Synthetic Datasets and Evaluation Tools for Inductive Neural Reasoning

Cristina Cornelio[1⋆] and Veronika Thost[2⋆]

[1] Samsung AI Centre - Cambridge c.cornelio@samsung.com
[2] IBM Research, MIT-IBM Watson AI Lab vth@zurich.ibm.com

Abstract. Logical rules are a popular knowledge representation language in many domains. Recently, neural networks have been proposed to support the complex rule induction process. However, we argue that existing datasets and evaluation approaches are lacking in various dimensions; for example, different kinds of rules or dependencies between rules are neglected. Moreover, for the development of neural approaches, we need large amounts of data to learn from and adequate, approximate evaluation measures. In this paper, we provide a tool for generating diverse datasets and for evaluating neural rule learning systems, including novel performance metrics.

1 Introduction

Logical rules are a popular knowledge representation language in many domains. They represent domain knowledge, encode information that can be derived from given facts in a compact form, and allow for logical reasoning. For example, given facts $parent(ann, bob)$ and $parent(bob, dan)$, the simple *Datalog* rule $grandparent(X, Z) \text{ :- } parent(X, Y), parent(Y, Z).$ encodes not only the fact $grandparent(ann, dan)$ but also describes its dependency on the other facts. Moreover, if the data grows and new facts are added, we can automatically derive new knowledge. Since rule formulation is complex and requires domain expertise, *rule learning* [25, 11] has been an area of active research in AI for a long time, also known as *inductive logic programming* (ILP). It has recently revived with the increasing use of knowledge graphs (KGs), which can be considered as large fact collections. KGs are used in the Semantic Web, the medical domain, and companies such as Google [7] and Amazon [16]. Useful rules over these knowledge bases would obviously provide various benefits. While the development of neural ILP systems is still in its early stages, we argue that current evaluations are insufficient. We demonstrate that the reported results are questionable, especially, in terms of generalization and because the datasets are lacking in various ways.

The evaluation of rule learning has changed over time. While the classical rule learning methods often focused on tricky problems in complex domains [14, 24] and proved to be effective in practical applications, current evaluations can roughly be divided into three categories. (1) Some consider very small example

⋆ Equal contribution. The work was partly conducted while C.C. was at IBM Research.

Table 1: Overview of dataset generators for Datalog rules

Name	Rules	Facts	Predicates
Family Tree [6]	5 hand written	arbitrary	9
General Graph [6]	5 hand written	arbitrary	7
On-the-fly [15]	5 templates (1 per dataset)	max 2194	5
GraphLog [30]	no rules with same body; no rules with predicates shared among head and body	arbitrary	arbitrary
RuDaS	full Datalog expressivity	arbitrary	arbitrary

problems with usually less than 50 facts and only few rules to be learned [10, 27, 18]. Often, these problems are *completely* defined, in the sense that all facts are classified as either true or false, or there are at least some negative examples given. Hence, the systems can be thoroughly evaluated based on classical measures, such as accuracy. (2) Others regard (subsets of) real KGs such as Wikidata[3] or DBpedia[4], some with millions of facts [12, 22, 13, 33]. Since there are no rules over these KGs, the rule suggestions of the systems are usually evaluated using metrics capturing the precision and coverage of rules based on the facts contained in the KG, such as standard confidence [12]. However, since the KGs are generally incomplete, the quality of the rule suggestions is not fully captured in this way. For instance, [22] present an illustrative example rule, $gender(X, male)$:- $isCEO(X, Y), isCompany(Y)$, which might well capture the facts in many existing KGs but which is heavily biased and does not extend to the entirety of valid facts beyond them. Furthermore, we cannot assume that the few considered KGs completely capture the variety of existing domains and especially the rules in them. For example, [17] propose rules over WordNet[5] that are of very simple nature – containing only a small number of the predicates in WordNet and having only a single body atom – and very different from the ones suggested in [12] for other KGs. Also the evaluation metrics vary, especially considering the intersection between more modern and classic approaches. We will show that most of the standard information retrieval measures used in machine learning are not adequate for a logic context because they neglect important facets like the size of the Herbrand universe[6] (e.g., this may yield too high accuracy). Some other measures have been used for neural ILP such as mean reciprocal rank, or precision/recall@k, but they can only be applied in specific cases (i.e., if the system outputs weighted/probabilistic rules or a ranking of facts). Yet, strict logic measures are not perfect either, since they are based on the assumption that the domain is very small and human understandable.

[3] https://www.wikidata.org/wiki/Wikidata:Main_Page
[4] https://wiki.dbpedia.org/
[5] https://wordnet.princeton.edu/
[6] We assume the reader to be familiar with first-order logic (FOL). See also Section 2 and Appendix A.

For this reason, the community should consider several metrics and define new metrics suitable for both worlds.
(3) Recent works have proposed synthetic, generated datasets but, as Table 1 shows, they are very simple and specifically restricted regarding the rules. There are well-known ILP competitions in the logic community,[7] which also provide synthetic datasets, but their evaluations are usually based only on test facts and not on rules. Further, benchmarks in the database community are related in that they cover schemata and rules. However, either their use cases and thus the kinds of rule sets considered are rather restricted (e.g., for schema mapping, there are rules from source to target schema, but the rules do not depend on each other in the sense that one is to be applied after the other [1, 2]), or there is a number of fixed test scenarios [3]. Further, database data is usually curated, and the benchmarks were developed to evaluate customized algorithms.

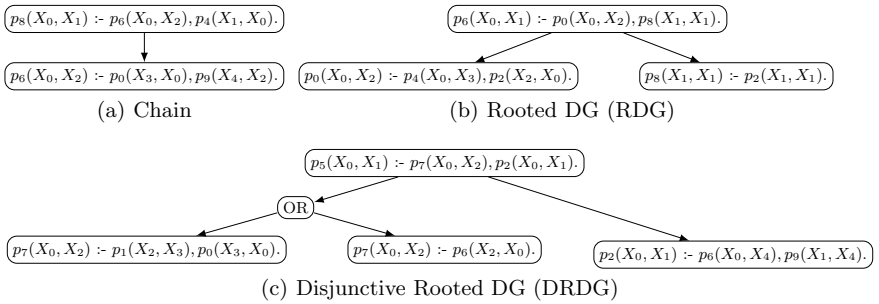
In summary, we claim that the existing evaluation approaches are not adequate for modern ILP. In particular, recently, inductive (logical) reasoning has come into focus in the machine learning community [26, 31], including neural approaches to ILP [35, 10, 18, 36]. However, we will show that these first neuro-symbolic systems cannot compete with well-engineered purely symbolic systems such as AMIE [12] yet.[8] In order to support the development of the former, our work – in contrast to the aforementioned, more traditional ILP evaluations – focuses on more arbitrary data, on learning systems needing large amounts of data to learn from (vs. a few, fixed test sets), and on approximate solutions.

In this paper, (1) we extend the categorization of rule learning datasets beyond the numbers of constants, predicates, and facts. In particular, we propose to consider the amount/type of noise (i.e., wrong or missing facts) and (in)completeness (i.e., share of consequences of rules present in the data). (2) We present RuDaS (**S**ynthetic **Da**tasets for **Ru**le Learning), a tool for generating synthetic datasets containing both facts and rules, and for evaluating neural rule learning systems, that overcomes the above mentioned shortcomings of existing works and offers complementary evaluation methods. RuDaS is parameterizable in the standard and in the new categories, and thus allows for a more fine-grained analysis of rule learning systems. (3) It also supports this analysis by computing classical and more recent metrics, including two new ones that we introduce. (4) Our experiments show that the datasets and evaluation measures provided by RuDaS help revealing the variety in the capabilities of existing ILP systems, and thus support and help researchers in developing and optimizing new, existing, and especially neural approaches. RuDaS is available at https://github.com/IBM/RuDaS.

