REGNUM: Generating Logical Rules with Numerical Predicates in Knowledge Graphs

Armita Khajeh Nassiri[1(✉)], Nathalie Pernelle[1,2(✉)], and Fatiha Saïs[1(✉)]

[1] LISN, CNRS (UMR 9015), Paris Saclay University, 91405 Orsay, France
{armita.khajehnassiri,nathalie.pernelle,fatiha.sais}@lri.fr
[2] LIPN, CNRS (UMR 7030), University Sorbonne Paris Nord, Paris, France

Abstract. Mining logical rules from a knowledge graph (KG) can reveal useful patterns for predicting facts, curating the KG, and identifying trends. However, many rule mining systems face challenges when working with numerical data because numerical predicates can take a large number of values, leading to a huge search space. In this work, we present REGNUM, a system that addresses this issue by generating rules with numerical constraints. REGNUM extends the body of rules mined from a KG by using supervised discretization of numerical values with decision trees to increase the confidence of the rules without sacrificing significance. Our experimental results show that the numerical rules have a higher overall quality than the parent rules and are effective at making better predictions.

Keywords: Rule Mining · Numerical Predicates · KG Completion

1 Introduction

Knowledge graphs (KG) are large collections of facts about the world or a specific domain stored in a machine-readable format. There is no surprise that these large KGs incorporate different forms of knowledge within them. There has been a tremendous amount of work in the literature trying to capture and mine this knowledge. One such line of work is mining logical rules in KGs. These rules can serve to complete the KG, detect erroneous data, or uncover the knowledge that is not explicitly stated in the ontology. For instance, if we don't know one's place of residence, we could infer that they live in the same place as their spouse. In addition, rules can serve to debug the KG. If the spouse of someone lives in a different city, then this may indicate a problem. Finally, rules are useful in downstream applications such as fact-checking [1], ontology alignment [11,16], or predicting completeness [9]. Furthermore, such rules have the advantage of being explainable, interpretable, and transferable to unseen entities.

One challenge in finding logical rules in KGs lies in the exponential size of the search space, which varies depending on the considered language bias. To address this issue, several recent approaches have relied on sampling or approximate confidence calculations [10,30], and [23]. Another common technique [21,23,30], from standard

C. Pesquita et al. (Eds.): ESWC 2023, LNCS 13870, pp. 139–155, 2023.
https://doi.org/10.1007/978-3-031-33455-9_9

inductive logic programming (ILP), is to mine not all rules but only enough rules to cover the positive examples. Likewise, this speeds up the computation but may lose many interesting rules. While existing rule mining approaches are effective, they are unable to consider rules that involve constraints defined on numerical values. We believe that such constraints can be highly relevant in domains such as finance, public health, or life science as they can help uncover useful information, such as the increased likelihood of a patient with heart disease having taken mood stabilizers for over five years. In this paper, we take a step forward toward incorporating numerical predicates into logical rules in a way that supports knowledge expansion. We focus on constraints that express the membership or non-membership to a value interval. One approach to implementing these constraints would be to discretize the values of the numerical predicates in a pre-processing step. However, this would not provide with relevant constraints for each rule and yield to loss of information. On the other hand, calculating constraints while the rule mining technique explores the search space (as it generalizes or refines the rule) results in having to re-calculate the interval at each step, making the approach time-consuming over large graphs. To this end, we propose a novel method that involves two steps: first, to obtain First-order logic (FOL) rules using existing efficient rule mining tools, and second, to enrich the rules with numerical predicates and constraints. For this second step, we consider the problem as a classification problem to obtain the intervals based on the correct and incorrect predictions of the rule, guided by the quality of the rules. The main contributions of our paper are:

- REGNUM is a novel approach that enhances the expressiveness of the rules generated by a rule mining system by incorporating numerical constraints expressed through value intervals. To our knowledge, REGNUM is the first approach to utilize intervals in rules mined from large RDF graphs.
- REGNUM efficiently selects intervals that increase the rule's confidence using existing supervised discretization techniques that best distinguish the correct and incorrect predictions made by the rule.
- Since some value intervals can be too specific to lead to relevant rules, REGNUM considers both the membership and the non-membership of a value to an interval to offer more possibilities of generating a rule with high quality.
- The experimental evaluation shows that the numerical rules generated by REGNUM using the rules provided by two state-of-the-art rule mining systems, AMIE and AnyBURL, have a higher overall quality score and can potentially improve prediction results.

2 Related Work

Rule mining over a dataset has received a lot of attention from researchers, resulting in many published works on the subject.

Association Rule Mining. Association rule mining (ARM) is a widely used data mining technique that identifies frequent patterns among items and transactions based on a minimum number of observations. It typically generates if-then patterns, represented by association rules $X \rightarrow Y$, indicating that the presence of X suggests the presence of Y in the same transaction. However, ARM faces challenges when dealing with

numerical attributes since the values of these attributes rarely repeat themselves. To address this, a special type of association rule called quantitative association rules have been developed, which involves at least one numerical attribute in the rule, such as $(25 < age < 40) \wedge (3K < salary < 5K) \rightarrow (120K < loan < 200K)$. Quantitative association rule mining (QARM) can be achieved through different strategies, including discretization-based approaches such as pre-processing steps to partition numerical data [27] or statistical analysis of variables and distribution of the numerical variables [2], and optimization-based approaches where numeric attributes are optimized during the mining process, for instance with the use of genetic algorithms [15,20,26].

Nevertheless, these patterns or dependencies are restricted to single variables and are different from the logical rules relevant to complex relationships present in knowledge graphs.

Inductive Logic Programming (ILP). Inductive logic programming (ILP) systems can automatically find rules based on positive and negative examples. For example, WARMR [7] extends the Apriori algorithm to mine association rules in multiple relations. Other ILP-based approaches, such as DL-Learner [5], focus on learning expressive concept definitions, including numerical constraints. However, ILP is not suitable for the open world assumption (OWA) in large KGs, where counter-examples are not declared, and missing information cannot be treated as negative but rather unknown.

FOL Rule on Knowledge Graphs. Finding logical rules on large knowledge graphs (KGs) has been addressed in several works that use specific language biases, pruning criteria, and optimization strategies to scale the rule-mining process.

