SPARQL-based relaxed rules for learning over knowledge graphs

Anne Mindika Premachandra[1,*], Kerry Taylor[1] and Sergio Rodríguez-Méndez[1]

[1]School of Computing, The Australian National University, Australia

Abstract

In today's world where explainable AI is gaining importance, rule learning is a classic but favourable machine learning approach that can be enhanced using advances in semantic web technologies. The expressiveness of rules learnt over Knowledge Graphs (KGs) is largely dependent on the language bias of a rule learner which in turn determines the complexity of the search space for models. In this paper, we propose a relaxed rule specification approach that allows rules to contain branches and tree shapes, negated graph patterns, and SPARQL style filtering which adds numerical and temporal comparisons into the rule's vocabulary. By targeting compatibility with SPARQL, we can tap into the advanced query optimisation features of triple stores which implement SPARQL to evaluate and apply weakly-constrained rules as SPARQL queries. We use an expert-guided rule-template based approach to generate candidate rules and we extend standard rule quality measures for such relaxed rules. We introduce a further extension that can be used to extract numeric attribute values in order to include numeric attributes within our symbolic rule learning framework.

Keywords

Rule learning, Rule mining, Knowledge graphs, Inductive logic programming, SPARQL

1. Introduction

Knowledge Graphs (KGs) are a modern database-meets-knowledge-representation encoding of semi-structured data that enables efficient querying and logical deductive reasoning. First-order-logic rules expressed over KGs can offer explainable patterns in the KG and can predict missing facts. Hence such rules are a useful option for representation of predictive models built from KGs as training data.

Typically, when learning logic rules over KGs, there is a difficult tradeoff in selecting a rule language that is adequately expressive for accuracy, predictive power, and compact explainability, but not so expressive that the search space for good rules exceeds computational feasibility. The selection of a rule language imposes a bias on the predictive models that can be learnt.

Many KGs contain schema information, usually in a form of an *OWL*[1] *ontology*, describing entity types, class relationships, relation domain and range, cardinality constraints, and so forth. Some rule learners also make use of this information when learning rules [1, 2].

We wish to address the problem of learning meaningful, accurate, and expressive rules for open-ended problems. According to Michalski and Chilausky [3], *"open-endedness implies that when we make inductive assertions about some piece of reality, there is no natural limit to the level of detail of descriptions of this reality, to the scope of concepts and operators used in the expressions of these assertions, or to the richness of their forms."* We wish to achieve richness of learnt rules by relaxing the language bias of the rule specification, which allows us to express more complex rules than is possible by the restrictive language biases adopted in related work.

SemIIM'24: Third International Workshop on Semantic Industrial Information Modelling, co-located with ISWC'24, 12th November 2024, Baltimore, USA

*Corresponding author.

Mindika.Premachandra@anu.edu.au (A. M. Premachandra); Kerry.Taylor@anu.edu.au (K. Taylor); Sergio.RodriguezMendez@anu.edu.au (S. Rodríguez-Méndez)

https://researchportalplus.anu.edu.au/en/persons/franciscu-premachandra (A. M. Premachandra); https://researchportalplus.anu.edu.au/en/persons/kerry-taylor (K. Taylor); https://researchportalplus.anu.edu.au/en/persons/sergio-rodriguez-mendez (S. Rodríguez-Méndez)

0000-0002-0877-7063 (A. M. Premachandra); 0000-0003-2447-1088 (K. Taylor); 0000-0001-7203-8399 (S. Rodríguez-Méndez)

[1]https://www.w3.org/TR/owl-overview/

This relaxation greatly increases the expressiveness of the rules and makes rule learning challenging given the increasing search space and complexity required to learn rules of such forms. To achieve this, we begin with manually crafted rule templates, that are constructed using expert domain knowledge and explicit ontology information. The rule templates provide human-in-the-loop hints to the automated process of model-building. Templates enable the simultaneous specification of both rule structural forms and rule vocabulary, that is, both the shape of rules and the KG terms to be used. Thus our approach is an expert guided technique, trading-off learning automation vs. rule expressiveness.

The main contributions of our work are the following:

- We propose a relaxed rule specification that allows rules to contain: branches and tree shapes, negated graph patterns (in contrast to negated atoms) and SPARQL[2] style filtering (including numerical comparisons, and variable inequality constraints). (Section 3.2)
- We define the standard rule quality measures of Rule Support, Standard Confidence and Head Coverage for the proposed relaxed rules. (Section 3.4)
- We use an expert-guided template-based approach to generate multiple rules of similar format, thus minimising the effort needed to separately specify rules that share the same structure. (Sections 3.3 and 3.5)

A benefit of our SPARQL compatibility is that the rule specification can be easily converted to a SPARQL query to directly work with RDF[3] KGs stored in RDF Triple Stores such as Virtuoso[4] and GraphDB[5], thus taking advantage of their inherent query optimisation techniques [4] when evaluating complex rules. We present examples of our relaxed rules defined over the *FutureSOILS* KG, recently built for the project *FutureSOILS: Future Proofing the Soils of Southern and Central NSW from Acidification and Soil Organic Carbon Decline*[6] to predict the pH response of the soil to various liming techniques adopted by farmers. Rule quality measures (such as Standard Confidence, Head Coverage, Rule Support, Rule Body Support, and Head Support) have been evaluated by issuing SPARQL queries generated from the rule to a Virtuoso triple store. Similarly, when making predictions from the rules, instance tables matching rule head and body patterns are retrieved from the KG by issuing SPARQL SELECT queries which can be generated from the rule specification.

In a separate additional use of this relaxed specification, we have allowed SPARQL style variable binding and projection as well as optional graph patterns in the rule specification (in Section 4), so that it can be used to retrieve tabular data from the KG which can be fed into other machine learning models which require typical tabular data as input. In the *FutureSOILS* work, such attribute extraction rules have been used to extract tabular data to train numeric prediction models such as *Random Forest Regressor* to predict soil acidity represented as pH values.