2 Rule Learning Preliminaries

We assume the reader to be familiar with basic first-order logic (FOL) concepts (e.g., inference). We consider Datalog *rules* [5]: α_0 :- $\alpha_1, \ldots, \alpha_m$. (1) of *length*

[7] For example: 2016, http://ilp16.doc.ic.ac.uk/competition
[8] A fact well-concealed in the papers, by simply ignoring symbolic competitors.



(a) Chain (b) Rooted DG (RDG)

(c) Disjunctive Rooted DG (DRDG)

Fig. 1: Example rule structure generated for the different categories with depth 2.

$m \geq 1$ where all *atoms* α_j, $0 \leq j \leq m$, are of the form $p(t_1, \ldots, t_n)$ with a predicate p of arity $n \geq 1$ and terms t_k, $1 \leq k \leq n$. A *term* is either a constant or a variable. α_0 is called the *head* and the conjunction $\alpha_1, \ldots, \alpha_m$ the *body* of the rule. All variables that occur in the head must occur in the body. A *fact* is an atom not containing variables.

Note that several classical ILP systems also consider more complex function-free Horn rules, (allowing existential quantification in the rule head or negation in the body), but most recent systems focus on datalog rules or restrictions of those [12, 10, 27]. In particular, most reasoning systems for KGs [35, 22] consider only binary predicates and *chain rules* of form:

$$p_0(X_1, X_{m+1}) \coloneq p_1(X_1, X_2), \ldots, p_m(X_m, X_{m+1}) \ . \tag{2}$$

We define the problem of *rule learning* in the most general way: given background knowledge in the form of facts, including a set of so-called *positive examples* (vs. *negative* or *counter*-examples), the goal is to learn rules that can be used to infer the positive examples from the background knowledge, based on standard FOL semantics. Intuitively, the positive examples are the consequences of the rules to be learned. As common, we do not separate the background knowledge into two types of facts but consider a single fact set as input. We recall that the *closed-world assumption* (CWA) (vs. *open world assumption* or OWA) states that all facts that are not explicitly given as true are assumed to be false.

3 RuDaS Datasets

RuDaS contains an easy-to-use generator for ILP datasets that generates datasets that vary in many dimensions and is highly parameterizable in a detailed set of metrics which can serve as a general classification scheme for ILP datasets and support evaluations. In a nutshell, we first generate a set of rules and a set of base facts, and then derive the *consequences* of the latter w.r.t. the rules. Note that, in contrast to existing approaches which are largely template-based, the RuDaS generation process includes much randomness. We also add noise and

remove some facts to make it realistic, details are given below. A full description of the generation process is given in Appendix B. Each generated dataset contains the rules and the facts in files in standard Prolog format. In this section, we describe the datasets including concrete examples, which can be found in our repository.

Symbols. Our datasets are domain independent, which means that we consider synthetic names p_i for predicates, c_i for constants, and X_i for variables with $i \geq 0$. While the kinds and numbers of the symbols used is random, it can be controlled by setting the generator parameters of the number of constants/predicates and min/max arity of predicates. Observe that these numbers influence the variability and number of generated rules and facts.

Rules. RuDaS datasets contain Datalog rules [5] (see also Appendix 2) of variable structure. The generation is largely at random in terms of which predicates, variables, and constants appear in the rules; that is, in the structure of every single rule. We only require the head to contain at least one variable.

We propose four categories of set of rules depending on the dependencies between them: *Chain*, *Rooted Directed Graph (DG)*, *Disjunctive Rooted DG*, and *Mixed*. Figure 1 shows a generated rule set for each category. The dependencies between the rules are represented as edges in a directed graph where the rules are the nodes. That is, an incoming edge shows that the facts inferred by the child node's rule might be used, during inference with the rule at the parent node. The node at the top is called the *root*. In the following, we use (rule) graph and DG interchangeably.

Category Chain. Each rule, except the one at the root, infers facts relevant for exactly one other rule (i.e., every node has at most one parent node) and, for each rule, there is at most one such other rule which might infer facts relevant for the rule (i.e., every node has at most one child node). However, recursive rules (where the predicate in the head occurs also in the body) represent an exception, they are relevant for themselves and for one other rule.

Category Rooted DG (RDG). It generalizes category Chain in that every rule can be relevant for several others (i.e., each node can have multiple parent nodes). Furthermore, for each rule, there may be several other rules which might infer facts relevant for the rule (i.e., a node may have several child nodes); and at least one such case exists. But, for each predicate occurring in the body of the former rule, there must be at most one other rule with this predicate in the head; that is, there are no alternative rules to derive facts relevant for a rule w.r.t. a specific body atom.

Category Disjunctive Rooted DG (DRDG). It generalizes category RDG by allowing for the latter alternative rules (represented as children of an "OR" node); and at least one such case exists.

Category Mixed. A rule graph that contains connected components of different of the above categories.

Figure 1 illustrates the differences between the categories. The numbers and categories of connected components are selected randomly by default. The shape of RuDaS rule sets can be influenced with the following parameters: number and

maximal length of rules; category of connected components; min/max number of connected components; and maximal depth of rule graphs (i.e., number of rules nodes in the maximum of the shortest paths between root and leaves).

Facts. The main advantage of the RuDaS datasets, the availability of the rules, allows for classifying the facts as well. More specifically, facts can be *(ir)relevant* for inference, depending on if their predicates do (not) occur in a rule body, and they may be consequences of inferences. Such a classification of facts is impossible for all the existing datasets that do not contain rules, but allows for a better evaluation of the rule learners' capabilities (see Section 5).

RuDaS fact sets vary in the following parameters: dataset size (S, M, L, XL); open-world degree $n_{\text{OW}} \in [0, 1]$; and amount of noise in the data $n_{\text{Noise+}}$, $n_{\text{Noise-}} \in [0, 1]$. An S dataset contains about 50-100 facts, an M dataset about 101-1,000, an L dataset about 1,001-10,000, and an XL dataset $>$ 10,000. Larger sizes are possible as well, however, since the main purpose of RuDaS is allowing the analysis of the rules learned (vs. scalability), we have not considered them so far. The open-world degree n_{OW} specifies how many of the consequences from an initial set of relevant facts, called *support* facts, are missing in the dataset (see Appendix B for more details). By noise, we mean facts that are not helpful in learning the rules either because they are not relevant for deriving the positive examples ($n_{\text{Noise+}}$) or because they are relevant but missing ($n_{\text{Noise-}}$).

Observe that our datasets are based on rules, as required for ILP, but can be applied for evaluating the closely related and very popular link prediction systems as well; specifically, auxiliary information provided with the datasets contains the missing consequences which can be used as test facts in that context.