AMIE [17] is a state-of-the-art rule mining system for KGs. It is fast and exhaustive, i.e., it mines all connected and closed rules given thresholds defined on quality measures (e.g., confidence and head coverage) and a specified maximum number of atoms. AMIE can discover rules that involve constants (e.g., `age(x,53)`). However, these rules can be too specific and not interesting when it comes to these predicates. RuDiK [24] proposes a non-exhaustive approach to mine logical rules that are more expressive. RuDiK can predict the absence of a fact and allows us to perform comparisons beyond equality by using relationships from the set $rel \in \{<, \leq, \neq, \geq, >\}$. An example of such rules would be: $R_1 : p_1(x, v_0) \wedge p_2(y, v_1) \wedge v_0 > v_1 \Rightarrow p_3(x, y)$, where v_0 and v_1 are values from the KG itself, not thresholds.

Rule mining systems, such as AnyBURL [19], focus only on rules based on graph paths. AnyBURL is a bottom-up approach that starts with sampling specific paths and uses generalization techniques to expand it such that the obtained rule has a high confidence. An extension of AnyBURL [18] uses reinforcement learning to sample better paths from the start. The advantage of these systems is that they are any-time, meaning they can trade time for rule quality and quantity. However, like AMIE, AnyBURL is not able to find interesting rules with numerical predicates and can only consider them as constants.

Another family of rule mining systems is differentiable rule-based inference methods such as NeuralLP [29]. This body of work maps each entity to a vector and each relation to an adjacency matrix. DRUM [25] proposes changes to the NeuralLP to support variable-length rules. Another extension, NeuralLP-num [28], can learn rules involving numerical features. Like RuDiK, these rules can involve negative atoms or make pair-wise comparisons between numerical values of different atoms in the rules

(e.g., R_1). Furthermore, the rules produced by NeuralLP-num can also include classification operators, which are sigmoid functions over numerical values of atoms with numerical predicates in the rule. For example, a rule with a classification operator could be of the form $f\{y_1, y_2 : p_1(X, y_1), p_2(X, y_2)\} > 0.5 \wedge p_3(X, Z) \Rightarrow p_4(X, Z)$ where f is the sigmoid function and p_1 and p_2 are numerical predicates.

To the best of our knowledge, RuDiK and NeuralLP-num are the only works in the literature that can mine interesting rules with numerical predicates on large KGs. However, both techniques are limited to using numerical values from the knowledge graph and applying functions or comparisons between them. They are unable to discover numerical intervals or thresholds as constraints to enhance the quality of the rules and derive additional knowledge. To fill this gap, REGNUM has been developed to incorporate such constraints into the rules.

3 Preliminaries

This section presents the definitions and notations used in the rest of the paper.

Definition 1. RDF Knowledge Graph. *In an RDF knowledge graph $\mathcal{G}$, a collection of facts is represented as triples of the form $\{$(subject, predicate, object) $\mid$ subject $\in \mathcal{I}$, predicate $\in \mathcal{P}$, object $\in \mathcal{I} \cup \mathcal{L}\}$ where the set of entities is denoted as $\mathcal{I}$, the set of predicates is denoted as $\mathcal{P}$, and the set of literals (such as numbers and strings) is denoted as $\mathcal{L}$. Additionally, we define $\mathcal{P}_{num}$ as the subset of predicates whose range consists solely of numerical values.*

Definition 2. Atom. *An atom is a basic well-formed first-order logic formula of the form $p(X, Y)$, where p is a predicate*[1] *and X, Y are either constants or variables*[2]*. If an atom's arguments are constants, the atom is said to be "grounded" and can be treated as a fact.*

Definition 3. (Horn) Rule. *A rule $r : \mathcal{B} \Rightarrow H$ is a first-order logic formula where the body $\mathcal{B}$ is a conjunction of atoms $B_1, ..., B_n$ and the head H is a single atom. A rule is closed if every variable appears at least twice in the rule. Two atoms are connected if they share at least one variable. A rule is connected if all atoms in the rule are transitively connected.*

Definition 4. Prediction of a Rule. *Given a rule $r : \mathcal{B} \Rightarrow H$ and a substitution of the rule $\sigma(r)$, $\sigma(H)$ is a prediction for r if all the atoms of $\sigma(\mathcal{B})$ belong to the knowledge graph $\mathcal{G}$. A prediction is correct if $\sigma(H) \in \mathcal{G}$.*

For a rule $r : \mathcal{B} \Rightarrow H$, we have the following quality measures as defined in [17]. In the absence of identity links (i.e., *owl:sameAs*), we assume that the Unique Name Assumption (UNA) is fulfilled. If identity links exist, a pre-processing step is required to compute the quality measures and functionality score accurately.

Definition 5. Support. *The support $supp(r) := |\{(x, y) : \mathcal{B} \wedge H(x, y)\}|$ measures the number of correct predictions made by the rule.*

[1] Membership to a class, can also be represented with a binary predicate, i.e., type(X, Y).

[2] Variables are represented using lowercase letters whereas capitalized letters denote constants.

Definition 6. Head Coverage. *Head coverage represents the proportion of instantiations of the head atom that are correctly predicted by the rule.*

$$hc(r) = \frac{supp(r)}{|\{(x,y) : H(x,y) \in \mathcal{G}\}|}$$

In order to calculate the confidence of a rule, counter-examples are necessary. Knowledge graphs are based on the open world assumption (OWA), meaning they only contain positive examples, and missing facts are not necessarily false. We adhere to the Partial Completeness Assumption (PCA) to account for counter-examples.

Definition 7. Functionality Score. *The functionality score of a predicate is a value between 0 and 1 that measures the ratio of subjects that the property is related to in $\mathcal{G}$ to the total number of triples with that predicate. The inverse functionality score $ifun(p)$ is the functionality score for the inverse of the predicate p.*

$$fun(p) := \frac{|\{x : \exists y : p(x,y) \in \mathcal{G}\}|}{|\{(x,y) : p(x,y) \in \mathcal{G}\}|}$$

Under PCA, if a fact $p(x,y) \in \mathcal{G}$ and if $fun(p) > ifun(p)$, then no other fact for x holding with the predicate p is correct and can be considered as a counter-example (i.e., $p(x,y') \notin \mathcal{G}$). On the other hand, if $ifun(p) > fun(p)$, then all $p(x',y) \notin \mathcal{G}$.