2. Background and related work

A KG contains a set of RDF triples of the form (s, p, o) where s is called the *subject*, o the *object* and p the *relation* (also called *predicate* in this paper), which we write as $p(s, o)$ following the convention of *first-order* logic. We consider KGs with *binary* relations of the above form, and *unary* relations conveniently written as $p(s, s)$.

A *first-order Horn rule* r is of the form

$$h \impliedby b_1 \wedge b_2 \wedge \ldots \wedge b_n \tag{1}$$

where h and b's are atoms of the form $p(t_1, t_2)$ where p is a predicate and t_1, t_2 are terms [5]. A term is either a variable, entity or literal value (such as a number, text value, date, etc.). Without loss of

[2] https://www.w3.org/TR/sparql11-overview/
[3] https://www.w3.org/TR/rdf11-primer/
[4] https://virtuoso.openlinksw.com/
[5] https://www.ontotext.com/products/graphdb/
[6] https://farmlink.com.au/futuresoils

generality, we use the term *constant* to refer to both entities and literal values. The atom h is the *head* of r and the conjunction of atoms $b_1, \ldots, b_n$ is the *body* of r.

For rule learning in KGs, a very restricted class of *first-order Horn rules* is commonly employed, *closed-path* (CP) rules, where the body atoms form a continuous path from the subject to the object of the head atom. A CP rule is of the following form

$$h(x_o, x_n) \impliedby b_1(x_o, x_1) \wedge b_2(x_1, x_2) \wedge \ldots \wedge b_n(x_{n-1}, x_n) \tag{2}$$

where a predicate (of the head and body atoms) may be interpreted as its *inverse* with the subject and object bindings swapped and x_j's ($0 \leq j \leq n$) can only be variables. The head predicate h may occur in the body.

We now recall the *closed* and *connected* properties of a rule. A variable is closed if it appears at least twice in the rule. A rule is closed if all the variables in the rule are closed. Two atoms in a rule are connected if they share a variable or constant. A rule is connected if every atom in the rule is connected to another atom. A CP rule is both closed and connected, but not vice versa [5].

Various extensions of CP rules have been proposed and learnt by other rule learners by relaxing the language bias and allowing unary predicates, thereby increasing the expressiveness of the learnt rules. For example, AMIE [6, 7, 8] can only learn *connected* and *closed* rules (which is a relaxation of CP rules) over binary predicates. AnyBURL [9, 10, 11] limits the occurrence of the variables in the learnt rule to occur at most twice and requires strict variable inequality while allowing entities in limited places. Omran et al. [12] define more general *open-path rules* by allowing an open variable in the head and the last body atom, but still requiring body atoms to form a continuous path linking each atom to its successive atom. They also define *inverse open-path rules* to represent branching out patterns (SHACL[7] shapes) which are systematically aggregated to generate tree-shaped rules.

Some rule learners have achieved more expressiveness by going beyond first-order Horn rules. For example, allowing negated atoms in the rule body [13], simple comparisons among numeric values [14, 15] and dealing with temporal values [16, 17] over a temporal KG where every atom has a timestamp [5]. Our proposed rule specification follows the structure of Horn rules in that it doesn't require a continuous path being formed by the body patterns thereby allowing branching and tree shapes, but goes beyond Horn rules by allowing multiple negated graph patterns in the body (in contrast with negated single atoms). It also allows complex numeric filtering and time comparisons while still working on a KG with binary predicates.

3. Relaxed rule specification

3.1. Context of the prediction

We would first like to introduce the context of the *FutureSOILS* [18] KG used in this study in order to motivate the need for the proposed relaxations. The KG contains information about several farm management units and controlled replicated trials of various management strategies used over several years, as well as measurements of soil characteristics and information about crop rotations and harvest yield. The data was required to conform to pre-determined data guidelines and the KG was modelled according to an ontology designed to match the scope of data available. Due to the abundance of N-ary relation patterns in the input data, the KG adopts the standard design for modelling such *N-ary* relations in RDF [19]. The complete concept map of the ontology and resources related to this paper is accessible in https://w3id.org/kgcp/SRRS.

Numerical values representing soil and crop data and management techniques were transformed to entity instances with expert guidance, or a frequency based approach when industry-adopted bin intervals were unavailable. For example, soil pH[8] values were binned into four intervals as follows [*, 4.5), [4.5, 4.8), [4.8, 5.5), [5.5, *].

[7] https://www.w3.org/TR/shacl/

[8] pH values measured using the 1:5 CaCl2 method.

In the project's scope, the experts were interested in asking several predictive scenarios and gained insights from the modelled data, such as the following, *given a particular method of liming (e.g. application of 3 t/ha of lime to the soil followed by a particular cultivation treatment) and given the starting pH value of the soil, what would be the pH (bin) value of the soil one year later? And also two/three years later?*. Additional information may be given in each scenario question, such as: Soil Aluminium %, Carbon %, crops planted, weather data, etc.

The structure of the KG that links to multiple N-ary relations[9] requires rules that can represent branching patterns and can compare numerical and time values. Consider the following example rule presented in natural language: *If the starting pH value is between 4.8-5.5 at soil depth 7.5-10.00 cm and the unit was limed at a rate 2.0-3.0 t/ha and cultivated with Low Soil Disturbance, then the pH value 1 year later at the same depth would be 4.5-4.8.*

Note that this rule would not have been possible in the other rule learners we have discussed, for e.g. AMIE or AnyBURL does not allow rules with branch and tree shaped structures as we have regarding the farm management unit in this rule (see Table 1 for a comparison of expressiveness capabilities among different learners).

