RuleHub: a Public Corpus of Rules for Knowledge Graphs

NASER AHMADI, EURECOM, France
THI-THUY-DUYEN TRUONG, EURECOM, France
LE-HONG-MAI DAO, EURECOM, France
STEFANO ORTONA, Meltwater, UK
PAOLO PAPOTTI, EURECOM, France

Entity-centric knowledge graphs (KGs) are now popular to collect facts about entities. KGs have rich schemas, with a large number of different types and predicates to describe the entities and their relationships. On these rich schemas, logical rules are used to represent dependencies between the data elements. While rules are useful in query answering, data curation, and other tasks, they usually do not come with the KGs. Such rules have to be manually defined or discovered with the help of rule mining methods. We believe this rule-collection task should be done collectively to better capitalize our understanding of the data and to avoid redundant work conducted on the same KGs. For this reason, we introduce *RuleHub*, our extensible corpus of rules for public KGs. RuleHub provides functionalities for the archival and the retrieval of rules to all users, with an extensible architecture that does not constrain the KG or the type of rules supported. We are populating the corpus with thousands of rules from the most popular KGs and report on our experiments on automatically characterizing the quality of a rule with statistical measures.

CCS Concepts: • **Information systems → Data mining**; **Data cleaning**.

Additional Key Words and Phrases: rule mining, knowledge graphs, graph dependencies

ACM Reference Format:
Naser Ahmadi, Thi-Thuy-Duyen Truong, Le-Hong-Mai Dao, Stefano Ortona, and Paolo Papotti. 2020. RuleHub: a Public Corpus of Rules for Knowledge Graphs. 1, 1 (July 2020), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Knowledge graphs (KGs) represent data with large collections of interconnected entities. Usually, a rich set of types (classes) are available to describe the entities (e.g., entity *Paris* is a *city*, *France* is a *country*), while predicates describe their relationships (a city *isCapital* of a country) and their properties (France has a *population:62M*). Over the last 15 years, lots of large academic [7, 10, 35, 39] and institutional [8, 14, 18] KGs have been presented. Entities across several KGs have been aligned to create the public web of *linked open data*, contributing to the creation of a larger graph [6]. Public and institutional KGs fuel several applications, including semantic search, personal assistants, and question answering in general [18, 27].

One of the great benefits of a structured dataset is that it is possible to define *dependencies* over it. KG dependencies (or *logical rules*) are used for identifying errors [28], adding new facts [20],

Authors' addresses: Naser Ahmadi, EURECOM, France, Naser.Ahmadi@eurecom.fr; Thi-Thuy-Duyen Truong, EURECOM, France, Thi-Thuy-Duyen.Truong@eurecom.fr; Le-Hong-Mai Dao, EURECOM, France, Le-Hong-Mai.Dao@eurecom.fr; Stefano Ortona, Meltwater, UK, stefano.ortona@gmail.com; Paolo Papotti, EURECOM, France, papotti@eurecom.fr.

XXXX-XXXX/2020/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

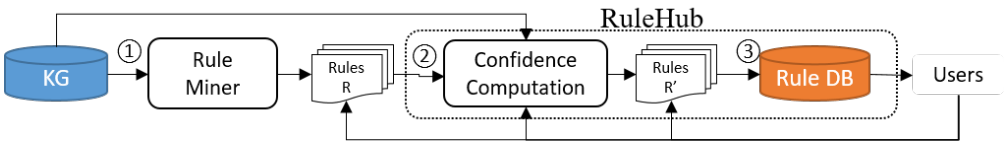


Fig. 1. Architecture of RuleHub.

executing queries faster [31], and reasoning for many tasks, such as explaining decisions in fact-checking [5]. Unfortunately, KGs do not come with a set of logical rules. In fact, manually crafting such rules is an expensive, human-intensive task. To support the definition of rules, several methods have been proposed. The literature for rule mining goes back to classic work on inductive logic programming and several methods have been recently proposed to mine large KGs [17, 20, 28, 32, 37]. This stream of works enables the mining of rules for any KG, thus limiting the user effort to the selection and the refinement of such rules. Users select valid rules from the many (possible thousands) discovered by a system and manually refine the ones that need changes in their conditions to be more general or more precise. Once a good set of semantically valid rules has been identified, they are usually annotated with a measure of their quality, such as a *confidence* of the rule applicability. This is necessary, as there are very few rules that are true for each and every case. As an example, consider a rule stating that "a country has always one capital". This is true for most countries, but there are 15 countries that have two or more capitals. Therefore the rule would have a very high confidence, but we cannot state that it is certain.

Collecting a set of high quality and well described rules is therefore an essential exercise conducted by users of KGs to improve the performance of their applications. While mining tools help in this task, there is still a lot of manual work in selecting good rules, refining them, and annotating them with their metadata. Given that several KGs are public and are used by thousands of researchers and practitioners around the world, a lot of human work on rule discovery is redundant and is not taking any benefit from this critical mass of users. We argue that rule discovery and managing should be a collective task, where we can capitalize our understanding of the data and avoid redundant work in collecting, refining, and annotating rules.

To support a collective rule discovery effort, we introduce *RuleHub*, a system that exposes in a website an extensible corpus of rules for public KGs (http://rudik.eurecom.fr). RuleHub is designed with rule mining tools as the principal source of rules in mind, as depicted in Figure 1. Users can query or browse the repository of rules, based on the KG and the predicate of interest. Moreover, they can manually specify and add new rules or update existing ones by providing more metadata, such as the confidence of a rule. We believe confidence is crucial for rules, as very few rules are completely correct or wrong in general. For this reason, we also provide a module that computes the confidence for a given rule over a KG. Rules for any KG and of any kind can be handled by the system. In this work, we report our experience in building RuleHub and populating it with rules discovered by existing rule mining systems. We discuss three main contributions:

(1) We introduce a new confidence computation method for negative rules, i.e., rules that identify contradictions in the data (Section 2).
(2) We present the first open corpus of (automatically generated) rules for public KGs and detail its data model and its components (Section 3).
(3) We report the lessons learned in creating such corpus and the experimental results showing how our model for computing rule confidence has remarkable correlation with manually annotated confidence values (Section 4).