Definition 8. PCA confidence. *The PCA confidence of the rule r measures the precision of the rule under the PCA, i.e., the ratio of correct predictions or support to the total number of predictions made by the rule. More precisely, if $fun(H) > ifun(H)$,*

$$pca\_conf(r) = \frac{supp(r)}{|\{(x,y) : \exists y' : \mathcal{B} \wedge H(x,y')\}|}$$

Based on definition of counter-examples under PCA, if $ifun(H) > fun(H)$, then the denominator, namely PCA body size, becomes $|\{(x,y) : \exists x' : \mathcal{B} \wedge H(x',y)\}|$ in the above equation.

4 REGNUM

In this section, we describe REGNUM, a system that automatically enriches the connected and closed rules mined on a given knowledge graph, regardless of the method used to mine them, with numerical predicates by constraining the introduced numerical values to specified intervals. REGNUM aims to enhance the PCA confidence in the considered rules while ensuring that the rules do not become overly specific.

4.1 Problem Statement

This approach aims to mine numerical rules that are defined as follows:

Definition 9. Numerical Rule. *A numerical rule is a first-order logic formula of the form: $\mathcal{B} \wedge C \Rightarrow H$, where B is a conjunction of atoms of the KG and where the range values of the numerical atoms of B can be constrained using C, conjunction (resp. a disjunction) of atoms that express their membership (resp. their non-membership) to an interval $[inf, sup]$.*

Example 1. Here are two examples of numerical rules with constraints defined on numerical predicates:
$r_1 : \text{worksIn}(x, y) \wedge \text{hasPopulation}(y, w) \wedge w \in [1000, 5500] \wedge \text{hasHusband}(x, z) \Rightarrow \text{worksIn}(z, y)$
$r_2 : \text{worksIn}(x, y) \wedge \text{hasHusband}(x, z) \wedge \text{age}(z, a) \wedge \text{hasPopulation}(y, w) \wedge (w \notin [1000, 5500] \vee a \notin [50, \infty)) \Rightarrow \text{worksIn}(z, y)$

Generating the complete set of numerical rules that fulfill quality measure thresholds (e.g., $minHC$ and $minconf$ in [17]) can be very time-consuming. This is because the intervals used to constrain the range of numerical predicates must be recalculated each time a rule is generalized or refined as the search space is explored. This ensures that the constraints applied to the rule are appropriate for the updated rule.

To overcome this issue, we propose an approach that builds on the shoulders of the rules mined by an existing rule-mining technique (i.e., parent rules) and expands the body of these rules through an enrichment process to generate numerical rules. More precisely,

- A numerical rule is considered relevant if it improves the PCA confidence of its parent rule by at least a $marginC$ without its head coverage decreasing more than $marginHC$. This criterion guarantees that the rule has higher PCA confidence than its parent rule while preventing over-fitting the KG.
- The enrichment process of a parent rule is driven by considering diverse sets of numerical predicates. This means that once a numerical rule involving a particular numerical predicate p is relevant in a constraint that involves n numerical atoms, the approach will not consider larger sets of atoms that involve p.
- The search strategy to obtain constraints on the numerical predicates relies on tree-based algorithms.

4.2 Rule Enrichment with Numerical Predicates Algorithms

Given a knowledge graph $\mathcal{G}$, a set of closed rules $\mathcal{R}$ mined from $\mathcal{G}$, called *parent rules*, and thresholds $marginC$ and $marginHC$ as introduced in Sect. 4.1, our approach REGNUM is able to enrich the parent rules in $\mathcal{R}$ to obtain relevant and diverse numerical rules. The algorithm we describe in Algorithm 1 performs the following steps.

(1) Pre-processing Step [line 1 in Algorithm 1]. As a pre-processing step, we first identify the set of numerical predicates, denoted as $\mathcal{P}_{num}$ in $\mathcal{G}$. We use the domain and range definition axioms if they are available in the ontology, and if not, we discover them by considering the range of values they take. We also compute the functionality score, as defined in Definition 7, for all predicates in the head of the parent rules $\mathcal{R}$.

Algorithm 1: REGNUM

Input:
- $\mathcal{G}$: knowledge graph
- $\mathcal{R}$: set of parent rules mined on $\mathcal{G}$
- $marginHC, marginC$: margins on head coverage and PCA confidence

Output: $\mathcal{E}$: set of enriched rules

1 Identify P_{num} and compute functionality degree of predicates P
2 $\mathcal{E} = \emptyset$
3 **foreach** $r : \mathcal{B} \Rightarrow H$ *in* $\mathcal{R}$ **do**
4 compute quality measure $hc(r)$ and $pca\_conf(r)$; compute $minHC$ and $minC$;
create an empty queue q_{atoms};
5 **for** $p_{num} \in P_{num}$ **do**
6 **if** $hc(p_{num}(x_i, x_{new}) \wedge \mathcal{B} \Rightarrow H) > minHC$ **then**
7 Enqueue $p_{num}(x_i, x_{new})$ in q_{atoms}
8
9 **end**
10 $ln = 1$ `// number of numerical atoms`
11 **while** $|q_{atoms}| > ln$ **do**
12 **for** B_{num} *created by combining* ln *atoms from* q_{atoms} **do**
13 $r_s : \mathcal{B}_{num} \wedge \mathcal{B} \Rightarrow H$
14 **if** $hc(r_s) > minHC$ **then**
15 $< X, Y > \leftarrow construct\_prediction\_classes(\mathcal{G}, r_s)$;
16 $r_{nodes} \leftarrow discretize(< X, Y >, minHC, minC)$;
17 **foreach** r_{node} *in* r_{nodes} **do**
18 **if** $hc(r_{node}) > minHC$ *and* $pca\_conf(r_{node}) > minC$ **then**
19 add $r_{node} \Rightarrow H$ to $\mathcal{E}$;
20
21 **end**
22
23 **end**
24 remove from q_{atoms} atoms that resulted in a numerical rule;
25 $ln+ = 1$
26 **end**
27 **end**
28 return $\mathcal{E}$;

Then, for each parent rule $r \in \mathcal{R}$, we proceed with the following steps.