Example Set of Datasets: RuDaS.v0. For demonstration purposes, we generated a set of datasets readily available for the community, and which we also used in our experiments (Section 5, for dataset details and statistics see Table 5 in Appendix C). The datasets model different possible scenarios, and mainly vary in the structures and sizes of the rule sets and in the sorts and quantities of facts. RuDaS.v0 contains 40 Chain, 78 RDG, and 78 DRDG datasets, of sizes S and M, and of depths 2 and 3, all evenly distributed. Note that each of the rules sets consists of exactly one connected component, and that we did not generate rule sets of category Mixed; Mixed datasets with connected components of possibly different categories can be easily created by combining our generated datasets. Further, we constrained both the maximal rule length and arity of atoms to two for practicality, because several existing rule learning systems require that. All the datasets were generated such that they are missing 20-40% of all consequences, 15-30% of the original support facts, and contain 10-30% facts that are irrelevant for the derivation of positive examples. Since real datasets may strongly vary in the numbers of missing consequences and noise and, in particular, since these numbers are generally unknown, we chose factors seeming reasonable to us. Also note that there is information regarding the accuracy of real fact sets such as YAGO[9] (95%) and NELL[10] (87%),

[9] https://github.com/yago-naga/yago3
[10] http://rtw.ml.cmu.edu/rtw/overview

that measures the amount of data correctly extracted from the Web etc. and hence corresponds to $1 - n_{\text{Noise+}}$ in our setting. Thus, our choices in this regard thus seem to be realistic. We hence simulated an open-world setting and incorporated noise. While we consider this to be the most realistic training or evaluation scenario, specific rule learning capabilities might be better evaluated in more artificial settings with either consequences or noise missing. Therefore, every dataset additionally includes files containing the incomplete set of facts without noise (i.e., $n_{\text{Noise+}} = 0$, $n_{\text{Noise-}} = 0$) and the complete fact set (i.e., $n_{\text{OW}} = 0$), with and without noise.

4 Evaluation Tools

RuDaS also contains an evaluator for comparing the rules produced by a rule learning system to the original rules in a given datase. We focus on three logic(-inspired) distances and four standard information retrieval measures that are relevant to our goal of capturing rule learning performance: 1) Herbrand distance, the traditional distance between Herbrand models; two normalized versions of the Herbrand distance 2) Herbrand accuracy (H-accuracy) and 3) Herbrand score (H-score), a new metric we propose in this paper; 4) accuracy 5) precision; 6) recall; and 7) F1-score. See Appendix A for preliminaries on Herbrand models.

Our test fact sets (both base facts and consequences) in the evaluation do not contain noise and all the consequences can be recovered by the original rules applied over the given facts. In line with that, we focus on measures that maintain the closed-world assumption, and do not consider measures that focus on the open-world aspect for evaluation (e.g., PCA in [12]). Our experiments will show that F1-score suits best the open-world evaluation over RuDaS datasets.

In what follows, $I(R, F)$ denotes the set of facts inferred by grounding the rules R over the support facts F excluding the facts in F. We denote an original rule set by $\mathcal{R}$, a learned one by $\mathcal{R}'$, and support facts by $\mathcal{F}$. Our evaluation is performed comparing two sets: 1) $I(\mathcal{R}', \mathcal{F})$ obtained by the application of the induced rules $\mathcal{R}'$ to the fact sets $\mathcal{F}$ using a forward-chaining engine (available in our tool); 2) $I(\mathcal{R}, \mathcal{F})$: the result of applying $\mathcal{R}$ to $\mathcal{F}$.[11]

Logic Measures. The *Herbrand distance* h_d between two logic programs (sets of rules), defined over the same set of constants and predicates, describes the number of facts in which the minimal Herbrand models of the programs differ:

$$h_d(\mathcal{R}, \mathcal{R}', \mathcal{F}) := |[I(\mathcal{R}, \mathcal{F}) \cup I(\mathcal{R}', \mathcal{F})] \setminus [I(\mathcal{R}, \mathcal{F}) \cap I(\mathcal{R}', \mathcal{F})]|.$$

The *standard confidence* s_c [12] is the fraction of correctly inferred facts w.r.t. all facts that can be inferred by the learned rules capturing their precision:

$$s_c(\mathcal{R}, \mathcal{R}', \mathcal{F}) := \frac{|I(\mathcal{R}, \mathcal{F}) \cap I(\mathcal{R}', \mathcal{F})|}{|I(\mathcal{R}', \mathcal{F})|}$$

[11] Note that $\mathcal{F} = \mathbb{S}'$ and $\mathbb{C}' = I(\mathcal{R}, \mathcal{F})$ given $\mathbb{S}'$ and $\mathbb{C}'$ as described in Appendix B - Output.

In our closed-world setting, this corresponds to the precision of a model, since it is easy to see that $|I(\mathcal{R},\mathcal{F}) \cap I(\mathcal{R}',\mathcal{F})|$ are the number of true positive examples and $|I(\mathcal{R}',\mathcal{F})|$ corresponds to the union of true and false positive examples. The *Herbrand accuracy* h_r corresponds to the Herbrand distance normalized on the Herbrand base: $h_r(\mathcal{R},\mathcal{R}',\mathcal{F}) := 1 - \frac{h_d}{u}$, where u is the size of the Herbrand base defined by the original program. We introduce a new metric, the *Herbrand score* (H-score) defined as:

$$\text{H-score}(\mathcal{R},\mathcal{R}',\mathcal{F}) : = \frac{|I(\mathcal{R},\mathcal{F}) \cap I(\mathcal{R}',\mathcal{F})|}{|I(\mathcal{R},\mathcal{F}) \cup I(\mathcal{R}',\mathcal{F})|} = 1 - \frac{h_d(\mathcal{R},\mathcal{R}',\mathcal{F})}{|I(\mathcal{R},\mathcal{F}) \cup I(\mathcal{R}',\mathcal{F})|}$$

H-score provides an advantage over the other metrics since it captures both how many correct facts a set of rules produces and also its completeness (how many of the facts inferred by the original rules $\mathcal{R}$ were correctly discovered), while the other measures consider these points only partially.

Note that Herbrand accuracy is not a significant measure if $\mathcal{F}$ or the Herbrand base is large, because, in these cases, it will be very high (close to 1) disregarding the quality of the rules. This happens because all the facts in $\mathcal{F}$ are considered correct predictions, also the facts in the Herbrand base that neither appear in $I(\mathcal{R},\mathcal{F})$ nor in $I(\mathcal{R}',\mathcal{F})$.

Information Retrieval Measures. We adapted the main measures used in the machine learning evaluations to a logic context. We define: the sets of true positive examples (TP) as the cardinality $|I(\mathcal{R},\mathcal{F}) \cap I(\mathcal{R}',\mathcal{F})|$, the set of false positive examples (FP) as the cardinality $|I(\mathcal{R},\mathcal{F}) \setminus I(\mathcal{R}',\mathcal{F})|$; the set of false negative examples (FN) as the cardinality $|I(\mathcal{R}',\mathcal{F}) \setminus I(\mathcal{R},\mathcal{F})|$; and the set of true negative examples (TN) as the cardinality of the difference between the Herbrand base and the union $I(\mathcal{R},\mathcal{F}) \cup I(\mathcal{R}',\mathcal{F})$. Given these four definitions, accuracy, precision, recall, F1-score etc. can be defined as usual [28].

Note that accuracy is not a significant measure if $\mathcal{F}$ or the Herbrand base is large, for the same reason mentioned with Herbrand accuracy above. Moreover, F1-score is similar to H-score, with the difference that F1-score gives more priority to the TP examples. We believe that giving uniform priority to FN, TP, and FP is more reasonable in the context of logic; this is in line with standard logic measures like h_d. However, F1-score (compared to H-score) better suits open-world settings where some of the consequences could be missing, and thus count as FP (despite being correct). For this reason F1 would give a better estimate of the quality of the induced rules since it focuses more on the TP examples and give less priority to the generated FP examples.