We believe RuleHub can be an enabler for a much needed collaborative work in defining metadata for public KGs, as witnessed in the large amount of recent efforts in the related literature (Section 5). Finally, we conclude the paper with our future directions of research (Section 6).

2 MINING RULES AND CONFIDENCE

2.1 Knowledge Graphs and Rules

A *Knowledge Graph* (KG) is a structured representation of information storing real-world entities and lexical values as nodes, and relationships between them as edges. Most KGs organize information in the form of (RDF) triples. Each triple contains a *predicate* (edge in the graph) expressing a binary relation between a *subject* and an *object* (nodes in the graph). We focus on Horn rules of the form:

$$\vec{B} \rightarrow r(x, y) \tag{1}$$

where $r(x, y)$ is a single atom (head of the rule) and $\vec{B}$ (body of the rule) is a conjunction of atoms $B_1(z_1, z_2) \wedge B_2(z_3, z_4) \wedge \cdots \wedge B_n(z_{n-1}, z_n)$. An atom is a predicate connecting two variables, two entities, an entity and a variable, or a variable and a lexical value. We distinguish two classes of rules. *Positive rules* define relationships between KG elements that identify missing triples, e.g., "if two persons have a child in common, they are in the spouse relation". *Negative rules* have a negated atom in the head identify false facts from the ones in the KG, e.g., "parents cannot marry their children". As an example of using negative rules for error detection, this negative rule $child(x, y) \rightarrow \neg spouse(x, y)$ can be rewritten as $child(x, y) \wedge spouse(x, y) \rightarrow \perp$ and the body is run as a query on the KG.

2.2 Rule Mining

Given a KG, a set of positive example facts, and a set of negative example facts, a rule mining algorithm aims at identifying a set of rules that, when executed on the KG, have all the positive examples facts in their output and none of the negative facts. In other words, the algorithms are after rules that are general, as they cover all true facts, and do not make mistakes, as they do not produce false facts. As these requirements are quite strict in the presence of noisy and incomplete KGs, they are relaxed in some mining algorithms with a notion of weight (or score) defined on the number of positive and negative examples in the output of the set of mined rules.

The goal of RuleHub is not to propose a new mining algorithm. Our focus is to manage a large number of rules, by computing an estimate of their quality and enabling their review, storing, and querying. In Building the corpus, we collected rules from many mining algorithms, but most of the rules come from AMIE [20] and RuDiK [28, 29]. We give more details about their algorithms in Appendix A and report other related proposals in Section 5.

2.3 Rule Confidence

We now discuss the computation of the confidence for positive and negative rules. We start with the notion of confidence of a positive rule from the literature [20] and extend it to negative rules.

The *support* of a rule is defined as the number of distinct pair of subjects and objects in the head of all instantiations of the rule that appear in the KG:

$$supp(\vec{B} \rightarrow r(x, y)) := \#(x, y) : \exists z_1, ..., z_n : \vec{B} \wedge r(x, y) \tag{2}$$

where $z_1, ..., z_n$ are the non target variables.

We remark that the support makes use of the non target variables, but count only the number of distinct pairs for the head values. Consider the rule "if two persons have a child in common, they are in the spouse relation" ($hasChild(x, v0) \wedge hasChild(y, v0) \rightarrow spouse(x, y)$) and the Obama family. Assume Barack Obama, Michelle Obama, and their two children are in the KG with the

correct Spouse and Child triples to represent their relationships. The support would count this family as one occurrence. While it uses the non target variables (referring to the child), the measure does not count this family twice, despite the rule can be instantiated twice (once for child). This design choice takes care of possible skew in the data, making sure that a rule does not get assigned a very high support when it applies for only one (distinct) pair of head entities.

The *counter-support* of a rule quantifies the number of false predictions over the existing KG. A challenge to compute this number is that KGs do not provide negative evidence, but we want to claim a mistake only if we have some evidence to support the case. To address this issue, we rely on Local Closed World Assumption. This assumption states that if we know one y (resp. x) for a given x (resp. y) and r, then we know all y (resp. x) for that x (resp. y) and r. This is widely used in practice and has proven to be an effective heuristic to overcome incompleteness of KGs [14, 20, 28]. Assume a rule concludes that $P(x_1, y_1)$ should exist. If this is the only triple involving P, x_1, and y_1, then we do not count it neither as supporting or not supporting. But if there is already a triple such as $P(x_1, y_2)$ in the KG, then we count this a counter support. This allows us to exploit counter-evidences less restrictively than the assumption 'all facts that are not in KG are false'.

$$counter\_supp(\vec{B} \rightarrow r(x,y)) := \#(x,y) : \exists z_1, ..., z_n, r(x,y') \vee r(x',y) : \vec{B} \wedge \neg r(x,y) \tag{3}$$

For example, a rule predicts that "Luke" and "Mary" are married, this triple is not in the KG, but "Luke" is already reported as married to someone else in the KG. To take into account both true and false predictions for a rule, we introduce *confidence scores* for positive and negative rules.

Positive Rules. Considering a positive rule $\vec{B} \rightarrow r(x,y)$ for relation $r(x,y)$. We define its confidence score as following:

$$conf(\vec{B} \rightarrow r(x,y)) := \frac{supp(\vec{B} \rightarrow r(x,y))}{supp(\vec{B} \rightarrow r(x,y)) + counter\_supp(\vec{B} \rightarrow r(x,y))} \tag{4}$$

This formula normalizes the support by number of pairs (x,y) satisfying the condition that there exists $r(x',y)$ or $r(x,y')$. As an example, consider the rule $hasDependant(a,b) \rightarrow hasChild(a,b)$ with the following triples $hasDependant(Laure, Mark)$, $hasDependant(Laure, Mary)$, $hasDependant(Laure, Anne)$, $hasChild(Laure, Mary)$, $hasChild(Laure, Anne)$, in other words, one person has three dependants and two of them are her children. Assume we also have two more triples for another person: $hasDependant(Gary, Rose)$, $hasChild(Gary, Mark)$. The confidence for this rule would be 0.5. In fact, support would be 2, coming from (Laure,Mary) and (Laure,Anne), and counter support would also be 2, coming from (Laure,Mark) and (Gary,Rose). The confidence value reflects the reality: being a dependant of someone does not imply that there is a child relationship, but it can happen. How often it happens in the given KG determines the support for the rule in that context, e.g., it may be different in Europe and USA because of legal or cultural reasons.