In favour of increased rule coverage, we relax some of the temporal definitions like 1 year to include ±2 months (i.e. 10-14 months later) and consider pH measurements taken within 2 months of liming. Some examples of variations of the same rule are given below:

1. Generalisation by shortening the rule (e.g. disregarding cultivation treatment)
2. Generalisation by replacing an entity with a variable (e.g. suffice to have that any cultivation treatment was given, or that liming occurred irrespective of depth and rate)
3. Variation of the rule by adding a negated sub-string (e.g. requiring that cultivation treatment was not done)
4. Refinement of of the rule by adding a further cultivation related information (e.g. also consider depth of cultivation)
5. Refinement of the rule by adding adding more pre-lime observations (e.g. soil Aluminium percentage, Total Carbon percentage)

See Michalski and Chilausky [3] for a comprehensive list of rule generalisation techniques like this for inductive learning.

3.2. Relaxed rules for inductive reasoning

We now formally specify our relaxed rule language that can support the increased expressiveness required to model complex rules as described above.

$$h \impliedby b_1 \wedge b_2 \wedge \ldots \wedge b_n \wedge \sim \{b_{n+1} \wedge b_{n+2} \wedge \ldots \wedge b_m\}^* \tag{3}$$

Atoms enclosed by $\sim \{\ldots\}$ have the meaning of the SPARQL FILTER NOT EXISTS operator[10] over graph patterns (i.e. a negated graph pattern which is not known to exist in the KG) and a rule may contain zero or more negated graph patterns (multiplicity denoted by the Kleene star *).

The rules also need to be *connected*, that is, every atom in the rule is connected to another atom by sharing a variable or constant with another atom. Additionally, all atoms within a negated graph pattern need to be connected with another atom in the pattern and at least one atom of the pattern should be connected to an outside atom(s) which is not part of a negated graph pattern. The rule does not necessarily need to be *closed*, but we require that at least one head variable occurs in a non negated atom(s) of the body.

Rules can optionally contain the additional constructs of BIND[11] and FILTER[12] clauses that need to be SPARQL compatible. FILTER clauses allow us to specify multiple additional constraints not usually

[9]Most particularly the branching structure around the farm management unit called an *ECCMO* in the ontology.
[10]https://www.w3.org/TR/sparql11-query/#func-filter-exists
[11]https://www.w3.org/TR/sparql11-query/#bind
[12]https://www.w3.org/TR/sparql11-query/#expressions

supported by traditional rule specifications (such as numerical comparisons, variable inequality constraints, type checking, etc.), allowing greater flexibility to work with existing KGs without introducing additional entity mappings into the KG. BIND clauses can be used for specifying new variable bindings which can be used in FILTER clauses.

These rules are closely related to Tree-shaped rules in Omran et al. [12] and rules with negated atoms in Ho et al. [13]. We have extended the concept of negated atoms to negated graph patterns and we have performed numerical and temporal comparisons using a KG with binary predicates as illustrated in the examples further below.

3.3. Template based rule generation

In bottom-up rule learners such as AnyBurl [11], one way of generalising a rule is by replacing entities with variables [3]. In our work, we take a top-down approach, where we work with a set of template rules and then generate variants of the template by replacing selected variables with all the possible combinations of entities that can appear in the respective rule atoms.

For example, consider the following rule template from *FutureSOILS* that attempts to predict the soil pH roughly one year following liming. The variables ?*m_X*, ?*m_pH*, ?*m_lime* represent literal values of type *xsd:positiveInteger* and a relaxed definition of time has been implemented by including numerical comparisons in the FILTER clauses where multiple filtering constraints can be specified. Note the negated graph pattern at the end of the rule which specifies that the unit was limed but not cultivated in this example. To differentiate between variables and constants, the variables in the rule are prefixed with **"?"**. Predicates and entities are in RDF prefixed format. Although the BIND clause is not strictly essential in this example, we show it here to demonstrate its usage.

```
BIND[(?m_X - ?m_lime AS ?gap)]
FILTER[(?gap >= 10 && ?gap <= 14), (?m_pH - ?m_lime <= 2 && ?m_lime - ?m_pH <= 2)]
fs-data:pH__(1-5-CaCl2)__PSC(?X,?pH_bin_X) <=
    fs-onto:hasObservation(?eccmo_X,?X),
        sosa:observedProperty(?X,fs-data:ObservedProperty-pH),
        sosa:usedProcedure(?X,fs-data:MoA-1-5-CaCl2),
        fs-onto:hasDepth(?X, ?d_pH),
        sosa:phenomenonTime(?X, ?t_X),
        fs-onto:hasMonthCount(?t_X, ?m_X),
    fs-onto:hasECCMO(?eccmoByRep_X, ?eccmo_X),
        rdf:type(?eccmoByRep_X, fs-onto:ECCMObyRep),
        fs-onto:hasControl(?eccmoByRep_X, ?eccmo_C),
    fs-onto:hasObservation(?eccmo_C, ?obs_pH),
        fs-data:pH__(1-5-CaCl2)__PSC(?obs_pH,?pH_bin),
        fs-onto:hasDepth(?obs_pH, ?d_pH),
        sosa:phenomenonTime(?obs_pH, ?t_pH),
        fs-onto:hasMonthCount(?t_pH, ?m_pH),
    fs-onto:hasObservation(?eccmo_C, ?obs_Al),
        sosa:phenomenonTime(?obs_Al, ?t_pH),
        fs-onto:hasDepth(?obs_Al, ?d_pH),
        fs-data:Aluminium-%-of-Cations__(Calculated)__%__PSC(?obs_Al, ?Al_bin),
    fs-onto:hasECCMO(?mu_lime, ?eccmo_X),
        fs-onto:hasTime(?mu_lime, ?t_lime),
        fs-onto:hasMonthCount(?t_lime, ?m_lime),
        fs-onto:hasManagement(?mu_lime,?mng_lime),
        rdf:type(?mng_lime, fs-onto:Management-Amelioration-Lime),
        fs-data:Lime-Application-Rate__PSC(?mng_lime, ?lr_bin),
    ~{fs-onto:hasECCMO(?mu_culti, ?eccmo_X),
        fs-onto:hasTime(?mu_culti, ?t_culti),
        fs-onto:hasMonthCount(?t_culti, ?m_lime),
        fs-onto:hasManagement(?mu_culti,?mng_culti),
        rdf:type(?mng_culti, fs-onto:Management-Amelioration)}
```