We observe that, if $I(\mathcal{R},\mathcal{F}) = I(\mathcal{R}',\mathcal{F})$, then H-score is equal to precision and both are equal to 1; and, if $I(\mathcal{R},\mathcal{F})$ and $I(\mathcal{R}',\mathcal{F})$ are disjoint, then both are 0. The two measures coincide if $I(\mathcal{R},\mathcal{F}) \subseteq I(\mathcal{R}',\mathcal{F})$. The main difference between the two is highlighted in the case where $I(\mathcal{R}',\mathcal{F}) \subseteq I(\mathcal{R},\mathcal{F})$. Then, precision $= 1$ but H-score is < 1. We designed our new H-score this way because we want to have an H-score of 1 only if the predicted facts are exactly those produced by the original rules while precision is 1 if all predicted facts are correct.

Rule-based Measures. Several distance metrics between two sets of logic rules have been defined in the literature [8, 9, 21, 23, 29]. However, most of them

strongly rely on the parse structure of the formulas and hence are more suitable for more expressive logics. For this reason, we propose a new metric, the *Rule-score (R-score)*, which is tailored to rules, in that it calculates a distance d_R between the rules that have the same head predicate and, for the latter, computes the pairwise distances d_A between the two rules' atoms. Note that, below, we consider datalog rules without constants and with only binary predicates for simplicity, but the definitions can be easily extended to non-binary atoms. An idea of how to deal with constants in addition is suggested in [9], for example.

Our distance d_A between two non-ground, binary atoms takes into account a specific mapping $\omega \in \Omega$ between their variables so that we can later lift it to rule level; Ω is the set of all possible *variable re-namings* between two sets of variables names (in our case from two rules). Given atoms $a_1 = p(X_1, X_2)$ and $a_2 = q(Y_1, Y_2)$ and such a mapping ω, $d_A(a_1, a_2, \omega) = 1$ if their predicates differ, $d_A(a_1, a_2, \omega) = 0$ if $\omega(X_1) = Y_1$ and $\omega(X_2) = Y_2$, otherwise:

$$d_A(a_1, a_2, \omega) = \frac{1}{4} \sum_{i=1}^{2} \mathbb{1}^c(\omega(X_i) = Y_i) \ .$$

where the *complement indicator function* $\mathbb{1}^c(e)$ of an event e is equal to 0 if the event is satisfied and 1 otherwise.

Note that d_A is based upon the main, and most used, distance metric between two *ground* atoms, the *Nienhuys-Cheng distance* [20]. In fact, it only differs from the latter in that it is defined for non-ground atoms using a variable re-naming.

Consider two rules r_1 and r_2 as follows:
r_1 : $p_1(A, B) \coloneq p_2(A, A), p_3(B, B), p_4(A, B)$.
r_2 : $p_1(X, X) \coloneq p_2(Y, X), p_2(X, X)$.
Re-namings $\omega_1 = \{A : X, B : Y\}$ and $\omega_1 = \{A : Y, B : X\}$ yield: 1) r_1 with ω_1: $p_1(X, Y) \coloneq p_2(X, X), p_3(Y, Y), p_4(X, Y)$. and 2) r_1 with ω_2: $p_1(Y, X) \coloneq p_2(Y, Y)$, $p_3(X, X)$, $p_4(Y, X)$. Let h_i and h_2 denote the head atoms of r_1 and r_2, respectively. Then, we get $d_A(h_1, h_2, \omega_1) = 0.25$ and $d_A(h_1, h_2, \omega_2) = 0.25$.

Intuitively, our distance d_R between two rules r_1 and r_2 considers matches between the rules, where a match consists of a pairing between their atoms together with a variable renaming, and takes the best match as distance (averaged over the number of atoms). More specifically, such a *pairing* for r_1 and r_2 is a set of pairs such that the first component is a body atom from r_1 and the second from r_2. It contains $\max(|b(r_1)|, |b(r_2)|)$ pairs ($|b(r_i)|$ denotes the number of atoms in the body of r_i). To represent a match, we require the atoms to have the same predicate and additionally allow for an empty placeholder atom (denoted by $-$), extending d_A such that it has maximal distance 1 to any other atom. The placeholder also accounts for the case that the rules are of different length.

For the rules of the previous example, there are two pairings:
a) $c_1 = \{(p_2(A, A),\ p_2(Y, X)),\ (p_3(B, B), -),\ (p_4(A, B), -)\}$;
b) $c_2 = \{(p_2(A, A),\ p_2(X, X)),\ (p_3(B, B), -),\ (p_4(A, B), -)\}$.

We now define the distance $d_R(r_1, r_2)$ between two rules r_1 and r_2 as:

$$\frac{1}{n_a} \min_{\omega \in \Omega} \Big(d_A(h_1, h_2, \omega) + \min_{c \in \mathcal{C}} \sum_{(a_1,a_2)\in c} d_A(a_1, a_2, \omega) \Big)$$

where $n_a = \max(|b(r_1)|, |b(r_2)|) + 1$ is the number of atoms of the rule with more atoms; h_i is the head atom of rule r_i; Ω is the set of all possible variable renamings between the variable names in the two rules; $\mathcal{C}$ is the set of all possible pairings for r_1 and r_2.

For the re-namings ω_1 and ω_2 and pairings c_1 and c_2 from the previous example, d_R represents the minimum of $0.25 + \sum_{(a_1,a_2)\in c_i} d_A(a_1, a_2, \omega_j)$, with the sums: 1a) $0.25+1+1 = 2.5$; 2a) $0.25+1+1 = 2.5$; 1b) $0+1+1 = 2.25$; 2b) $0.5+1+1 = 2.75$. Given that $n_a = 4$, we obtain $d_R(r_1, r_2) = 2.25/4 = 0.5625$.

Finally, our novel metric *Rule-score (R-score)* for two logic programs, the original program $\mathcal{R}$ and the induced program $\mathcal{R}'$ (rules of form (2)), is defined as:

$$\text{R-score}(\mathcal{R}, \mathcal{R}') = 1 - \frac{1}{|\mathcal{R}|} \Big(\sum_{r_1 \in \mathcal{R}} \min_{r_2 \in \mathcal{R}'[hp(r_1)]} d_R(r_1, r_2) \Big)$$

where the function $hp(r)$ corresponds to the head predicate of a given rule r; $\mathcal{R}[p]$ denotes the rules in $\mathcal{R}$ with head predicate p.

5 Experiments

The goal of our experiments is to demonstrate the need for a portfolio of diverse datasets for evaluating rule learning systems. As mentioned above, the existing datasets are not sufficient to comprehensively evaluate ILP methods (e.g., often fall into category Chain). Note that the purpose of our experiments is not to provide an exhaustive analysis of existing ILP systems, but to show that this kind of analysis should be part of system development, by pointing out that important aspects of rule learning that have been ignored so far may considerably impact system performance. Our evaluation does not solely focus on neuro-symbolic approaches, but includes ILP systems that are representatives for the different methodologies adopted by researchers during the years to approach the problem of rule learning. We will see that these systems represent strong baselines. We compared the following systems (configuration details in the appendix): 1) FOIL [24], a traditional ILP system; 2) AMIE+ [12], a rule mining system; 3) Neural-LP [35]; and 4) NTP [27]. The latter are both neural approaches. AMIE+, Neural-LP, and NTP output confidence scores for the learned rules. We therefore filtered their output using a system-specific threshold, obtained using grid search over all datasets. To not disadvantage Neural-LP and NTP, which use auxiliary predicates, we ignored the facts produced on those.[12]

[12] Notice that NTP requires additional information in the form of rule templates, obviously representing an advantage.