Negative Rules. Consider a negative rule $\vec{B'} \rightarrow \neg r(x,y)$ for relation $r(x,y)$. We define its confidence score as following:

$$conf(\vec{B'} \rightarrow \neg r(x,y)) := \frac{counter\_supp(\vec{B'} \rightarrow r(x,y))}{counter\_supp(\vec{B'} \rightarrow r(x,y)) + supp(\vec{B'} \rightarrow r(x,y))} \tag{5}$$

Intuitively, by definition, the support (resp. counter-support) of negative rule $\vec{B'} \rightarrow \neg r(x,y)$ is indeed the counter-support (resp. support) of corresponding positive rule $\vec{B'} \rightarrow r(x,y)$. As an example, consider the rule $spouse(a,b) \rightarrow \neg hasChild(a,b)$ and triples $spouse(Laure, Mark)$, $hasChild(Laure, Anne)$, $spouse(Gary, Rose)$, and $hasChild(Gary, Micheal)$. The confidence for this rule would be 1. In fact, counter support would be 2, coming from (Laure,Mark) and (Gary,Rose), and

support would be 0. If we add one case such as $spouse(Paul, Ron)$ without children, the confidence does not change. If we add $spouse(Mary, George)$, $hasChild(Mary, George)$, then the support would become 1 and the confidence would drop to 0.67.

Issues in Confidence Measures. In negative rules, support evidences are dominating, but counter-evidences are more important. For example, consider the negative rule:

$$r_1 : birthPlace(x, v0) \wedge country(v1, v0) \wedge child(y, v1) \rightarrow \neg spouse(x, y)$$

Due to the presence of the intermediate atom $country(v1, v0)$ and the common atom $birthPlace(x, v0)$, we can generate a large number of support examples for $spouse$. Meanwhile, the counter-support of this rule is always lower than the total number of facts $spouse$ in KG. If formula (5) is used to evaluate the rule, we obtain a very high confidence score (≈ 1.0), which is misleading because the number of supporting examples is way higher then negative ones.

With the same computation, a rule $child(x, y) \rightarrow \neg spouse(x, y)$ gets a confidence score of 0.96. From these scores, the two rules seem to be of comparable quality, but intuitively rule $child(x, y) \rightarrow \neg spouse(x, y)$ can cover a much larger number of facts and its confidence score should be higher. Here, considering the body of the rule, $spouse(x, y)$ is false. Therefore, its score should be much higher than rule r_1.

The explanation is that the support $\#(x, y)$ for rule r_1 is very large, but because of the Cartesian product, $\#x$ can dominate $\#y$ or vice versa. This undesired property overestimates the support and thereby underestimates the counter-support of a rule. To avoid this issues we define a new support for negative rules by using the minimum number of facts satisfying the head predicate, denoted as min_supp:

$$min\_supp(\vec{B} \rightarrow \neg r(x, y)) := min(\#x, \#y) : \exists z_1, ..., z_n, r(x, y') \vee r(x', y) : \vec{B} \wedge \neg r(x, y) \quad (6)$$

Which returns the smaller across the numbers of occurrences for variables x and y in the KG that already satisfy the predicate. Now, (5) becomes:

$$conf(\vec{B'} \rightarrow \neg r(x, y)) := \frac{min\_supp(\vec{B'} \rightarrow \neg r(x, y))}{min\_supp(\vec{B'} \rightarrow \neg r(x, y)) + counter\_supp(\vec{B'} \rightarrow \neg r(x, y))} \quad (7)$$

Here we re-write the support of the corresponding positive rule (second term of the denominator) as $counter\_supp$ for better clarification, but essentially $counter\_supp(\vec{B'} \rightarrow \neg r(x, y)) = supp(\vec{B'} \rightarrow r(x, y))$.

As an example, consider three more rules: $r_2 : parent(x, y) \rightarrow \neg spouse(x, y)$; $r_3 : relative(x, y) \rightarrow \neg spouse(x, y)$; $r_4 : occupation(x, v0) \wedge occupation(y, v0) \rightarrow \neg spouse(x, y)$.

Rule	min_supp	$counter\_supp$	Conf. Score Equation (7)	Conf. Score Equation (8)
r_2	8174	43	0.995	0.881
r_3	1859	123	0.938	0.375
r_4	11792	1757	0.870	0.211

Table 1. Support, Counter_Support, Confidence Scores for r_2, r_3, r_4

Consider in Table. 1 the computed values for min_supp, $counter\_support$ and confidence score computed from Equation (7) for r_2, r_3, r_4. Although there are significant differences in term of $min\_support$ and $counter\ support$ among the rules, they still get very high confidence score. In common sense, r_2 is better than r_3, and r_4 is not a useful rule. But their $min\_support$ values are still large enough to hide the differences in the $counter\ support$ and lead to an overestimate of the confidence. In other words, the ratio of $counter\_supp : min\_supp$ of the three rules are 0.005, 0.066, 0.150, respectively, but is not reflected in the values for (7). We argue that counter-evidence is more

important in measuring the quality of negative rules, so it is critical to make it contribute more to the confidence computation. A reasonable way to achieve this is to make the counter-evidence comparable in values to the support evidences. We achieve this by multiplying the *counter_supp* by a factor κ, as follows:

$$conf(\vec{B'} \rightarrow \neg r(x,y)) := \frac{min\_supp(\vec{B'} \rightarrow \neg r(x,y))}{min\_supp(\vec{B'} \rightarrow \neg r(x,y)) + \kappa * counter\_supp(\vec{B'} \rightarrow \neg r(x,y))} \quad (8)$$