Now, if we replace some of the variables of the above template with the possible combinations of candidate entity values for the variables, we can generate multiple rule variants from the above template. In the *FutureSOILS* domain, replacing ?*pH_bin_X*, ?*d_pH*, ?*pH_bin*, ?*Al_bin*, ?*lr_bin* in the above template yielded 6144 rule variants with the head atom containing a variable and a constant. This was done by declaring (predicate, variable name) combinations to be replaced in a JSON file (see Figure 1). The candidate entity values for the variables can be retrieved from the KG itself at the time of generating the rule variants (refer Algorithms 1 and 2).

```
{
"rules":[],
"combinations": {
  "fs-data:pH__(1-5-CaCl2)__PSC" : ["?pH_bin_X", "?pH_bin"],
  "fs-onto:hasDepth" : ["?d_X", "?d_pH"],
  "fs-data:Lime-Application-Rate__PSC" : ["?lr_bin"],
  "fs-data:Total-Carbon-(Any-MoA)-%__PSC": ["?C_bin"],
  "fs-data:Aluminium-_PC_-of-Cations__(Calculated)__%__PSC" : ["?Al_bin"],
  "fs-data:Effective-Cation-Exchange-Capacity__(Calculated)__cmol(+)/kg__PSC" : ["?ECEC_bin"],
  "skos:broader" : ["?culti_lvl"],
  "fs-data:Depth-Of-Cultivation__PSC" : ["?d_culti_bin"]
  }
}
```

Figure 1: A sample JSON file specifying variable names to be replaced

Algorithm 1 Retrieving entities for replacing variables

Require: The KG and the set of predicate, variable names $\{(p, v)\} = \{(p_1, v_1), (p_2, v_2), \ldots (p_n, v_n)\}$ to be replaced (assuming unique v_n values)

$c \leftarrow dictionary()$ ▷ Declare an empty dictionary
for each p **do**
    $c[p] \leftarrow \{object | (subject, predicate, object) \in \text{KG} \ \& \ predicate == p\}$ ▷ assuming that v will be referred in the the rule atom as $q(t_x, v)$
end for
return c

Algorithm 2 Rule variant generation from template

Require: A template rule r, the set of predicate, variable names $\{(p, v)\}$ to be replaced and combination values c for each predicate from Algorithm 1

$d \leftarrow dictionary()$ ▷ Declare an empty dictionary
for each atom $q(t1, t2) \in r$ **do**
    if $(q, t_2) \in \{(p, v)\}$ **then**
        $d[v] \leftarrow c[p]$ ▷ Assume v is unique
    end if
end for
$rule\_variants \leftarrow list()$ ▷ Declare an empty list
for each $(c_1, c_2, \ldots, c_n) \in CartesianProduct(d[v\_1], d[v\_2], \ldots, d[v\_n])$ **do**
    $r' \leftarrow$ Replace $(v_1, v_2, \ldots, v_n)$ variable names in rule r with $(c_1, c_2, \ldots, c_n)$
    Append r' to $rule\_variants$
end for
return $rule\_variants$

Each rule variant can then be assessed by evaluating rule confidence measures such as Standard Confidence, Head Coverage and Rule Support by issuing SPARQL queries generated from the rule to

the KG triple store. This process is parallelised to save time.

3.4. Relaxed rule quality measures

We now define standard rule quality measures for our relaxed rules: Rule Support (RS), Standard Confidence (SC) and Head Coverage (HC). All of these measures are calculated over the known KG data as it binds to the body and head of the rule, so that a closed-world assumption is made for this evaluation. RS is a count of the number of distinct facts that the rule predicts correctly and is the numerator when calculating SC and HC. SC measures the proportion of correct predictions wrt all distinct predictions that the rule can make given the body bindings in the KG. Note here that missing predictions and known-false predictions are treated equivalently here. HC measures the proportion of the known facts satisfying the rule head that are correctly predicted by the rule, and gives an indication of the generalisation power of the rule.

Since the existing rule quality measures in the literature are based on more restrictive language biases than ours, we define rule quality measures for our relaxed rules which uses a relaxed language bias. These definitions encompass the traditional rule quality measures [12, 8, 13] and falls back to the traditional definitions when the same language bias referred in the traditional measures are being used.

Consider the following rule of form (3), where we have expanded the atoms to depict the variables and constants subject to the constraints specified above. A predicate is interpreted also as its inverse with the subject and object bindings swapped around.

$$\begin{aligned} h(t_0, t') \impliedby & \wedge_{i=1}^{n} b_i(t_{(2i-2)}, t_{(2i-1)}) \\ & \wedge \sim \{\wedge_{j_1=(n+1)}^{n_1} b_{j_1}(t_{(2j_1-2)}, t_{(2j_1-1)})\} \\ & \wedge \sim \{\wedge_{j_2=(n_1+1)}^{n_2} b_{j_2}(t_{(2j_2-2)}, t_{(2j_2-1)})\} \\ & \wedge \ldots \wedge \sim \{\wedge_{j_m=(n_{(m-1)}+1)}^{n_m} b_{j_m}(t_{(2j_m-2)}, t_{(2j_m-1)})\} \end{aligned} \tag{4}$$

where h and each b is a predicate in the KG. To satisfy the *connectedness* constraints of the relaxed rule, some terms (t's) will be equal and t_0 is a closed variable. We denote the set of term equalities as $E = \{(t_x, t_y) | t_x = t_y, t_x \in T, t_y \in T\}$, where $T \in \{t', t_0, \ldots, t_{(2j_m-1)}\}$ (for example, when $t_1 = t_2$). We denote the set of constraints defined on T specified by the BIND and FILTER statements as F. Recall that some terms in T may be constants.