Table 2: Impact of different metrics, each averaged on 120 datasets with uniformly distributed categories $\in$ {CHAIN, RDG, DRDG}, sizes $\in$ {S,M}, and graph depths $\in$ {2,3}; $n_{\text{OW}} = 0.3$, $n_{\text{Noise-}} = 0.2$, $n_{\text{Noise+}} = 0.1$.

	FOIL	AMIE+	Neural-LP	NTP
H-accuracy	0.9873	0.8498	0.9850	0.9221
Accuracy	0.9872	0.8494	0.9849	0.9219
F1-score	0.2151	0.3031	0.1621	0.1125
H-score	0.1520	0.2321	0.1025	0.0728
Precision	0.5963	0.2982	0.1687	0.1021
Recall	0.2264	0.7311	0.2433	0.3921
R-score	0.2728	0.3350	0.1906	0.1811

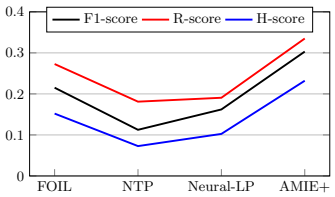


Fig. 2: Visualization shows metric similarity.

We evaluated the systems over RuDaS.v0 (see Section 3 and Table 5 in the appendix) in four main experiments to understand, respectively, the variety of the performance metrics, and the impact of missing consequences, noise, rule dependencies, and dataset size. We set a time limit of 24h both for system executions and evaluations.[13] However, we did not penalize the instances that exceeded the limit since we are interested in the rules that can be learned. Generally, the runtimes varied greatly over the datasets, were often surprisingly long (AMIE, NTP can take hours; only Neural-LP usually terminated within seconds), and provide not much insight. We do not report the standard deviation since the results span over different dataset categories and sizes.

5.1 Overall Results: Comparing Different Metrics

In this experiment, we regard the overall results, reported in Table 2, in terms of the metrics introduced in Section 4. As expected, the results for F1-score and Herbrand score are very similar, the only difference is that F1-score is a more "optimistic" measure, giving advantage to the methods with a higher number of true positive examples. The results for R-score are in line with these metrics and, although translated, follow especially close the trend of H-score (see Figure 2 for a visualization of the comparison). Hence, R-score indeed represents a valid alternative with the additional advantage of computational efficiency, since it does not require the computation of the induced facts $I(\mathcal{R}', \mathcal{F})$.

Also Herbrand accuracy and accuracy provide similar results. Observe that these two measures are not meaningful in our settings since they yield always very high performances. Note that precision and H-score are very close for AMIE+, Neural-LP, and NTP, but not for FOIL. This could be explained by the fact that the training of the former systems maximizes functions that are similar to precision, while FOIL uses heuristics to produce the rules that induce the

[13] If a system learns very many (usually wrong) rules, the computation of measures based on a closure may become unfeasible.

maximum number of facts in the training set and minimum number of facts not in the training set. The great discrepancy between the two measures with FOIL means that the rules it learns do not produce many false facts but only a subset of the facts induced by the original rules. For AMIE+ instead, since precision and H-score are similar, we have that its rules produce most of the consequences of the original rules and, thanks to the good performance, they do not produce too many false facts. Considering Neural-LP and NTP the two measures are also very similar, but very low: their rules produce most of the positive examples but also a lot of false facts. We report H-score, but the R-scores we computed for comparison showed the same trends.

5.2 Impact of Missing Consequences and Noise

In this experiment, we evaluated the performance of the systems in the presence of complete information, incomplete information, and incomplete information with noise. This was performed analyzing the impact of the different parameters given in RuDaS: n_{OW}, $n_{\text{Noise+}}$, and $n_{\text{Noise-}}$. The results are reported in Table 3. The noise parameters are defines as follows (where the set memberships are intended to mean "uniformly distributed over"): *complete* datasets $n_{\text{OW}} = 0$, $n_{\text{Noise-}} = 0$, and $n_{\text{Noise+}} = 0$, *incomplete* datasets $n_{\text{OW}} \in \{0.2, 0.3, 0.4\}$, $n_{\text{Noise-}} = 0$, and $n_{\text{Noise+}} = 0$, and *incomplete + noise* datasets $n_{\text{OW}} \in \{0.2, 0.3, 0.4\}$, $n_{\text{Noise-}} \in \{0.15, 0.3\}$, and $n_{\text{Noise+}} \in \{0.2, 0.3\}$. Moreover, in order to give an impression of some of the datasets considered in existing evaluations, we included one manually created dataset, EVEN, inspired by the corresponding dataset used in [10][14], which contains complete information. We notice that FOIL shows best performance if the information is exact and complete but considerably worse performance in more noisy scenarios. This is a result of the assumptions FOIL is based upon: it assumes negative examples to be given in addition in order to guide rule learning and, in particular, missing facts to be false (see Section 4.1 in [24]). AMIE+ seems to perform constant on average, showing robustness to noise and incomplete data in all the datasets. Neural-LP and NTP seem to be robust to noise and incomplete data but yield much worse performance generally.

5.3 Impact of Dependencies Between Rules

In this experiment, we analyze the impact of the kind of the dependencies between rules. The results are reported in Table 4. As expected, the systems perform very different depending on the datasets' rule categories. We notice that the systems perform better on the Chain datasets while only learning partially RDG and DRDG rules, meaning that the available rule learning systems are not yet able to capture complex rule set structures.

[14] In our version, $even(X)$:- $even(Z), succ(Z, Y), succ(Y, X)$ is the only rule, and the input facts are such that we also have an accuracy of 1 if the symmetric rule $even(Z)$:- $even(X), succ(Z, Y), succ(Y, X)$ is learned (using the original fact set it would be 0). AMIE+ and Neural-LP do not support the unary predicates in EVEN.

Table 3: Effect of missing consequences and noise on 144 datasets. Each H-score value is averaged on 48 datasets, with uniformly distributed categories ∈ {RDG, DRDG}, sizes ∈ {S,M}, and graph depths ∈ {2,3}.

	EVEN	Compl.	Incompl.	Incompl.+Noise
FOIL	1.0	0.4053	0.1919	0.0849
AMIE+	-	0.2021	0.2098	0.2075
Neural-LP	-	0.0633	0.0692	0.0649
NTP	1.0	0.0482	0.0617	0.0574

Table 4: Impact of (1) dataset category, H-score averaged on 40 datasets; (2) dataset size and rule graph depth, H-score averaged on 30 datasets. Datasets as in Section 5.1.