How to set κ depends on how rare are errors. This is modeled by the actual percentage of errors in the KG at hand. We assume that we are given (or can estimate) the ϵ error rate of the KG, i.e., the ratio of erroneous triples over all triples. We set κ to $\frac{1}{\epsilon}$. KGs with very low error rate will get higher values for κ to preserve the $counter\_supp : min\_supp$ ratio in the original confidence computation formula. Yago reports an estimated accuracy of 95% [35] and other papers report higher estimated values [28]; we experimentally found that an estimate accuracy of 96% works better for our confidence computation. Given accuracy of 96%, an ϵ equals to 0.04 implies $\kappa = 25$.

Using Equation (8), new confidence scores of r_2, r_3, r_4 are re-calculated in the last column of Tab. 1. The new confidence values match our intuitions about the rules.

3 A CORPUS OF RULES

There are different algorithms to discover rules over KGs, as a consequence there are many rules that can be generated from different experiments over time. With the aim to store and share discovered rules as well as to expose them with metadata, such as the confidence discussed in Section 2, we have built a website with an initial corpus of more than 7000 mined rules (http://rudik.eurecom.fr). Besides basic information about the rule itself, we expose different kinds of confidence score: *computed confidence*, which is calculated automatically, while *human evaluation* and *human confidence* are evaluated manually. Our web portal allows users to easily search for rules, add new rules, and export them in different formats. Our long term vision is to have users contributing by opening the database to users for editing as in Wikidata. For now the system has an admin validation panel, where rules submitted by users get reviewed before being added to the corpus. In this section, we describe the information in the corpus and how to use it. Screenshots and details about the portal are reported in Appendix C.

3.1 Human measures

We start presenting our two measures of quality for the rules based on human judgment.

3.1.1 Quality evaluation. The first one is the *quality evaluation*, a subjective assessment about the rule semantic correctness, i.e., a score of their logical meaning expressed by a human. In this measure, a user expresses a subjective assessment of the quality of a rule by reading the rule in its logical form. For example, the rule $spouse(a,b) \rightarrow spouse(b,a)$ can be judged as clearly correct. We define five levels of accurateness as following:

Level 1: Good rules, the precision is more than 80 percent.
Level 2: Acceptable rules, the precision is around 60 - 80 percent.
Level 3: Neutral rules, the precision is around 40 - 60 percent.
Level 4: Rules make sense only in certain contexts, the precision is around 20 - 40 percent.
Level 5: Illogical rules, not recommended for use.

Because each individual has her own evaluation for this score, we allow different users to express their assessment in our system.

3.1.2 Human confidence. The second score is the *human confidence*. This score is calculated manually based on the examination of the triples resulting from the application of a given rule. For each rule, we apply it on the KG and randomly pick 20 instances from such output. Every instance in this sample is manually validated according to external resources, such as the Web, and human assessment.

Evaluation process:

(1) Given a rule, we apply it over the KG and randomly select from the results 20 instances as a sample for human confidence computation.
(2) Three different annotators manually check instances independently, and conflicts are resolved with a majority voting strategy. For each positive rule, an instance is labeled as 1 if it is true, 0 otherwise. For each negative rule, an instance is labeled as 1 if it is erroneous, 0 otherwise.
(3) From the result of the labeling process, the human confidence of a rule is then computed as the ratio of the number of labels 1 out of all items in the sample.

Even if computed on a sample, the second score is more objective than the human assessment and we use it as our reference to evaluate the quality of the confidence computed with the measures discussed in Section 2.

3.2 Rule information

Each rule is stored as a JSON (JavaScript Object Notation [24]) file in a MongoDB instance with metadata about its provenance, support, and manual evaluation results.

- *id*: the rule internal identifier.
- *knowledge_graph*: the knowledge graph in which the rule is valid.
- *rule_type*: true or false corresponds to positive or negative rule.
- *predicate*: the target predicate.
- *premise*: the rule body.
- *hashcode*: it is computed from the predicate and rule premise, it is used to check uniqueness of rules in the system.
- *human_confidence*: indicates the human confidence.
- *computed_confidence*: indicates the support score.
- *source*: indicates the rule's origin. Rule can be added by users or obtained from a discovery system.
- *configuration*: configuration of the mining system that found the rule.
- *quality_evaluation*: subjective human evaluation.

The last three fields can have multiple occurrences for the same rule.

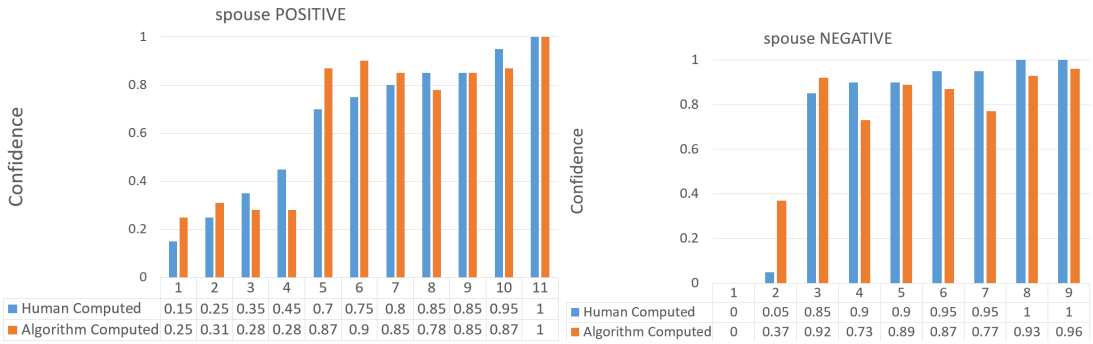
3.3 Rule forms

Besides exploring rules in our web portal directly, we also provide rules in different formats which can be easily consumed by other systems.