First, let us consider rules with two closed variables in the rule head, so we have $t' = t_{2n-1}$. Since the order of the atoms is irrelevant, without loss of generality, we require the variables in the head to be connected to the first and last non-negated atoms.

Definition 3.1 (Quality measures for relaxed rules with two closed variables in the head)**.** Let r be a relaxed rule of the form (4) and $t' = t_{2n-1}$. Then a pair of constants (e_0, e') satisfies the body of r denoted $body_r(e_0, e')$, if there exists constants $e_0, \ldots, e_{2n-1}$ in the KG (with $e' = e_{2n-1}$) such that $b_1(e_0, e_1), b_2(e_2, e_3), \ldots, b_n(e_{2n-2}, e_{2n-1})$ are facts in the KG and for each negated graph pattern $\{\wedge_{j_k=(n_{(k-1)}+1)}^{n_k} b_{j_k}(e_{(2j_k-2)}, e_{(2j_k-1)})\}$ (in $1 \leq k \leq m$ where $n_0 = n$), the atoms (predicates and the variable bindings) specified in the entire negated graph pattern does not exist in the KG, subject to the term equalities E and constraints F. Also (e_0, e') satisfies the head of r denoted $head_r(e_0, e')$, if $h(e_0, e')$ is a fact in the KG. The rule support (RS), standard confidence (SC) and head coverage (HC) of r are given respectively by

$$RS(r) = |\{(e_0, e') \,:\, body_r(e_0, e') \; and \; head_r(e_0, e')\}|$$

$$SC(r) = \frac{RS(r)}{|\{(e_0, e') \,:\, body_r(e_0, e')\}|}$$

$$HC(r) = \frac{RS(r)}{|\{(e_0, e') \,:\, head_r(e_0, e')\}|}$$

Now, let us consider rules with a constant in the rule head.

Definition 3.2 (Quality measures for relaxed rules with a constant in the head)**.** Let r be a relaxed rule of the form (4) and $t' = c$, where c is a constant (either an entity or literal value). Then the constant e_0 satisfies the body of r denoted $body_r(e_0)$, if there exists constants $e_0, \ldots, e_{2n-1}$ in the KG such that $b_1(e_0, e_1), b_2(e_2, e_3), \ldots, b_n(e_{2n-2}, e_{2n-1})$ are facts in the KG and for each negated graph pattern $\{\wedge_{j_k=(n_{(k-1)}+1)}^{n_k} b_{j_k}(e_{(2j_k-2)}, e_{(2j_k-1)})\}$ (in $1 \leq k \leq m$ where $n_0 = n$), the atoms (predicates and the variable bindings) specified in the entire negated graph pattern does not exist in the KG, subject to the term equalities E and constraints F. Also e_0 satisfies the head of r denoted $head_r(e_0, c)$, if $h(e_0, c)$ is a fact in the KG. The rule support (RS), standard confidence (SC) and head coverage (HC) of r are given respectively by

$$RS(r) = |\{e_0 \ : \ body_r(e_0) \ and \ head_r(e_0, c)\}|$$

$$SC(r) = \frac{RS(r)}{|\{e_0 \ : \ body_r(e_0)\}|}$$

$$HC(r) = \frac{RS(r)}{|\{e_0 \ : \ head_r(e_0, c)\}|}$$

Finally, let us consider rules with an open variable in the rule head which implies t' does not occur in the rule body.

Definition 3.3 (Quality measures for relaxed rules with an open variable in the head)**.** Let r be a relaxed rule of the form (4) and the variable $t' \notin \{t_0, \ldots, t_{(2j_m-1)}\}$. Then the constant e_0 satisfies the body of r denoted $body_r(e_0)$, if there exist constants $e_0, \ldots, e_{2n-1}$ in the KG such that $b_1(e_0, e_1), b_2(e_2, e_3), \ldots, b_n(e_{2n-2}, e_{2n-1})$ are facts in the KG and for each negated graph pattern $\{\wedge_{j_k=(n_{(k-1)}+1)}^{n_k} b_{j_k}(e_{(2j_k-2)}, e_{(2j_k-1)})\}$ (in $1 \leq k \leq m$ where $n_0 = n$), the atoms (predicates and the variable bindings) specified in the entire negated graph pattern does not exist in the KG, subject to the term equalities E and constraints F. Also e_0, satisfies the head of r denoted $head_r(e_0, e')$, if $h(e_0, e')$ is a fact in the KG for any constant e'. The rule support (RS), standard confidence (SC) and head coverage (HC) of r are given respectively by

$$RS(r) = |\{e_0 \ : \ \exists e', body_r(e_0) \ and \ head_r(e_0, e')\}|$$

$$SC(r) = \frac{RS(r)}{|\{e_0 \ : \ body_r(e_0)\}|}$$

$$HC(r) = \frac{RS(r)}{|\{e_0 \ : \ \exists e', head_r(e_0, e')\}|}$$

In our experience, we found that the SC and HC which are the most widely adopted rule measures were not sufficient to measure the quality of a rule. For example, a rule that correctly predicts 4 out of 5 times and another rule that correctly predicts 80 out of 100 times would get the same SC value of 0.8 and would not convey the number of instances covered by the body of the rule. Those instances provide the evidence for the confidence, while the number of instances covered by the head indicate the generalisation power of the rule. Hence, we also retain the denominator of SC as Rule Body Support (BS) and the denominator of HC as Head Support (HS), which are related to the sample size of SC and HC respectively.

3.5. Template generation

In Section 3.1, we discussed more examples of rule generalisation and refinement involving rule shortening, extending and considering negated graph patterns. We can think of these variations as different templates. The process of template generation (with example parameter settings for the pH prediction rules in *FutureSOILS*) and rule generation (as explained in Section 3.3) is given in Figure 2.

