Weikum. Completeness-aware rule learning from knowledge graphs. In *ISWC*, 2017.

[44] Hai Dang Tran, Daria Stepanova, Mohamed H. Gad-elrab, Francesca A Lisi, and Gerhard Weikum. Towards nonmonotonic relational learning from knowledge graphs. *26th ILP*, 2016.

[45] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357, 2016.

[46] Denny Vrandecic and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *CACM*, 57(10):78–85, 2014.

[47] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743, 2017.

[48] Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *IJCAI*, 2015.

[49] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119, 2014.

[50] Zhichun Wang and Juan-Zi Li. Rdf2rules: Learning rules from RDF knowledge bases by mining frequent predicate cycles. *CoRR*, abs/1512.07734, 2015.

[51] Zhigang Wang and Juan-Zi Li. Text-enhanced representation learning for knowledge graph. In *IJCAI*, 2016.

[52] Janusz Wojtusiak. Rule learning in healthcare and health services research. In *Machine Learning in Healthcare Informatics*, pages 131–145. 2014.

[53] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. SSP: semantic space projection for knowledge graph embedding with text descriptions. In *AAAI*, 2017.

[54] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2014.

[55] Fan Yang, Zhilin Yang, and William W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, pages 2316–2325, 2017.

[56] Kaja Zupanc and Jesse Davis. Estimating rule quality for knowledge base completion with the relationship between coverage assumption. In *The Web Conference 2018*, 2018.