- *JSON* is a popular light-weight format designed for easy parsing.
- *SPARQL* is an RDF query language. SPARQL queries return rule output from the KG [30].

4 EXPERIMENTS

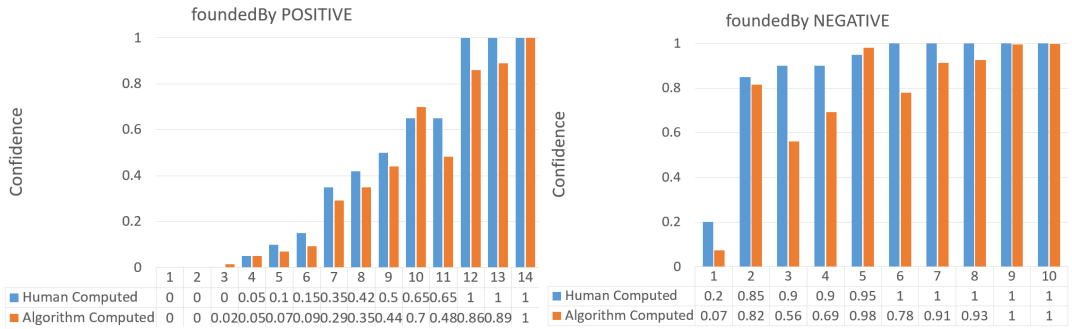
In this section, we evaluate our confidence measures and report on the lessons learned in annotating rules. We group our evaluations into four parts: *(i)* showing the accuracy of our confidence measures by comparing them with the human computed confidences (see Section 3.1). *(ii)* discussing the effect of the κ parameter on the results. *(iii)* evaluating the performance of the proposed measure



	1	2	3	4	5	6	7	8	9	10	11
Human Computed	0.15	0.25	0.35	0.45	0.7	0.75	0.8	0.85	0.85	0.95	1
Algorithm Computed	0.25	0.31	0.28	0.28	0.87	0.9	0.85	0.78	0.85	0.87	1

	1	2	3	4	5	6	7	8	9
Human Computed	0	0.05	0.85	0.9	0.9	0.95	0.95	1	1
Algorithm Computed	0	0.37	0.92	0.73	0.89	0.87	0.77	0.93	0.96

Fig. 2. Confidence results for DBpedia predicate *spouse*.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Human Computed	0	0	0	0.05	0.1	0.15	0.35	0.42	0.5	0.65	0.65	1	1	1
Algorithm Computed	0	0	0.02	0.05	0.07	0.09	0.29	0.35	0.44	0.7	0.48	0.86	0.89	1

	1	2	3	4	5	6	7	8	9	10
Human Computed	0.2	0.85	0.9	0.9	0.95	1	1	1	1	1
Algorithm Computed	0.07	0.82	0.56	0.69	0.98	0.78	0.91	0.93	1	1

Fig. 3. Confidence results for DBpedia predicate *foundedBy*.

by reporting its execution time. *(iv)* measuring the impact of missing values and errors in the computation of confidence.

For this experiment, we focus on rules for two popular KGs: DBpedia [7] and Yago [35]. DBpedia is a KG derived from Wikipedia, we used version "2016-04". Yago is an open source KG with information from Wikipedia, Wordnet, and GeoNames; we used Yago3.

We mined rules with rule learning systems and then chose a subset of predicates for the evaluation. Predicates were chosen based on two considerations: *(1)* it had at least five positive and negative rules in the corpus; *(2)* the computed confidences for its rules cover a wide range of values. Considering these constraints, we selected seven DBpedia predicates (*spouse*, *foundedBy*, *relative*, *founder*, *publisher*, *employer* and *influencedBy*) and two Yago predicates (*isMarriedTo* and *hasChild*) for manual annotation. For both KGs, we use their online endpoint to have the latest version available. The annotators of the triples are authors of the paper, therefore familiar with the process and considered experts. They checked triples independently and, given that we conducted the evaluation over general purpose KGs, they have been able to use internet resources (e.g., Wikipedia) to verify the validity of the randomly selected claims.

4.1 Accuracy of Confidence Measures

Figure 2 reports the confidence values for 20 rules for the *spouse* predicate in DBpedia. We divided positive and negative rules in two plots and report for every rule both the confidence computed by our method and the *human confidence*. Each point on the x-axis represents an individual rule with its id, and the two bar charts associated with a rule show the confidence measure computed

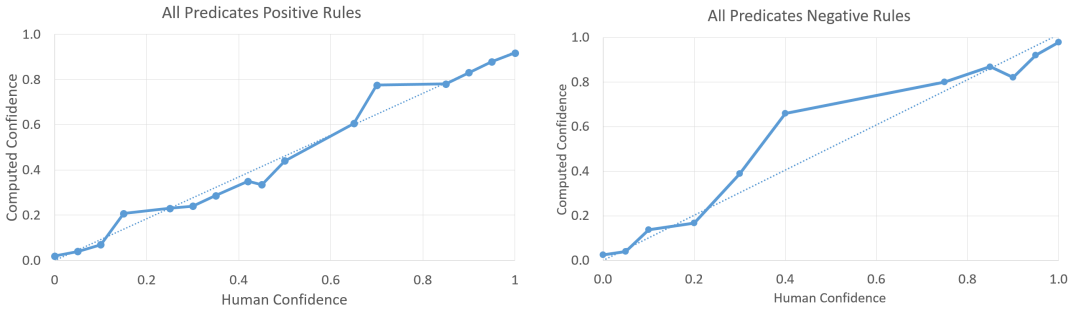
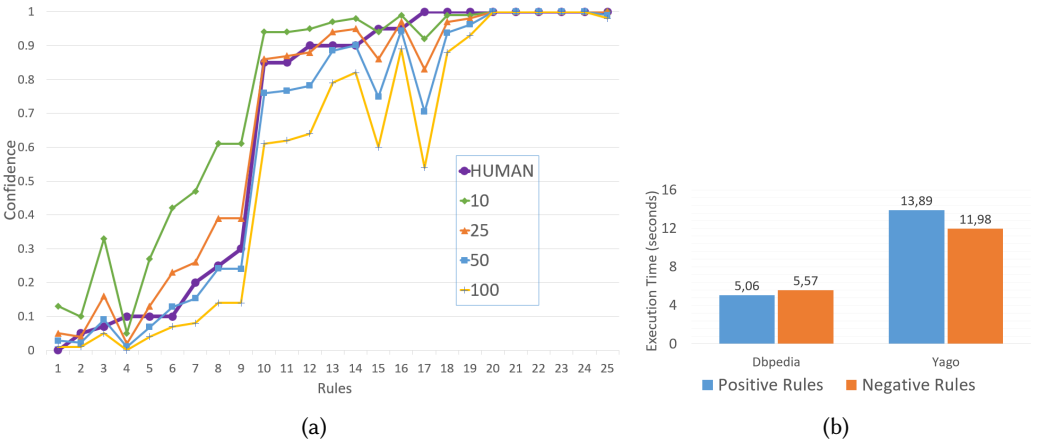