In our work, we have generated these rule templates using a custom function which constructively generates rule templates for the pH prediction example presented in Section 3.1. This was done by using

variations of a list of relevant questions that the project wanted to answer and considering different input scenarios, such as varying how many months following liming (e.g. 10-14, 22-26, 34-38) do we want to predict the pH?, what other pre-lime observations to consider other than the starting pH (e.g. Al)? and what management combinations of Liming, Cultivation, etc. to consider?, whether to disregard Cultivation or require no Cultivation treatment?, etc.).

Thus our expert-guided template-based approach to generate multiple rules of similar format, minimises the effort needed to separately specify rules that share the same structure.

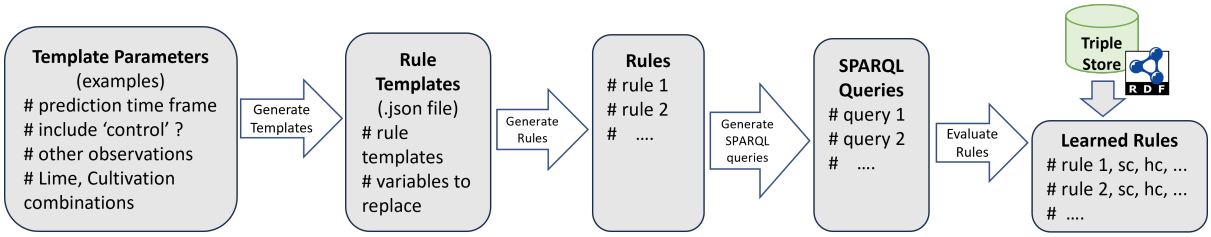

Figure 2: Template and rule generation process

4. Rules for attribute extraction

An extension of the relaxed rules specification can be used for the purpose of attribute extraction, with the introduction of *optional patterns*, that are optional parts of a graph pattern.

$$h \Longleftarrow b_1 \wedge b_2 \wedge \ldots \wedge b_n \wedge \{b_{n+1} \wedge b_{n+2} \wedge \ldots \{b_{m+1} \wedge b_{m+2} \wedge \ldots \wedge b_o\}^* \ldots \wedge b_m\}^* \wedge \sim \{b_{o+1} \wedge b_{o+2} \wedge \ldots \wedge b_q\}^* \quad (5)$$

Atoms enclosed by {...} have the meaning of the SPARQL OPTIONAL keyword[13] and a rule may contain zero or more optional patterns (denoted by *) and can be nested inside other optional patterns. An optional pattern needs to be connected (within the pattern) and at least one atom of the outermost negated graph patterns should be connected to an outside atom(s) which is not part of an optional or negated graph pattern. Nested optional patterns need to be connected to an outer optional pattern. In this form, the head atom may or may not contain an open variable.

Attribute extraction rules can also contain SELECT clauses in addition to the BIND and FILTER clauses. SELECT clauses allow us to declare additional variables to be projected when using attribute extraction rules for training other ML models or data visualisation.

The example below is an attribute extraction rule from the *FutureSOILS* domain. The rule extracts data to train an off-the-shelf numerical model to predict the soil pH value roughly one year following liming. In contrast to the rule template given in Section 3.3 where we referred to the binned numeric entities, here we refer to the numeric attribute values (for example ?*pH_val* instead of ?*pH_bin*, ?*Al_val* instead of ?*Al_bin*, ?*lr_val* instead of ?*lr_bin*, etc.). Note that the rule head contains the not-*closed* variable ?*pH_val_X* that becomes the target value to train models. This example computes the mid-point of the soil depth range as ?*d_pH_avg* and number of months since liming as ?*gap*.

```
SELECT[((?d_pH_to + ?d_pH_from)/2 AS ?d_pH_avg), (?m_X - ?m_lime AS ?gap)]
BIND[]
FILTER[(?m_X - ?m_lime >= 10 && ?m_X - ?m_lime <= 14),
        (?m_pH - ?m_lime <= 2 && ?m_lime - ?m_pH <= 2)]
fs-data:pH__(1-5-CaCl2)__ASC(?X,?pH_val_X) <=
    fs-onto:hasObservation(?eccmo_X,?X),
        sosa:observedProperty(?X,fs-data:ObservedProperty-pH),
```

[13] https://www.w3.org/TR/sparql11-query/#OptionalMatching

```
    sosa:usedProcedure(?X,fs-data:MoA-1-5-CaCl2),
    fs-onto:hasDepth(?X, ?d_pH),
    sosa:phenomenonTime(?X, ?t_X),
    fs-onto:hasMonthCount(?t_X, ?m_X),
fs-onto:hasECCMO(?eccmoByRep_X, ?eccmo_X),
    rdf:type(?eccmoByRep_X, fs-onto:ECCMObyRep),
    fs-onto:hasControl(?eccmoByRep_X, ?eccmo_C),
fs-onto:hasObservation(?eccmo_C, ?obs_pH),
    fs-data:pH__(1-5-CaCl2)__ASC(?obs_pH,?pH_val),
    fs-onto:hasDepth(?obs_pH, ?d_pH),
    fs-onto:from(?d_pH, ?d_pH_from),
    fs-onto:to(?d_pH, ?d_pH_to),
    sosa:phenomenonTime(?obs_pH, ?t_pH),
    fs-onto:hasMonthCount(?t_pH, ?m_pH),
fs-onto:hasObservation(?eccmo_C, ?obs_Al),
    sosa:phenomenonTime(?obs_Al, ?t_pH),
    fs-onto:hasDepth(?obs_Al, ?d_pH),
    fs-data:Aluminium-%-of-Cations__(Calculated)__%__ASC(?obs_Al, ?Al_val),
fs-onto:hasECCMO(?mu_lime, ?eccmo_X),
    fs-onto:hasTime(?mu_lime, ?t_lime),
    fs-onto:hasMonthCount(?t_lime, ?m_lime),
    fs-onto:hasManagement(?mu_lime,?mng_lime),
    rdf:type(?mng_lime, fs-onto:Management-Amelioration-Lime),
    fs-onto:hasLimeApplicationRate(?mng_lime, ?lr_val),
    {fs-onto:hasDepthOfLimePlacement(?mng_lime, ?d_lime)},
{fs-onto:hasECCMO(?mu_culti, ?eccmo_X),
    fs-onto:hasTime(?mu_culti, ?t_culti),
    fs-onto:hasMonthCount(?t_culti, ?m_lime),
    fs-onto:hasManagement(?mu_culti,?mng_culti),
    rdf:type(?mng_culti, fs-onto:Management-Amelioration),
    fs-onto:hasCultivationTreatment(?mng_culti, ?culti),
    skos:broader(?culti, ?culti_lvl),
    fs-onto:index(?culti_lvl, ?culti_idx),
    {fs-onto:hasDepthOfCultivation(?mng_culti, ?d_culti_val)}}
```