	CHAIN	RDG	DRDG
FOIL	0.2024	0.0877	0.1633
AMIE+	0.3395	0.2275	0.1293
Neural-LP	0.1291	0.1050	0.0734
NTP	0.1239	0.0538	0.0368

	S-2	S-3	M-2	M-3
FOIL	0.2815	0.2074	0.0356	0.0934
AMIE+	0.1449	0.1319	0.4392	0.2124
Neural-LP	0.1155	0.0673	0.1281	0.0992
NTP	0.1512	0.0432	0.0652	0.0374

Our results also confirm the system descriptions w.r.t. the rules they support. For instance, AMIE+ does not consider reflexive rules and requires rules to be connected (i.e., every atom must share an argument with each of the other atoms of the rule) and closed (all variables appear at least twice). And, by chance, the Chain datasets more often satisfy these conditions than the other datasets (this is not true in general: our generator produces comprehensive datasets that do not necessarily satisfy this property). Though, rules that are not fully supported can be recognized partially. We saw this by analyzing the rules learned by Neural-LP, which only supports chain rules of form $p_0(X_1, X_{m+1}) \coloneq p_1(X_1, X_2), \ldots, p_m(X_m, X_{m+1})$. NTP also performs better on Chain datasets, but the discrepancy with the other types of datasets is not substantial. This can be explained by the fact that we provided all necessary templates (for details about the system requirements see [27]). We cannot draw significant conclusions for FOIL given its unstable behaviour regarding the dataset type.

In summary, our experiments show the importance of considering datasets with different kinds of rules sets and different measures of performance to be able to fully understand the weaknesses and strengths of an ILP system.

5.4 Scalability: Impact of Dataset Size

In this experiment, we analyzed the impact of the dataset size considering four different size-depth combinations: the results for S-2, S-3, M-2, and M-3 datasets are reported in Table 4. We can observe that FOIL is not scalable, since there is a 20% performance gap from the S-dataset to the M-dataset. Although it does not

seem to be influenced by the rules dependency tree depth, showing support to nested rules. AMIE+ seemingly shows constant performance and thus scalability. We can observe that there is a noticeable drop in performance if we increase the depth of the rule dependency graphs. Neural-LP and NTP are robust to noise and incomplete data but NTP is not scalable yielding good accuracy only on the very small and simple instances (S-2), while Neural-LP seems to be more scalable (we cannot see a decrease of performance, augmenting the size of the dataset) but does not support nested rules.

6 Conclusions and Future Work

In this paper, we have presented RuDaS, a system for generating datasets for rule learning and for evaluating rule learning systems. RuDaS specifically fits neural approaches that need large amounts of data to learn from. Our experiments have shown that it is important to have diverse datasets of different sizes, with several rule types, and different amounts and types of noise; and different performance metrics to fully understand the systems' strengths and, more important, weaknesses. There are various future directions such as probabilistic logics and predicate types.

References

1. Alexe, B., Tan, W.C., Velegrakis, Y.: Stbenchmark: Towards a benchmark for mapping systems. Proc. VLDB Endow. **1**(1) (Aug 2008)
2. Arocena, P.C., Glavic, B., Ciucanu, R., Miller, R.J.: The ibench integration metadata generator. Proc. VLDB Endow. **9**(3) (Nov 2015)
3. Benedikt, M., Konstantinidis, G., Mecca, G., Motik, B., Papotti, P., Santoro, D., Tsamoura, E.: Benchmarking the chase. In: Proc. of PODS. ACM (2017)
4. Campero, A., Pareja, A., Klinger, T., Tenenbaum, J., Riedel, S.: Logical rule induction and theory learning using neural theorem proving. CoRR **abs/1809.02193** (2018)
5. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). In: IEEE Trans. on Knowl. and Data Eng. (1989)
6. Dong, H., Mao, J., Lin, T., Wang, C., Li, L., Zhou, D.: Neural logic machines. In: Proc. of ICLR (2019)
7. Dong, X.L., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proc. of KDD (2014)
8. Estruch, V., Ferri, C., Hernandez-Orallo, J., Ramrez-Quintana, M.: Distance based generalisation. In: Proc. of ILP. Springer-Verlag (2005)
9. Estruch, V., Ferri, C., Hernandez-Orallo, J., Ramrez-Quintana, M.: An integrated distance for atoms. In: Proc. of FLOPS (04 2010)
10. Evans, R., Grefenstette, E.: Learning explanatory rules from noisy data. J. Artif. Intell. Res. **61**, 1–64 (2018)
11. Fürnkranz, J., Gamberger, D., Lavrac, N.: Foundations of Rule Learning. Cognitive Technologies, Springer (2012)

12. 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), code available at https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie/
13. Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G.: Rule learning from knowledge graphs guided by embedding models. In: Proc. of ISWC, Part I. pp. 72–90 (2018)
14. ILP: ILP Applications and Datasets. https://www.doc.ic.ac.uk/~shm/applications.html (year na), accessed: 2020-09-03
15. de Jong, M., Sha, F.: Neural theorem provers do not learn rules without exploration. ArXiv **abs/1906.06805** (2019)
16. Krishnan, A.: Making search easier (2018), https://blog.aboutamazon.com/innovation/making-search-easier, accessed 2020-09-03
17. Minervini, P., Bosnjak, M., Rocktäschel, T., Riedel, S.: Towards neural theorem proving at scale. Proc. of NAMPI (2018)
18. Minervini, P., Bonjak, M., Rocktschel, T., Riedel, S., Grefenstette, E.: Differentiable reasoning on large knowledge bases and natural language. In: Proc. of AAAI (2020)
19. Muggleton, S.: Inverse entailment and progol. New Generation Comput. **13**(3&4), 245–286 (1995)
20. Nienhuys-Cheng, S., de Wolf, R.: Foundations of Inductive Logic Programming., vol. 1228. Springer (1997)
21. Nienhuys-Cheng, S.H.: Distance between herbrand interpretations: A measure for approximations to a target concept. In: Inductive Logic Programming. vol. 1297. Springer (1997)
22. Omran, P.G., Wang, K., Wang, Z.: Scalable rule learning via learning representation. In: Proc. of IJCAI. pp. 2149–2155 (2018)
23. Preda, M.: Metrics for sets of atoms and logic programs. Annals of the University of Craiova **33**, 67–78 (01 2006)
24. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning **5**, 239–266 (1990), code available at http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/learning/systems/foil/foil6/0.html
25. Raedt, L.D.: Logical and relational learning. Cognitive Technologies, Springer (2008)
26. Ren, H., Hu, W., Leskovec, J.: Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In: Proc. of ICLR (2020)
27. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: Proc. of NeurIPS. pp. 3791–3803 (2017), code available at https://github.com/uclmr/ntp
28. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall Press (2002)
29. Seda, A.K., Lane, M.: On continuous models of computation: Towards computing the distance between (logic) programs. In: Proc. of IWFM (2003)
30. Sinha, K., Sodhani, S., Pineau, J., Hamilton, W.L.: Evaluating logical generalization in graph neural networks. ArXiv **abs/2003.06560** (2020)
31. Sinha, K., Sodhani, S., Pineau, J., Hamilton, W.L.: Evaluating logical generalization in graph neural networks (2020)
32. Stepanova, D., Gad-Elrab, M.H., Ho, V.T.: Rule induction and reasoning over knowledge graphs. In: Reasoning Web. International Summer School, Tutorial Lectures. pp. 142–172 (2018)
33. Vaclav Zeman, T.K., Svtek, V.: Rdfrules: Making rdf rule mining easier and even more efficient. semantic-web-journal (2019)

34. Wang, Z., Li, J.: Rdf2rules: Learning rules from RDF knowledge bases by mining frequent predicate cycles. CoRR **abs/1512.07734** (2015)
35. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: Proc. of NeurIPS. pp. 2316–2325 (2017), code available at https://github.com/fanyangxyz/Neural-LP
36. Yang, Y., Song, L.: Learn to explain efficiently via neural logic inductive learning. In: Proc. of ICLR (2020)

A Additional Preliminaries

Our measures are based on the concepts of Herbrand base and structure. Generally formal logic distinguishes between syntax (symbols) and its interpretation. Herbrand structures, which are used for interpretation, are however defined solely by the syntactical properties of the vocabulary. The idea is to directly take the symbols of terms as their interpretations which, for example, has proven useful in the analysis of logic programming.