Fig. 4. Average computed and human confidence over rules for all predicates.

either manually and automatically. The mean error is 8.4% for positive rules and 10.4% for negative. Similarly, we report results for 24 rules for the predicate *foundedBy* in Figure 3, with 5.4% mean error for positive and 11.2% for negative rules. Both figures show a highly correlation between the manually computed confidence and the confidence computed by the proposed measures. What is most important for us is that the quality of the computed confidence for negative rules is close to the traditional computed confidence for positive ones, despite computing the quality of negative rules from data is harder.

Figure 4 reports the computed and human confidences for all the rules over the nine predicates (DBpedia *spouse*, *foundedBy*, *relative*, *founder*, *publisher*, *employer*, *influencedBy*; Yago *isMarriedTo* and *hasChild*). We grouped rules by their human confidence and for each group we report a point. The horizontal (x) axis is the human confidence and the vertical (y) axis represents the *average* of the computed confidence for that group of rules. The trend-lines show the similar trends for computed and human confidences, with only 7.8% mean errors for positive rules and 9.0% for negatives. The aggregates results confirm the effectiveness of our measure for computing confidence. More experimental results are reported in Appendix B.



(a) (b)

Fig. 5. (a) Confidence for different values of κ. (b) Avg execution times for computing confidence.

4.2 Effect of the κ Parameter

In previous experiments, we used $\kappa = 25$. To show the effect of this parameter, we report results for executions with different κ values $(10, 25, 50, 100)$ for 25 negative rules over 7 predicates[1]. Figure 5a reports the results with the rules in increasing order of human confidence. The plot shows that the computed confidence for $\kappa = 25$ is the closest to the human confidence. By looking in more detail at rules 4, 15, and 17, we can explain the significant difference between human confidence and computed confidence. Rule 4 is a clearly incorrect rule ($almaMater(object, v0) \wedge country(v0, v1) \wedge birthPlace(subject, v1) \wedge relative(subject, object) \rightarrow \bot$) and this is correctly reflected by its computed confidence (0.02). However, due to the sampling process on 20 triples, two spurious correct triples lead to an estimated human confidence of 0.1. Although rules 15 and 17 seem to be true, they have 30 and 11 *counter_support* triples in DBpedia, respectively. Some of these *counter_supports* are exceptions for the rule and the rest are errors. For example, rule 15 states that a person cannot be in a spouse relation with his parent's spouse. There are 30 *counter_supports* for this rule in DBpedia and by checking them we saw that 22 of them are errors and other 8 *counter_support* triples are exceptions for the rule, such as ancient monarchs and fictional characters. Considering the importance of *counter_support* in formula 8, these numbers reduce the confidence for these rules.

4.3 Computed Confidence Execution Time

We report the execution times of our method in computing the confidence measure. We computed confidence values for 199 positive and 307 negative rules for 4 DBpedia predicates and 36 positive and 40 negative rules for 6 Yago predicates. As reported in Figure 5b, average execution times are within seconds in both KGs despite we are using web APIs and more than 90% of the time goes in collecting responses. Execution times between the two KGs are not comparable as their endpoints are on different servers, they have different sizes in terms of predicates and triples, and different numbers of rules.

4.4 Impact of Quality Issues on Computing Confidence

Two main issues affect the quality of data in KGs: i) factual mistakes, such as incorrect or outdated data, and ii) incompleteness. In this experiment, we study the impact of data issues on the performance of the proposed confidence measure for negative rules. For this task, for rules we used in Section 4.2 (rules are reported in Table 3 in the Appendix), we manually identified *factual mistakes* and *missing* facts in their counter support. We then computed the confidence and observed the changes in the error rate w.r.t. the human confidence.

Factual mistakes are triples that are wrongly considered as counter support, as they are incorrect or outdated. These triples should be removed to have a correct counter support set. For example, some of the triples in the counter support of a rule (# 15) are entities in a *child* relation with themselves. As another example, another rule (#19) states that a person cannot be in a *spouse* relation with someone who died before she was born. This rule is logically correct but there are 13 real errors in the counter support for it in DBpedia.

Missing facts are triples that, based on KG relations, should be considered as counter support but because of incompleteness are not present in the KG. For example, for rule (# 10: $spouse(v0, object) \wedge parent(subject, v0) \wedge spouse(subject, object) \rightarrow \bot$), we observe in DBpedia: $spouse(Kaumualii, Deborah\_Kapule)$, with Kaumualii and Deborah_Kapule assigned to variables v0 and object, respectively; $parent(Kealiiahonui, Kaumualii)$ (subject,v0); and $spouse(Deborah\_Kapule, Kealiiahonui)$ (object,subject). The triple

[1]We report the rules in Appendix B.2.