The optional pattern in this example enables extraction of data about limed units that were either cultivated or not. The extension enables simple extraction of tabular data by converting the rule to a SPARQL SELECT query. The attribute rules can be generated constructively similar to rule template generation mentioned in Section 3.5. The process of model training by extracting data using attribute extraction rules is given in Figure 3.

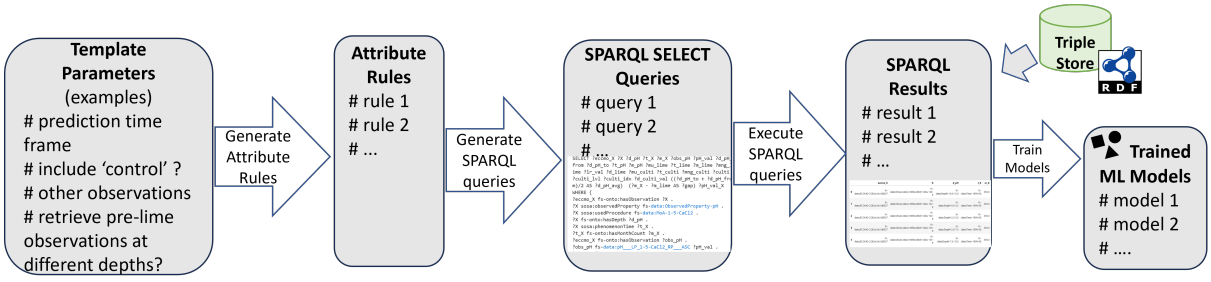

Figure 3: Attribute Rule Generation and Model Training Process

5. Discussion

Here we presented our SPARQL-based relaxed rule specification and extended standard rule quality measures. In summary,

- we allow meaningful and more expressive inductive rules to be defined over non-trivial KGs by proposing a relaxed rule specification that allows rules to contain branches and tree shapes, negated graph patterns and SPARQL style filtering which can make numerical and temporal comparisons,
- and define generalised rule quality measures of Rule Support, Standard Confidence and Head Coverage in order to evaluate the performance of the proposed relaxed rules.

Our approach has the capacity to represent rules with greater expressiveness over KGs with binary predicates. This expressiveness requirement was driven by large-scale application domain modelling, following recommended ontology design patterns and reusing fragments of popular and standard ontologies. We also proposed an expert-guided rule-template-based learning approach to mitigate the issue of the complexity of the search space resulting from the relaxed language bias. Our approach permits expert domain knowledge about the likely relationships in the ontology to be leveraged in the search for good rules. We further extended our relaxed rule specification to include optional patterns which can be used for attribute extraction from KGs to train other ML models. We have not used the optional patterns in the specification of rule templates and generation of variants. Indeed optional patterns in rules have no effect on the application, prediction and quality measures of rules, but may add explanatory value. Ultimately, our rule language, including both the domain-expert-specifiable templates and the automated rule extensions, is expressed in SPARQL. This can be re-expressed in a more conventional rule language as a Datalog sub-language following the translation by Polleres [20].

Since our contribution is mainly a specification, we assess the expressiveness of our proposed relaxed rules against relevant related work in Table 1. As future work, we hope to extend our use-case scenario of *FutureSOILS* to include geographic data (such as from TERN[14]) and adopt our relaxed rule specification to retain compatibility with GeoSPARQL[15]. We also hope to apply our relaxed rules to existing large and complex KGs to demonstrate the scalability of our method and perform a quantitative analysis related to the existing work. We would also like to explore whether large language models could substitute for human domain knowledge to propose initial domain-informed rule templates.

Table 1
Relaxed rule evaluation

Rule-learning approach	Branches and Trees	Numerical relations	Temporal relations	Negated patterns
`Relaxed-Rules (our approach)`	Y	Y	Y	Y
`AMIE [8]`	N	N	N	N
`AnyBURL [11]`	N	N	N	N
`Learning SHACL shapes [12]`	Limited	N	N	N
`Rules with negated atoms [13]`	N	N	N	Limited
`Rules with numeric comp.[15, 14]`	N	Limited	N	N
`Temporal rules [17, 16]`	N	N	Limited	N

Acknowledgments

This work was funded by the Australian Government through the National Landcare Program. We thank FarmLink, NSW Department of Primary Industries, Australian National University, Holbrook

[14]https://www.tern.org.au/
[15]https://www.ogc.org/standard/geosparql/

Landcare Group and Central West Farming Systems. We are grateful to Pouya Ghiasnezhad Omran for many discussions that influenced our work.

References

[1] D. Ratcliffe, K. Taylor, Refinement-based owl class induction with convex measures, in: Semantic Technology: 7th Joint International Conference, JIST 2017, Gold Coast, QLD, Australia, November 10-12, 2017, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2017, p. 49–65. URL: https://doi.org/10.1007/978-3-319-70682-5_4. doi:10.1007/978-3-319-70682-5_4.