(2) Computation of $minHC$ and $minC$ [line 4 in Algorithm 1]. A numerical rule obtained by enriching a parent rule r is considered relevant if its PCA confidence increases by at least $marginC$ and if its head coverage does not decrease by more than $marginHC$. We first query the KG to compute $pca\_conf(r)$ and $hc(r)$ if the rule mining system does not provide them, and we calculate the $minHC : (1 - marginHC) * hc(r)$ and $minC : (1 + marginC) * pca\_conf(r)$ that the enriched rule must satisfy in order to be considered relevant.

(3) Enqueue Numerical Atoms [lines 5–8 in Algorithm 1.] In this step, we find all possible numerical atoms that can enrich the parent rule and store them in q_{atoms}. We consider the set of all variables $vars = \{x_1, \ldots, x_n\}$ that appear in the rule r, and enqueue all the atoms $p_{num}(x_i, x_{new})$ with $x_i \in vars$, x_{new} is a new variable and such that the head coverage of $p_{num}(x_i, x_{new}) \wedge \mathcal{B} \Rightarrow H$ is greater than $minHC$. Otherwise, the atom is discarded since its conjunction with other atoms will only lead to lower head coverage due to monotonicity.

Example 2. Let r_1 : $\text{workPlace}(x_1, x_2) \Rightarrow \text{birthPlace}(x_1, x_2)$, be a parent rule. The atom involving the numerical predicate hasPopulation with variables x_1 will be pruned as the rule $\text{hasPopulation}(x_1, x_3) \wedge \text{workPlace}(x_1, x_2) \Rightarrow \text{birthPlace}(x_1, x_2)$ does not satisfy the $minHC$.

(4) Selection of Numerical Atoms and Generation of the Best Intervals [lines 11–20 in Algorithm 1]. Our objective is to identify relevant numerical rules that meet the quality measure requirements by utilizing the fewest numerical predicates. To construct these numerical rules, we iteratively search through the space of possible conjunctions of atoms in q_{atoms}. We begin with a single numerical atom ($ln = 1$) and continue until the queue q_{atoms} does not contain ln atoms or more. At iteration ln, we apply the following steps:

Enriching r with ln numerical atoms [lines 11–12 in Algorithm 1] We retrieve ln atoms from q_{atoms} and consider their conjunctions $\mathcal{B}_{num}$: $p_{num_1}(x_i, x_{n+1}) \wedge \ldots \wedge p_{num_l}(x_j, x_{n+l})$ to construct:

$$r_s : \mathcal{B}_{num} \wedge \mathcal{B} \Rightarrow H$$

We query the knowledge graph and proceed with the enrichment process only if r_s satisfies the $minHC$ as constraining the values of these numerical predicates will not satisfy $minHC$ either.

Example 3. At iteration 3, we can have r_s : $\text{worksIn}(x, y) \wedge \text{hasHusband}(x, z) \wedge \text{hasPopulation}(y, w) \wedge \text{age}(x, v) \wedge \text{hasRevenue}(x, u) \Rightarrow \text{worksIn}(z, y)$ involving three different numerical predicates hasPopulation, age and hasRevenue that satisfies the $minHC$.

Classification problem based on rule predictions [line 14 in Algorithm 1]. The rules r_s created in the previous step are used to classify the instantiations of $\mathcal{B}_{num}$: $p_{num_1}(x_i, x_{n+1}) \wedge \ldots \wedge p_{num_l}(x_j, x_{n+l})$ as correct or incorrect examples and define a binary classification problem.

In this classification step, we build a class A to represent the set of instantiations $(x_{n+1}, ..., x_{n+l})$ of the numerical values of B_{num} that lead to a correct prediction $H(x_a, x_b)$ for the rule r_s. The examples of A are defined as follows:

$$\{(x_{n+1}, \ldots, x_{n+l}) \mid \mathcal{B}(x_1, \ldots, x_n) \wedge \mathcal{B}_{num}(x_i, \ldots, x_j, x_{n+1}, \ldots, x_{n+l}) \wedge H(x_a, x_b)\}$$

Moreover, we build a class B to represent the set of instantiations $(x_{n+1}, ..., x_{n+l})$ of the numerical values of B_{num} that lead to an incorrect prediction for the rule r_s under

PCA if the $fun(H) > ifun(H)$, i.e., the predictions $H(x_a, x_b)$ such that $H(x_a, x_b) \notin \mathcal{G} \wedge \exists x'_b\ H(x_a, x'_b) \in \mathcal{G}$.

$$\{(x_{n+1}, \ldots, x_{n+l}) \mid \mathcal{B}(x_1, \ldots, x_n) \wedge \mathcal{B}_{num}(x_i, \ldots, x_j, x_{n+1}, \ldots, x_{n+l}) \ \wedge \exists\, x_{b'} H(x_a, x'_b)\}$$

If $ifun(H) > fun(H)$, we classify the instantiation as incorrect if a fact does not exist in the KG for the target object and if there exists at least one subject for this object.

We generate a data structure $< X, Y >$ that represents for each correct and incorrect prediction $H(x_a, x_b)$, the set of numerical values per numerical predicate of B_{num} (since the numerical predicates can be multi-valued), and the label Y. The correct and incorrect example values are retrieved from the KG through the queries defined for the label A and B.

Example 4. Let $r_2 : \text{worksIn}(x, y) \wedge \text{hasHusband}(x, z) \Rightarrow \text{worksIn}(z, y)$ be the considered parent rule. One possible refinement r_s of this rule to consider in step 2 would be: $\text{hasPopulation}(y, w) \wedge \text{worksIn}(x, y) \wedge \text{hasHusband}(x, z) \wedge \text{hasRevenue}(z, r) \Rightarrow \text{worksIn}(z, y)$ In this rule, the target variable is z since $ifun(\text{worksIn}) < fun(\text{worksIn})$. Consider the following facts in $\mathcal{G}$: $\{\text{worksIn}(Marie, Lyon), \text{worksIn}(Marie, Gordes), \text{worksIn}(Joe, Lyon), \text{hasPopulation}(Lyon, 513\ 000), \text{hasPopulation}(Gordes, 2\ 000), \text{hasHusband}(Marie, Joe), \text{hasRevenue}(Joe, 1500), \text{hasRevenue}(Joe, 800)\}$. The triple $\text{worksIn}(Joe, Lyon)$ is a correct prediction of r_s, and for the introduced numerical features $< \text{hasPopulation}_y, \text{hasRevenue}_z >$ the two sets of numerical values $(513000, 800)$, and $(513000, 1500)$ are examples that belong to class A. $\text{worksIn}(Joe, Gordes)$ is an incorrect prediction. The numerical values $(2000, 800)$ and $(2000, 1500)$ are examples that belongs to class B. If we do not know where Joe works, the possible instantiations of the numerical values do not belong to any class.