$spouse(Kealiiahonui, Deborah\_Kapule)$ would be counter support to this rule, but it is not in the KG, despite the presence of $parent(Kealiiahonui, Kaumualii)$ and the fact that the rule $spouse(a, b) \rightarrow spouse(b, a)$ is always true. We therefore count $spouse(Kealiiahonui, Deborah\_Kapule)$ as a missing fact for this rule.

Rule	Hum. Conf.	*C_S* 1	MF	IF	*C_S* 2	Conf. 1	Conf. 2	Conf. MF	Conf. IF
1	0	56	15	0	71	0.05	0.04	0.4	0.05
2	0.05	477	89	0	566	0.04	0.03	0.03	0.04
3	0.07	15	4	0	19	0.16	0.13	0.13	0.16
4	0.1	107	59	0	166	0.02	0.01	0.01	0.02
5	0.1	44	0	0	44	0.13	0.13	0.13	0.13
6	0.1	51	90	0	141	0.23	0.10	0.10	0.23
7	0.2	1	1	0	2	0.26	0.15	0.15	0.26
8	0.25	20	9	0	29	0.20	0.15	0.15	0.20
9	0.3	306	220	0	526	0.39	0.27	0.27	0.39
10	0.85	32	9	0	41	0.86	0.83	0.83	0.86
11	0.85	1	0	0	1	0.87	0.87	0.87	0.87
12	0.9	25	0	0	25	0.88	0.88	0.88	0.88
13	0.9	1	0	0	1	0.94	0.94	0.94	0.94
14	0.9	15	6	0	21	0.95	0.93	0.93	0.95
15	0.95	30	0	22	8	0.88	0.96	0.88	0.96
16	0.95	4	5	0	9	0.97	0.94	0.94	0.97
17	1	11	0	0	11	0.80	0.80	0.80	0.80
18	1	7	1	0	8	0.97	0.96	0.96	0.97
19	1	13	1	13	0	0.98	1	0.98	1
20	1	0	0	0	0	1	1	1	1
21	1	0	0	0	0	1	1	1	1
22	1	0	0	0	0	1	1	1	1
23	1	0	0	0	0	1	1	1	1
24	1	0	0	0	0	1	1	1	1
25	1	1	0	1	0	1	1	1	1

Table 2. Negative rules information: rule #, human confidence obtained from triple annotation (Hum. Conf.), number of missing facts (MF) and incorrect facts (IF), original (*C_S* 1) and updated counter support (*C_S* 2), computed confidence before (*Conf. 1*) and after refining counter support (*Conf. 2*), computed confidence by only adding missing facts (*Conf. MF*) and by only removing incorrect facts (*Conf. IF*).

Adding missing facts and removing incorrect facts affect the counter supports of rules and therefore changes the confidence measure value. For measuring this impact, for every negative rule in this experiment (listed in Table 3), we report in Table 2 its confidence before and after refining their counter supports. We manually checked every rule to compute the new counter support (*newC_S*) after adding missing facts (MF) and removing incorrect facts (IF). The original confidence measure (*Conf. 1*) and the one obtained with the new counter support (*Conf. 2*) are also reported. Identifying incorrect and missed counter supports for negative rules decreases the average error rate of the computed confidence measure w.r.t. human confidence (*Hum. Conf.*, obtained by annotating triples) from 4.3% to 3.3%.

We also report the updated confidence of every rule by only adding missing facts (*Conf. MF*) and by only removing incorrect facts (*Conf. IF*). We added 509 missing facts, which affected 13

rules (every rule has 20.4 missed facts by average), and removed 36 mistakes, which affected 3 rules (1.4 for each rule by average). By adding missing facts only to counter supports (Conf. MF), we decrease the average error rate of the computed confidence measure w.r.t. human confidence from 4.3% to 3.6%, while the average error rate *Conf. IF* is 4.0% by only removing mistakes. We observe that adding missing facts has a bigger positive impact on the confidence computation.

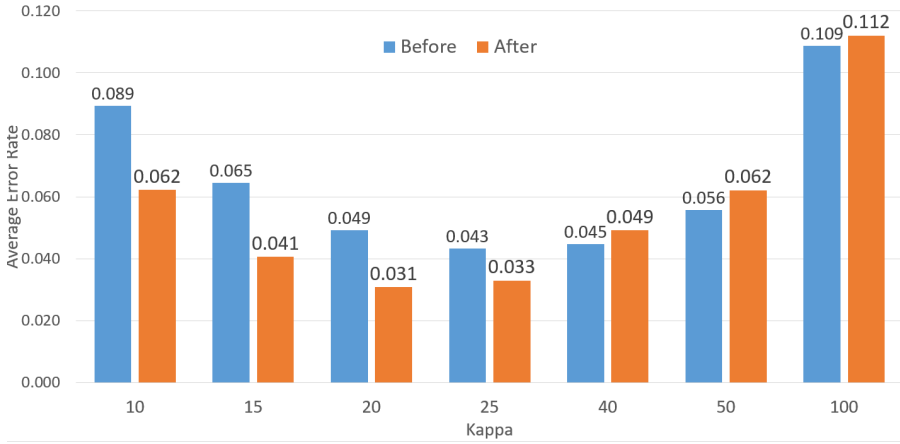


Fig. 6. Computed confidence error rate w.r.t. human quality with and without manual cleaning of the triples for different κ values.

The Effect of KG Cleaning on κ. As discussed in Section 2.3, the correct value of κ depends on the accuracy of a KG. We report the effect on the confidence computation quality when removing missing and incorrect facts with different κ values.

Results in Figure 6 show that cleaning counter supports has a positive impact on lower κ values while it can increase error rate for high values. Even a small amount of cleaning effectively decreases errors and incompleteness in KGs. As κ depends on KG accuracy, the results consistently show that by increasing the accuracy, a lower value for κ (20) has the minimum average error rate. Improvement can be observed for all values smaller than 25, while larger values report worse results, thus confirming that the estimate of the error rate is too big and a smaller κ value should be used.