The focus is on ground terms (e.g., atoms and rules), which are terms that do not contain variables.

The Herbrand base of a FOL vocabulary (of constants, predicates, etc.) is the set of all ground terms that can be formulated in the vocabulary. If the vocabulary does not contain constants, then the language is extended by adding an arbitrary new constant.

A Herbrand model interprets terms over a Herbrand base, hence it can be seen as a set of ground atoms. Let T be the set of all variable-free terms over a vocabulary V. A structure S over V and with base U is said to be a Herbrand structure iff $U = T_0$ and $c^S = c$ for every constant $c \in V$.[15]

B Dataset Generation

In this section, we describe the generation process of the rules and facts in detail, assuming the generator parameters (also *configuration*) listed in Section 3 to be set.

Preprocessing. As already mentioned, most parameters are determined randomly in a preprocessing step if they are not fixed in the configuration, such as the symbols that will be used, the numbers of DGs to be generated, and their depths. However, all random selections are within the bounds given in the configuration under consideration; for instance, we ensure that the symbols chosen suffice to generate rule graphs and fact sets of selected size and that at least one graph is of the given maximal depth.

Rule generation. According to the rule set category specified and graph depths determined, rules (nodes in the graphs) of form (2) are generated top down breadth first, for each of the rule graphs to be constructed. The generation is largely at random, that is, w.r.t. the number of child nodes of a node and which body atom they relate to; the number of atoms in a rule; and the predicates within the latter, including the choice of the *target predicate* (i.e., the predicate in the head of the root) in the very first step. RuDaS also offers the option that all graphs have the same target predicate. To allow for more derivations, we currently only consider variables as terms in head atoms; the choice of the remaining terms is based on probabilities as described in the following. Given the atoms to be considered (in terms of their number and predicates) and an arbitrary choice of head variables, we first determine a position for each of the

[15] Recall that we disregard function symbols.

latter in the former. Then we populate the other positions one after the other: a head variable is chosen with probability $p_h = \frac{1}{5}$; for one of the variables introduced so far, we have probability $p_v = (1 - p_h) * \frac{3}{4}$; for a constant, $p_c = (1-p_h)*(1-p_v)*\frac{1}{10}$; and, for a fresh variable, $p_f = (1-p_h)*(1-p_v)*(1-p_c)$. While this conditional scheme might seem rather complex, we found that it works best in terms of the variety it yields; also, these probabilities can be changed easily.

Fact generation. The fact generation is done in three phases: we first construct a set $\mathbb{D}$ of relevant facts in a closed-world setting, consisting of support facts $\mathbb{S}$ and their consequences $\mathbb{C}$, and then adapt it according to n_{OW} and $n_{\text{Noise*}}$.

As it is the (natural) idea, we generate facts by instantiating the rule graphs multiple times, based on the assumption that rule learning systems need positive examples for a rule to learn that rule, and stop the generation when the requested number of facts has been generated. We actually stop later because we need to account for the fact that we subsequently will delete some of them according to n_{OW}. More specifically, we continuously iterate over all rule graphs, for each, select an arbitrary but fresh variable assignment σ, and then iterate over the graph nodes as described in the following, in a bottom-up way. First, we consider each leaf n and corresponding rule of form (2) and generate support facts $\sigma(\alpha_1), \ldots, \sigma(\alpha_m)$. Then, we infer the consequences based on the rules and all facts generated so far. For every node n on the next level and corresponding rule of form (2), we only generate those of the facts $\sigma(\alpha_1), \ldots, \sigma(\alpha_m)$ as support facts which are not among the consequences inferred previously. We then again apply inference, possibly obtaining new consequences, and continue iterating over all nodes in the graph in this way. We further diversify the process based on two integer parameters, n_{DG} and n_{Skip}: in every n_{DG}-th iteration the graph is instantiated exactly in the way described; in the other iterations, we skip the instantiation of a node with probability $1/n_{\text{Skip}}$ and, in the case of DR-DGs, only instantiate a single branch below disjunctive nodes. We implemented this diversification to have more variability in the supports facts, avoiding to have only complete paths from the leaves to the root.

In the open-world setting, we subsequently construct a set $\mathbb{D}_{\text{OW}}$ by randomly deleting consequences from $\mathbb{D}$ according to the open-world degree given: assuming $\mathbb{T} \subseteq \mathbb{C}$ to be the set of *target facts* (i.e., consequences containing the target predicate), we remove $n_{\text{OW}}\%$ from $\mathbb{C} \setminus \mathbb{T}$, and similarly $n_{\text{OW}}\%$ from $\mathbb{T}$. In this way, we ensure that the open-world degree is reflected in the target facts. Though, there is the option to have it more arbitrary by removing $n_{\text{OW}}\%$ from $\mathbb{C}$ instead of splitting the deletion into two parts.

The noise generation is split similarly. Specifically, we construct a set $\mathbb{D}_{\text{OW+Noise}}$ based on $\mathbb{D}_{\text{OW}}$ by arbitrarily removing $n_{\text{Noise-}}\%$ from $\mathbb{S}$, and by adding arbitrary fresh facts that are neither in $\mathbb{C}$ (i.e., we do not add facts we have removed in the previous step) nor contain the target predicate such that $\mathbb{D}_{\text{OW+Noise}} \setminus \mathbb{T}$ contains $n_{\text{Noise+}}\%$ of noise. In addition, we add arbitrary fresh facts on the target predi-

Table 5: Overview of our generated datasets, altogether 196; column # is the count of datasets described in the corresponding row. All other numbers are averages. For Chain, we have $n_{\text{OW}} = 0.3$, $n_{\text{Noise-}} = 0.2$, and $n_{\text{Noise+}} = 0.1$. For RDG and DRDG: $n_{\text{OW}} \in \{0.2, 0.3, 0.4\}$, $n_{\text{Noise-}} \in \{0.15, 0.2, 0.3\}$, and $n_{\text{Noise+}} \in \{0.1, 0.2, 0.3\}$. Note that the size bounds of our fact sets are not strict, some sizes are slightly larger than expected (e.g., 1065 for size M) because our initial generation needs to take into account that some facts, e.g., consequences, may be removed thereafter.