Constraining the numerical rule to intervals [lines 16–19 in Algorithm 1]. To obtain a set of rules for classes A and B defined by a rule r_s, we can discretize the values of the numerical predicates. For this purpose, different methods can be considered, including decision-tree-based approaches, e.g., CART [4], sequential covering approaches, e.g., RIPPER [6], or FURIA [14], QARM techniques introduced in Sect. 2, or other discretization techniques.

We aim to find the purest intervals that can effectively differentiate between examples in class A and class B. However, if we limit ourselves to constraints that only express interval membership for correct groundings of class A, the resulting rule may have low head coverage if the interval is too specific. On the other hand, if we exclude intervals that lead to incorrect predictions, we may overlook rules with high confidence that can enhance the accuracy of predictions in KG completion tasks.

Therefore, we consider both candidate rules. For instance, it is common for people to work in the city where they were born if that city has between 50,000 and 500,000 inhabitants. However, we can also consider a rule that excludes megacities with over 1,000,000 inhabitants.

Hence, we decided to employ a supervised method to discretize the continuous values of numerical predicates in $\mathcal{B}_{num}$ and keep track of the number of correct and

incorrect predictions falling in each interval to consider both membership and non-membership constraints. This method involves constructing a univariate CART Decision Tree (DT), where the numerical predicates serve as features. The DT is binary and built using impurity-based criteria, specifically entropy, as the splitting criteria.

The root of the tree corresponds to the numerical $r_s : \mathcal{B}_{num} \wedge C \ \wedge \ \mathcal{B} \Rightarrow H$ where C is initially empty. At each split, the instance space is divided into two subspaces by constraining the range values of one of the atoms in $\mathcal{B}_{num}$ to the split threshold and hence updating C. The rules at each child node r_{node} are created according to the rule of the parent node and the split made at that node.

More specifically, if a split is made using an atom $p(x, y)$ and a threshold α at node i, a membership constraint creates a rule for the left child by updating C with $\wedge\ y \in (-\infty, \alpha]$ and $\wedge\ y \in [\alpha, \infty)$ for the right child. A non-membership constraint, however, creates the rule for its left child by updating C with $\vee\ y \notin (-\infty, \alpha]$ for the left child and $\vee\ y \notin [\alpha, \infty)$ for the right child.

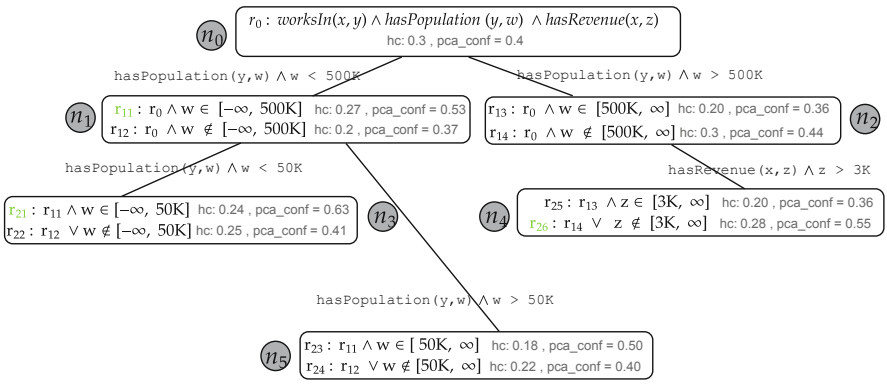

Fig. 1. Example of a part of the DT and the considered rules at each node

Example 5. Consider the parent rule $r : \text{worksIn}(x, y) \Rightarrow \text{livesIn}(x, y)$ and enriching the body of r with two predicates $r_0 : \text{worksIn}(x, y) \wedge \text{hasPopulation}(y, w) \wedge \text{hasRevenue}(x, z)$. Figure 1 depicts the construction of the rules at each node for a part of the tree to constrain the values of w and z with membership or non-membership. The $minHC = 0.23$ and the $minC = 0.5$. Node n_1 shows the inclusion and exclusion rules as well as their respective head coverage and PCA confidence constructed using the constraint $\text{hasPopulation}(y, w) \wedge w < 500K$.

Furthermore, we ensure that each node only contains the most concise rule. This means that if an atom $p(x, y)$ has already been selected for a split in the path from the root to the child nodes, the constraint already exists in the body of the parent node. Therefore, instead of adding the constraint, we update the constraint on y (i.e., the range values of the atom $p(x, y)$).

Example 6. At node n_5, the rule r_{23} is expressed as $\text{worksIn}(x, y) \wedge \text{hasPopulation}(y, w) \wedge w \in [50K, 500K] \wedge \text{hasRevenue}(x, z)$. At node r_{22}, the rule is $\text{worksIn}(x, y) \wedge \text{hasPopulation}(y, w) \wedge w \notin (-\infty, 50K] \wedge \text{hasRevenue}(x, z)$. At node n_4, the rule r_{26} is expressed as $\text{worksIn}(x, y) \wedge \text{hasPopulation}(y, w) \wedge \text{hasRevenue}(x, z) \wedge (w \notin [500K, \infty) \vee z \notin [3k, \infty))$.

We use the stopping criteria based on $minHC$ and $minC$ of each node's inclusion and exclusion rules to decide when to stop splitting further. To do this, we calculate the size of class A and B using $< X, Y >$ as defined in the previous step. We stop if the number of different instantiations for the head that belong to class A is less than $minHC * headsize(r)$, and if the number for class B is less than $pca\_bodysize(r_s) - \frac{supp(r_s)}{minC}$. In other words, if neither of the rules can satisfy the expected $marginC$ and $marginHC$, we stop splitting.