[2] L. Bühmann, J. Lehmann, P. Westphal, Dl-learner—a framework for inductive learning on the semantic web, Journal of Web Semantics 39 (2016) 15–24. URL: https://www.sciencedirect.com/science/article/pii/S157082681630018X. doi:https://doi.org/10.1016/j.websem.2016.06.001.

[3] R. Michalski, R. Chilausky, Inductive learning as rule-guided generalization and conceptual simplification of symbolic descriptions: unifying principles and a methodology, in: Papers of the Workshop on Machine Learning, Carnegie-Mellon University, Pittsburgh, 1980.

[4] A. Gubichev, T. Neumann, Exploiting the query structure for efficient join ordering in sparql queries, in: V. Leroy, V. Christophides, V. Christophides, S. Idreos, A. Kementsietsidis, M. Garofalakis, S. Amer-Yahia (Eds.), Advances in Database Technology - EDBT 2014, Advances in Database Technology - EDBT 2014: 17th International Conference on Extending Database Technology, Proceedings, OpenProceedings.org, University of Konstanz, University Library, 2014, pp. 439–450. doi:10.5441/002/edbt.2014.40, 17th International Conference on Extending Database Technology, EDBT 2014 ; Conference date: 24-03-2014 Through 28-03-2014.

[5] H. Wu, Z. Wang, K. Wang, P. G. Omran, J. Li, Rule Learning over Knowledge Graphs: A Review, Transactions on Graph Data and Knowledge 1 (2023) 7:1–7:23. URL: https://drops.dagstuhl.de/entities/document/10.4230/TGDK.1.1.7. doi:10.4230/TGDK.1.1.7.

[6] L. A. Galárraga, C. Teflioudi, K. Hose, F. Suchanek, AMIE: association rule mining under incomplete evidence in ontological knowledge bases, in: Proceedings of the 22nd International Conference on World Wide Web, WWW '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 413–422. URL: https://doi.org/10.1145/2488388.2488425. doi:10.1145/2488388.2488425.

[7] L. Galárraga, C. Teflioudi, K. Hose, F. M. Suchanek, Fast rule mining in ontological knowledge bases with AMIE++, The VLDB Journal 24 (2015) 707–730. URL: https://doi.org/10.1007/s00778-015-0394-1. doi:10.1007/s00778-015-0394-1.

[8] J. Lajus, L. Galárraga, F. Suchanek, Fast and exact rule mining with AMIE 3, in: The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2020, p. 36–52. URL: https://doi.org/10.1007/978-3-030-49461-2_3. doi:10.1007/978-3-030-49461-2_3.

[9] C. Meilicke, M. W. Chekol, D. Ruffinelli, H. Stuckenschmidt, Anytime bottom-up rule learning for knowledge graph completion, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'19, AAAI Press, 2019, p. 3137–3143.

[10] C. Meilicke, M. W. Chekol, M. Fink, H. Stuckenschmidt, Reinforced anytime bottom up rule learning for knowledge graph completion, CoRR abs/2004.04412 (2020). URL: https://arxiv.org/abs/2004.04412. arXiv:2004.04412.

[11] C. Meilicke, M. W. Chekol, P. Betz, M. Fink, H. Stuckeschmidt, Anytime bottom-up rule learning for large-scale knowledge graph completion, The VLDB Journal 33 (2024) 131–161. URL: https://doi.org/10.1007/s00778-023-00800-5. doi:10.1007/s00778-023-00800-5.

[12] P. Omran, K. Taylor, S. Rodríguez Méndez, A. Haller, Learning shacl shapes from knowledge graphs, Semantic Web 14 (2022) 1–21. doi:10.3233/SW-223063.

[13] V. T. Ho, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, G. Weikum, Rule learning from knowledge graphs guided by embedding models, in: D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa,

V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, E. Simperl (Eds.), The Semantic Web – ISWC 2018, Springer International Publishing, Cham, 2018, pp. 72–90.

[14] S. Ortona, V. V. Meduri, P. Papotti, Rudik: Rule discovery in knowledge bases, Proc. VLDB Endow. 11 (2018) 1946–1949. URL: https://api.semanticscholar.org/CorpusID:51989311.

[15] P.-W. Wang, D. Stepanova, C. Domokos, J. Z. Kolter, Differentiable learning of numerical rules in knowledge graphs, in: 8th International Conference on Learning Representations (ICLR-20), 2020. URL: https://openreview.net/forum?id=rJleKgrKwS.

[16] Y. Liu, Y. Ma, M. Hildebrandt, M. Joblin, V. Tresp, Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs, Proceedings of the AAAI Conference on Artificial Intelligence 36 (2022) 4120–4127. doi:10.1609/aaai.v36i4.20330.

[17] P. G. Omran, K. Wang, Z. Wang, Learning temporal rules from knowledge graph streams, in: AAAI Spring Symposium Combining Machine Learning with Knowledge Engineering, 2019. URL: https://api.semanticscholar.org/CorpusID:135466806.

[18] S. J. Rodríguez-Méndez, P. G. Omran, K. Taylor, P. Kan-John, T. Gedeon, A.-M. Farley, The "FutureSOILS Project": A Knowledge Graph-based Predictive Model for Improving Soil Treatments in Agriculture, in: 2022 Australasian Computer Science Week (2022 ACSW), 2022. URL: https://acsw.core.edu.au/2022-home/2022-posters-2.

[19] N. Noy, A. Rector, Defining N-ary Relations on the Semantic Web, W3C Note, W3C, 2006. URL: https://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/.

[20] A. Polleres, From SPARQL to rules (and back), in: Proceedings of the 16th International Conference on World Wide Web, WWW '07, Association for Computing Machinery, New York, NY, USA, 2007, p. 787–796. URL: https://doi.org/10.1145/1242572.1242679. doi:10.1145/1242572.1242679.