#	Rule type	Size	Depth	#Rules			#Facts			#Pred			#Const		
				min	avg	max	min	avg	max	min	avg	max	min	avg	max
10	CHAIN	S	2	2	2	2	51	74	95	5	7	9	31	47	71
10	CHAIN	S	3	3	3	3	49	70	97	7	8	9	31	43	64
10	CHAIN	M	2	2	2	2	168	447	908	9	10	11	97	259	460
10	CHAIN	M	3	3	3	3	120	508	958	8	10	11	52	230	374
22	RDG	S	2	3	3	3	49	84	122	6	9	11	28	50	84
12	RDG	S	3	4	5	6	56	104	172	8	10	11	41	55	75
22	RDG	M	2	3	3	3	200	646	1065	6	11	11	71	370	648
22	RDG	M	3	4	5	7	280	613	1107	10	11	11	149	297	612
22	DRDG	S	2	3	4	5	60	100	181	6	9	11	29	55	82
12	DRDG	S	3	4	7	11	58	144	573	8	10	11	34	58	89
22	DRDG	M	2	3	4	5	149	564	1027	10	11	11	88	327	621
22	DRDG	M	3	4	7	12	111	540	1126	10	11	11	70	284	680

cate that are not in $\mathbb{T}$ already such that subset of $\mathbb{D}_{\text{OW+Noise}}$ on that predicate finally contains $n_{\text{Noise+}}\%$ of noise.

Output. The dataset generation produces: the rules; a *training* set ($\mathbb{D}_{\text{OW+Noise}}$), which is of the requested size, and fulfills n_{OW}, $n_{\text{Noise+}}$, and $n_{\text{Noise-}}$; and custom fact sets $\mathbb{S}'$ and $\mathbb{C}'$ for our evaluation tools generated in the same way as $\mathbb{S}$ and $\mathbb{C}$. For further experiments, RuDaS also outputs $\mathbb{D}$, $\mathbb{D}_{\text{Noise}}$ (an adaptation of $\mathbb{D}$ which contains noise but all of $\mathbb{C}$), $\mathbb{D}_{\text{OW}}$, $\mathbb{S}$, and $\mathbb{C}$ (see also the end of Section C).

C Statistics of RuDaS.v0

Table 5 provides statistics regarding RuDaS.v0: the generated set of datasets available to the community in our repository.

D Approaches to Rule Learning

Classical ILP systems such as FOIL [24] and Progol [19] usually apply exhaustive algorithms to mine rules for the given data and either require false facts as counter-examples or assume a closed world (for an overview of classical ILP systems see Table 2 in [32]). The *closed-world assumption* (CWA) (vs. *open*

world assumption or OWA) states that all facts that are not explicitly given as true are assumed to be false.

Today, however, knowledge graphs with their often incomplete, noisy, heterogeneous, and, especially, large amounts of data raise new problems and require new solutions. For instance, real data most often only partially satisfies the CWA and does not contain counter-examples. Moreover, in an open world, absent facts cannot be considered as counter-examples either, since they are not regarded as false. Therefore, successor systems, with AMIE+ [12] and RDF2Rules [34] as the most prominent representatives, assume the data to be only partially complete and focus on rule learning in the sense of mining patterns that occur frequently in the data. Furthermore, they implement advanced optimization approaches that make them applicable in wider scenarios. In this way, they address already many of the issues that arise with today's knowledge graphs, still maintaining their processing exhaustive.

Recently, neural rule learning approaches have been proposed: [35, 27, 10, 17, 22, 4]. These methodologies seem a promising alternative considering that deep learning copes with vast amounts of noisy and heterogeneous data. The proposed solutions consider vector or matrix embeddings of symbols, facts and/or rules, and model inference using differentiable operations such as vector addition and matrix composition. However, they are still premature: they only learn certain kinds of rules or lack scalability (e.g., searching the entire rule space) and hence cannot compete with established rule mining systems such as AMIE+ yet, as shown in [22], for example.

E System Configurations

All the systems have the same computational restrictions (i.e. CPU, memory, time limit, etc.). The reader can find all the details (scripts etc.) in the RuDaS GitHub repository.

FOIL

- **Paper:** Learning logical definitions from relations. Machine Learning, 5:239-266, 1990. https://www.semanticscholar.org/paper/Learning-logical-definitions-from-relations-Quinlan/554f3b32b956035fbfabba730c6f0300d6955dce
- **Source Code:** http://www.rulequest.com/Personal/ or http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/learning/systems/foil/foil6/0.html , Version: 6
- **Running configuration:**

```
\$SYSDIR/\$SYSTEM/FOIL/./foil6 -v0 -n -m 200000
   < \$PREPROCESSINGFOLDER\$FILENAME.d
```

  `-m 200000`: used when the max tuples are exceeded
- **Parameter for accepting the rules:** NA – all the rules are accepted

Amie+

- **Paper:** Fast Rule Mining in Ontological Knowledge Bases with AMIE+. Luis Galrraga, Christina Teflioudi, Fabian Suchanek, Katja Hose. VLDB Journal 2015. https://suchanek.name/work/publications/vldbj2015.pdf
- **Source Code:** https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie/ , Version of 2015-08-26
- **Running configuration:**

```
java -jar \$SYSDIR/\$SYSTEM/amie\_plus.jar
    -mins 3 -minis 3 -minpca 0.25
    -oute \$DATA/\$SYSTEM/\$NAME/train.txt
    > \$DIR/../output/binary/\$SYSTEM/\$NAME/results.txt
```

- **Parameter for accepting the rules:** learned using grid-search = 0.7 – all the rules with PCA Confidence > 0.7 are accepted

Neural-LP

- **Paper:** Differentiable Learning of Logical Rules for Knowledge Base Reasoning.Fan Yang, Zhilin Yang, William W. Cohen. NIPS 2017. https://arxiv.org/abs/1702.08367
- **Source Code:** https://github.com/fanyangxyz/Neural-LP
- **Running configuration:**

```
python \$SYSDIR/\$SYSTEM/src/main.py
    --datadir=\$DATA/\$SYSTEM/\$NAME
    --exp\_dir=$DIR/../output/binary/$SYSTEM
    --exp\_name=\$NAME
    > \$DIR/../output/binary/\$SYSTEM/\$NAME/log.txt
```

- **Parameter for accepting the rules:** learned using grid-search = 0.6 – all the rules with ri-normalized prob > 0.6 are accepted

Neural-theorem prover (ntp)

- **Paper:** End-to-end Differentiable Proving. Tim Rocktaeschel and Sebastian Riedel. NIPS 2017. http://papers.nips.cc/paper/6969-end-to-end-differentiable-proving
- **Source Code:** https://github.com/uclmr/ntp
- **Running configuration:**

```
python \$SYSDIR/\$SYSTEM/ntp/experiments/learn.py
    \$DATA/\$SYSTEM/\$NAME/run.conf
    > \$DIR/../output/binary/\$SYSTEM/\$NAME/log.txt
```

- **Parameter for accepting the rules:** learned using grid-search = 0.0 – all the rules are accepted

```
{
    "data": {
        "kb": "$DATAPATH/$TRAIN.nl",
        "templates": "$DATAPATH/rules.nlt"
    },
    "meta": {
        "parent": "$SYSTEMSPATH/conf/default.conf",
        "test_graph_creation": False,
        "experiment_prefix": "$NAME",
        "test_set": "$TEST",
        "result_file": "$OUTPUTPATH/results.tsv",
        "debug": False
    },
    "training": {
        "num_epochs": 100,
        "report_interval": 10,
        "pos_per_batch": 10,
        "neg_per_pos": 1,
        "optimizer": "Adam",
        "learning_rate": 0.001,
        "sampling_scheme": "all",
        "init": None, # xavier initialization
        "clip": (-1.0, 1.0)
    },
    "model": {
        "input_size": 100,
        "k_max": 10,
        "name": "???",
        "neural_link_predictor": "ComplEx",
        "l2": 0.01, # 0.01 # 0.0001
        "keep_prob": 0.7
    }
}
```