Example 7. In Fig. 1, rule r_{24} will not appear in node n_5 because generalizing r_{12} with the constraint of the split would not exclude enough incorrect predictions to satisfy the $minC$.

We could select all nodes that meet the requirements of $minHC$ and $minC$. However, to limit the number of generated numerical rules and avoid redundant rules covering the same instances, we have implemented a strategy that selects the most general rules along each path from the root to the leaves. Specifically, we choose the rules with the highest PCA body size.

Example 8. In Fig. 1, rules corresponding to nodes n_1, n_3, and n_4 in Fig. 1, namely r_{11}, r_{21}, and r_{25} respectively, meet the requirements of $minHC$ and $minC$. We include r_{11} and r_{25} in $\mathcal{E}$ because nodes r_{11} and r_{21} are along the same path from the root, and r_{11} is the more general rule.

Rule Diversity[line 21 in Algorithm 1]. In order to maintain diversity, after each iteration, we remove any atoms that have led to a numerical rule with parent rule r that meets the conditions of $minHC$ and $minC$ from q_{atoms}, as explained in Sect. 4.1.

5 Experimental Evaluation

We have conducted two groups of experiments. First, we evaluate the quality of the set of enriched rules vs. their parent rules. The parent rules have been obtained by running rule mining techniques of AMIE [17] and AnyBURL [18]. Secondly, we have evaluated the performance of KG completion task using these enriched rule sets.

Datasets. We consider three different benchmark datasets that involve numerical values. *FB15K-237-num* and *DB15K-num* are variants of Freebase and DBPedia knowledge graphs involving numerical predicates and values proposed in [12]. *LitWD19K* is one of the three datasets proposed in LiterallyWikidata [13], which is a recent dataset gathered from Wikidata and Wikipedia with a special focus on literals. Table 1 shows the statistics for these datasets.

Table 1. Statistics of the benchmark datasets.$|\mathcal{G}_t|$ denotes the size of test set.

Dataset	$\|\mathcal{I}\|$	$\|\mathcal{P}\|$	$\|\mathcal{P}_{num\|}$	$\|\mathcal{G}\|$	$\|\mathcal{G}_t\|$
DB15K-num	12,867	278	251	79,345	9,789
FB15K-237-Num	14,541	237	116	272,115	1,215
LitWD19K	18,986	182	151	260,039	14,447

Experimental Setup. All experiments are run on a single machine with a processor 2.7 GHz, 8 cores, and 16GB of RAM that runs Mac OS X 10.13. REGNUM is written in Python and we have used Stardog[3] RDF data management system. The source code and the datasets used in our experiments are publicly available[4]. The time taken for rule generation ranges from 20 minutes to 15 hours, depending on the number of parent rules, their quality, and the KG.

5.1 Rules Quality Assessment

In this first set of experiments, we compare the quality of the parent rules that could be enriched with the set of enriched rules. Specifically, we compare the percentage of gain in terms of PCA confidence and head coverage. To measure the overall quality of the rules, we rely on $F_r = 2 * \frac{pca\_conf(r) * hc(r)}{pca\_conf(r) + hc(r)}$, which is a harmonic mean between the pca_conf and hc. This is because just a high pca_conf or a high hc is not a good indicator of the overall quality of a rule (i.e., the rule can be too specific or not yield good predictions).

Setup. We have run AMIE with default values $minHC = 0.01$ and $min\_pca\_conf = 0.1$, and maximum rule length of 3, on the *LitWD19K* and *DBPedia15K* datasets. To limit the number of parent rules, we used an increased $minHC = 0.1$ when running AMIE on *FB15K-237-num*. As AMIE mines only closed rules, no post-processing of the rules obtained was needed. We have run AnyBURL with default parameters except for the maximum rule length being set to 3, and the rules are learned for 100s with the $min\_conf = 0.03$. We performed post-processing on the obtained rules to retain only closed rules. REGNUM enriches the parent rules with $marginC = 20\%$, $marginHC = 10\%$.

Table 2. Statistic of rules mined by AMIE, compared to numerical rules in terms of the quality measure.

Dataset	$\|\mathcal{R}\|$	$\|\mathcal{R}_{enriched}\|$	$\|\mathcal{E}\|$	level 1	level 2	level 3	g_{conf}	g_{hc}	g_F
DB15K-num	4,163	402	2,783	2,747	36	0	+38.3%	−1.2%	+9.9%
FB15K-237-num	9,591	1,187	5,434	4,640	789	5	+28.6%	−4.2%	+9.8%
LitWD19K	2,481	859	9,068	7,764	1,272	12	+31.2%	−2.5%	+3.5%

[3] https://www.stardog.com/.
[4] https://github.com/armitakhn/REGNUM.

Table 2 and Table 3 detail the number of parent rules mined $\mathcal{R}$ by AMIE and AnyBURL, respectively. The number of parent rules that could be enriched with numerical predicates $\mathcal{R}_{enriched}$ and the number of numerical rules obtained by REGNUM $\mathcal{E}$ are also presented. On the three datasets, we compute the average of the rules' pca_conf, hc, and F measure of $\mathcal{R}_{enriched}$ and on the numerical rules $\mathcal{E}$. In the Tables 2 and 3, we provide the percentage of improvement of PCA confidence, head coverage, and F measure of $\mathcal{E}$ over parent rules $\mathcal{R}_{enriched}$, denoted by g_{conf}, g_{hc}, and g_F, respectively. The results indicate that the pca_conf of the numerical rules increased significantly across all benchmark datasets, irrespective of the rule mining technique used for obtaining parent rules. This improvement has been achieved without sacrificing much head coverage, and the overall quality of the rules (F measure) increased.

When we set a more relaxed value for $marginHC$, we noticed a decrease in the overall quality of rules. However, we were able to obtain more numerical rules. For instance, setting $marginHC$ to 20% on FB15K-237-num in Table 2 reduces the g_F from 9.8% to 6.22%, but the number of enriched rules increases to 10,141 with 1,744 parent rules that could be enriched.

Table 3. Statistic of rules mined by AnyBURL, compared to numerical rules regarding the quality measure.

Dataset	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{R}_{enriched}\rvert$	$\lvert\mathcal{E}\rvert$	level 1	level 2	level 3	g_{conf}	g_{hc}	g_F
DB15K-num	1,539	515	2,184	2,052	132	0	+30.4%	−3.5%	+3.3%
FB15K-237-num	7,959	1,688	7,252	6,597	654	1	+29.1%	−3.5%	+8.1%
LitWD19K	1,758	787	7,721	6,407	1,266	48	+29.0%	−3.8%	+4.5%

Approximately 25% of the rules on across all datasets incorporate membership constraints that include intervals. For example, the LitWD19K dataset mines 2,585 rules with membership constraints and 6,483 rules with non-membership constraints using parent rules from AMIE. Similarly, the AnyBURL dataset comprises 2,157 numerical rules with membership and 5,564 numerical rules with non-membership. As elaborated in Sect. 4.2, we expect that membership rules will generally have lower head coverage and higher PCA confidence compared to non-membership rules, which exclude incorrect predictions. This is demonstrated to be true when we limit the rules to only inclusion or only exclusion rules. For instance, for the LitWD19K dataset, the membership rules obtained from AMIE parent rules show g_{conf} of 36.8%, and g_{hc} of -3.0%, whereas for non-membership rules g_{conf} is 30.1%, and g_{hc} is -2.2%. We observe the same trend in all datasets.

We have also explored the use of the Minimum description length principle (MDLP) [8], and Optimal Binning [22] as supervised discretization techniques. However, these methods can only discretize a single numerical predicate at a time and cannot handle combinations of numerical predicates. To ensure a fair comparison, we have limited the rules of REGNUM to level 1. We have found that DT can enrich more parent rules by providing more relevant intervals. For example, on the FB15K-237-num dataset in Table 2, using MDLP results in the enrichment of 940 parent rules ($\lvert\mathcal{R}_{enriched}\rvert$), leading

to a total of 7,005 numerical rules ($|\mathcal{E}|$) with a g_F of 11.7%. On the same dataset, the results of Optimal Binning are $|\mathcal{R}_{enriched}|$ = 874, $|\mathcal{E}|$ = 3,832, and g_F = 12.0%. Finally, using only REGNUM rules with one numerical predicate (level 1) enriches 1,042 parent rules, resulting in 4640 numerical rules and a g_F of 10.13%, which is higher than the other two methods.

5.2 KG Completion

In this second set of experiments, we focus on the task of knowledge graph completion, where we aim to evaluate the efficacy of integrating the numerical rules obtained via REGNUM with the parent rules for knowledge graph completion. KG completion aims to predict a missing object o in a fact $(s, p, o) \notin \mathcal{G}$. While most current research on KG completion employs sub-symbolic approaches that involve embedding the graph into a low-dimensional vector space, rule-based methods offer the advantage of interpretability and explainability.

For each test data (s, p, o), we examine all the rules mined with predicate p in the head of the rule, i.e., $p(x, y)$. For each such rule, we execute a SPARQL query by substituting x with the subject of the test data s and obtain a set of predictions generated by the rule. Each candidate c can be given by a set of rules $C = \{R_1, ..., R_n\}$. We use four different aggregation strategies to assign a score to each candidate based on the rules that predicted them. The aggregation methods we considered are:

1. The democracy aggregation where the score depends on the number of rules that fired a candidate $\mathcal{S}_c = |C|$.
2. The max-aggregation $\mathcal{S}_c = max\{pca\_conf(R_1), ..., pca\_conf(R_n)\}$ where the rule with the highest PCA confidence defines the score.
3. The noisy-or aggregation $\mathcal{S}_c = 1 - \prod_{i=1}^{n}(1 - pca\_conf(R_i))$.
4. The weighted-F aggregation $\mathcal{S}_c = \Sigma_{i=1}^{n} \frac{1}{\#Prediction(R_i)} * f(R_i)$ which penalizes the rules that result in many predictions (candidates).

The rules with statistics reported in Table 2 are used to find the candidates. To assess the performance of the rules, we report the Hits@10 result, which is the number of correct head terms predicted out of the top 10 predictions. Table 4 shows the results of KG completion on three datasets using the rules mined by AMIE vs. the numerical rules of REGNUM added to the set of rules of AMIE. The four different aggregations are used to score the candidates and report the hits@10 results in the filtered setting (i.e., a prediction that already exists in $\mathcal{G}$ or $\mathcal{G}_t$ will not be ranked).

On all three datasets, we found that adding the rules of REGNUM to the set of rules from AMIE improved the performance of knowledge graph completion when using the Max aggregation method. This suggests that numerical rules can improve predictions. With the Max aggregation method, we know that whenever a candidate is selected, it is because a numerical rule of REGNUM with higher confidence than its parent rule has been chosen. If no numerical rule exists, the parent rule will be chosen.

The marginal benefit of numerical rules on these benchmark datasets can be attributed to the small number of rules that could be enriched, as well as the generic nature of the datasets that do not heavily rely on numerical predicates for accurate predictions. Hence, to better understand the impact of the enriched rules, we focus only

Table 4. Hits@10 results of KG completion with rules of AMIE ($\mathcal{R}$) and numerical rules of REGNUM with the rules of AMIE ($\mathcal{R} \cup \mathcal{E}$)

	FB15K-237-num		DBPedia15K		LitWD19K	
	AMIE	AMIE+REGNUM	AMIE	AMIE+REGNUM	AMIE	AMIE+REGNUM
Democ	61.6	61.0	33.8	35.8	31.9	31.6
Max	70.5	71.7	34.5	36.9	32.4	32.6
Noisy-or	68.1	66.9	34.7	37.0	32.5	32.4
Weighted-F	69.1	68.3	34.7	37.0	32.9	32.8

on the rules that could be enriched, $R_{enriched}$, and use them for knowledge graph completion. Table 5 shows the results using the Max aggregation method, indicating improvements in the accuracy of knowledge graph completion when enriched rules are combined with their respective parent rules.

Table 5. Hits@1 and Hits@10 results of KG completion with $\mathcal{R}_{enriched}$ and $\mathcal{R}_{enriched} \cup \mathcal{E}$

	AMIE		AMIE+REGNUM	
	Hits@1	Hits@10	Hits@1	Hits@10
DBPedia15K	4.6	7.7	6.4	10.2
FB15K-237-num	5.5	14.7	6.3	15.3
LitWD19K	12.6	22.5	13.9	23.6

6 Conclusion and Future Work

In this paper, we introduced REGNUM, a novel approach that builds numerical rules on the shoulders of the rules mined by a rule-mining system. The parent rules are enriched with numerical predicates, with their values being constraints to membership or non-membership to intervals obtained through supervised discretization. We showed that the enriched rules have a higher average quality and can assist in improving the accuracy of rule mining systems on the knowledge graph completion task.

Future work will explore alternative methods of obtaining constraints, such as using sequential covering approaches and applying numerical rules to other domains where numerical values are crucial for predictions. We also plan to investigate more complex aggregation techniques, such as latent-based Aggregation [3], and consider using an in-memory database to improve query run-time, as proposed in AMIE3 [17]. Ultimately, we intend to compare our results regarding both run-time and optimality with an approach that finds optimal intervals while mining the numerical rule. We expect our approach to be faster but less accurate.

Acknowledgements. This work has been supported by the project PSPC AIDA: 2019-PSPC-09 funded by BPI-France.

References

1. Ahmadi, N., Lee, J., Papotti, P., Saeed, M.: Explainable fact checking with probabilistic answer set programming. CoRR abs/1906.09198 (2019)
2. Aumann, Y., Lindell, Y.: A statistical theory for quantitative association rules. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 1999, pp. 261–270. Association for Computing Machinery, New York (1999). https://doi.org/10.1145/312129.312243
3. Betz, P., Meilicke, C., Stuckenschmidt, H.: Supervised knowledge aggregation for knowledge graph completion. In: Groth, P., et al. (eds.) ESWC 2022. LNCS, vol. 13261, pp. 74–92. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06981-9_5
4. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. Taylor & Francis, Milton Park (1984)
5. Bühmann, L., Lehmann, J., Westphal, P.: Dl-learner-a framework for inductive learning on the semantic web. J. Web Semant. **39**, 15–24 (2016)
6. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann (1995)
7. Dehaspe, L., Toironen, H.: Discovery of Relational Association Rules. In: Džeroski, S., Lavrač, N. (eds.) Relational Data Mining, pp. 189–208. Springer, Heidelberg (2001). https://doi.org/10.1007/978-3-662-04599-2_8
8. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI (1993)
9. Galárraga, L., Razniewski, S., Amarilli, A., Suchanek, F.M.: Predicting completeness in knowledge bases. In: de Rijke, M., Shokouhi, M., Tomkins, A., Zhang, M. (eds.) Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, 6–10 February 2017, pp. 375–383. ACM (2017)
10. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB J. **24**(6), 707–730 (2015)
11. Galárraga, L.A., Preda, N., Suchanek, F.M.: Mining rules to align knowledge bases. In: Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, AKBC@CIKM 2013, San Francisco, California, USA, 27–28 October 2013, pp. 43–48. ACM (2013)
12. García-Durán, A., Niepert, M.: KBLRN: end-to-end learning of knowledge base representations with latent, relational, and numerical features. In: Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI) (2018)
13. Gesese, G.A., Alam, M., Sack, H.: LiterallyWikidata - a benchmark for knowledge graph completion using literals. In: Hotho, A., et al. (eds.) ISWC 2021. LNCS, vol. 12922, pp. 511–527. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88361-4_30
14. Hühn, J., Hüllermeier, E.: FURIA: an algorithm for unordered fuzzy rule induction. Data Mining Knowl. Discov. **19**(3), 293–319 (2009). https://doi.org/10.1007/s10618-009-0131-8
15. Jaramillo, I.F., Garzás, J., Redchuk, A.: Numerical association rule mining from a defined schema using the VMO algorithm. Appl. Sci. **11**(13), 6154 (2021). https://doi.org/10.3390/app11136154
16. Khajeh Nassiri, A., Pernelle, N., Saïs, F., Quercini, G.: Generating referring expressions from RDF knowledge graphs for data linking. In: The Semantic Web – ISWC 2020 (2020)
17. Lajus, J., Galárraga, L., Suchanek, F.: Fast and exact rule mining with AMIE 3. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 36–52. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_3
18. Meilicke, C., Chekol, M.W., Fink, M., Stuckenschmidt, H.: Reinforced anytime bottom up rule learning for knowledge graph completion. arXiv preprint arXiv:2004.04412 (2020)

19. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 3137–3143 (7 2019)
20. Minaei-Bidgoli, B., Barmaki, R., Nasiri, M.: Mining numerical association rules via multi-objective genetic algorithms. Inf. Sci. **233**, 15–24 (2013). https://doi.org/10.1016/j.ins.2013.01.028. https://www.sciencedirect.com/science/article/pii/S0020025513001072
21. Muggleton, S.: Learning from positive data. In: Muggleton, S. (ed.) ILP 1996. LNCS, vol. 1314, pp. 358–376. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63494-0_65
22. Navas-Palencia, G.: Optimal binning: mathematical programming formulation abs/2001.08025 (2020). http://arxiv.org/abs/2001.08025
23. Ortona, S., Meduri, V.V., Papotti, P.: Robust discovery of positive and negative rules in knowledge bases. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 1168–1179 (2018)
24. Ortona, S., Meduri, V.V., Papotti, P.: Rudik: rule discovery in knowledge bases. Proc. VLDB Endow. **11**(12), 1946–1949 (2018)
25. Sadeghian, A., Armandpour, M., Ding, P., Wang, D.Z.: DRUM: End-to-End Differentiable Rule Mining on Knowledge Graphs. Curran Associates Inc., Red Hook (2019)
26. Salleb-Aouissi, A., Vrain, C., Nortet, C.: Quantminer: a genetic algorithm for mining quantitative association rules. In: IJCAI, pp. 1035–1040 (2007)
27. Srikant, R., Agrawal, R.: Mining quantitative association rules in large relational tables. In: ACM SIGMOD Conference (1996)
28. Wang, P.W., Stepanova, D., Domokos, C., Kolter, J.Z.: Differentiable learning of numerical rules in knowledge graphs. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=rJleKgrKwS
29. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
30. Zeng, Q., Patel, J.M., Page, D.: Quickfoil: scalable inductive logic programming. Proc. VLDB Endow. **8**(3), 197–208 (2014)