4.5 Lessons Learned in Annotating Rules

One of the observations from our work is that evaluating rules is a non trivial task. In fact, we advocate that, when possible, rules should be evaluated by looking and annotating the output triples as true or false. Even with a small number of randomly sampled triples, the estimated confidence is usually more reasonable than a human quality evaluation.

On the other hand, it is much faster to come up with a subjective measure of the quality of a rule by looking at its logical form. This can be effective for some rules, but more difficult for others. We conducted experiments on quality evaluation to find more evidence of the gap between the two measures and to better understand what are the more complicated cases for annotators.

We recruited three graduate students, not involved in this work and not familiar with KGs. We gave them the task to assign a *quality evaluation* score to 20 (positive and negative) rules. Each annotator assigned individually a score from 1 to 5 to every rule, where 5 is a completely incorrect rule and 1 is a correct one (see Section 3.1).

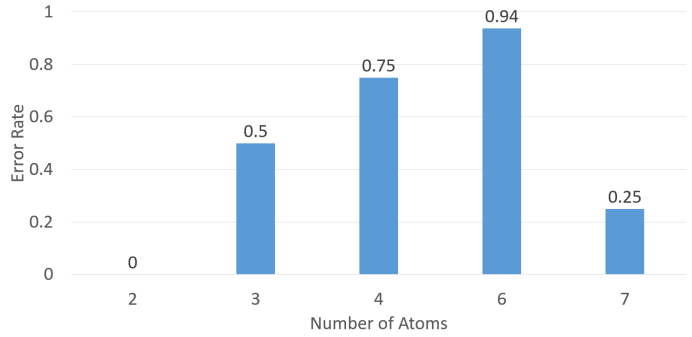


Fig. 7. Impact of number of rules atoms on quality evaluation annotations.

Experimental results show a good correlation between the average of the quality evaluation score from the non-expert annotators and the human confidence values based on triples annotations. Detailed results are reported in Appendix B.3. After a more detailed analysis of the results, the triple annotation based method is clearly closer to the correct evaluation. In fact, quality evaluation scores are not always in agreement and the Kendall's tau-b correlation [25] score over the 20 rules is 0.49 as annotators had different understanding of some of the non-trivial rules.

There are multiple factors that affect the understanding of a rule. An important feature of rules is their type. In our experiments, quality evaluation annotators had a lower error rate in evaluating positive rules (0.50) in comparison to negative ones (0.72). Another factor is the number of atoms in the rule. Figure 7 shows that rules with a higher number of atoms seem to be more difficult to evaluate. The low error rate in Figure 7 is explained by the fact that only two rules have 7 atoms and both of them are positive.

5 RELATED WORK

For *relational data*, dependencies are discovered over the attributes of a given schema and encoded into formalisms, such as Functional Dependencies [2, 22] and Denial Constraints [12, 16]. However, relational techniques cannot be applied to KGs for three main reasons: *(i)* the schema-less nature of RDF data and the open world assumption; *(ii)* traditional approaches rely on the assumption that data is either clean or has a small amount of errors, which is not the case with KGs; *(iii)* even when the algorithms are designed to support more errors [1, 26], there are scalability issues on large RDF datasets: a direct application of relational database techniques on the graph requires the materialization of all possible predicate combinations into relational tables. These reasons have motivated a lot of work on discovering rules for KGs. Recently, algorithms to discover Functional Dependencies on Graphs have been introduced [17]. Traditional approaches rely on general purpose ILP systems [13, 32], while more recently new algorithms have been designed specifically for KG mining. RuDiK is a system that mines both positive and negative rules in RDF KGs [28]. Other approaches are designed for positive rule discovery with an emphasis on scalability [11, 20, 40, 42] and accuracy [15, 19]. Recent work has also started to look at how to use external information in the mining of rules, such as edit history [37] and graph embeddings [21]. For the algorithmic details, we refer the reader to a recent survey on rule mining systems [36].

Other works recommend new facts by using association rule mining techniques [3] or detect data contradictions by discovering axioms concerning properties' domain and range restrictions [38]. Another approach identifies outliers after grouping subjects by type [41]. Finally, the generation of new facts in a graph is related to the task of *link prediction* [9, 23, 33].

6 CONCLUSION

We introduced *RuleHub*, a system managing an open corpus of more than 7,000 rules collected from different systems for three KGs. Our system exposes a web portal to the users to let them retrieve rules for their tasks, add new rules, and refine and annotate rules in the corpus. We believe that metadata play a key role to make rules really effective. Towards this goal, we introduce a new confidence measurement to evaluate the quality of negative rules. An experimental comparison against human computed confidences show that our measure gives valid quality estimations.

Looking forward, we plan to keep enriching the corpus by supporting more kinds of rules, such as graph functional dependencies [17]. Indeed, these dependencies have a syntax that is not modelled by the Horn rules supported in our current system. We will improve the interface to make it more convenient for users and better allow them to contribute to the hub. We also believe that, in terms of metadata, there is a great opportunity in going beyond confidence and extend our system to compute more statistical measures, such as unexpectedness [34]. Finally, we plan to design a novel rule mining algorithm that makes use of the proposed method to estimate the confidence for negative rules.

With the continuous development of KGs as well as the evolution of rule mining techniques, we believe that *RuleHub* will play an important role in collective storing and managing of KG rules.

REFERENCES

[1] Z. Abedjan, C. G. Akcora, M. Ouzzani, P. Papotti, and M. Stonebraker. Temporal rules discovery for web data cleaning. *PVLDB*, 9(4):336–347, 2015.

[2] Z. Abedjan, L. Golab, and F. Naumann. Data profiling: A tutorial. In *SIGMOD*, pages 1747–1751, 2017.

[3] Z. Abedjan and F. Naumann. Amending RDF entities with new facts. In *ESWC*, pages 131–143, 2014.

[4] N. Ahmadi, V.-P. Huynh, V. V. Meduri, S. Ortona, and P. Papotti. Mining expressive rules in knowledge graphs. *Journal of Data and Information Quality (JDIQ)*, 12(2):1–27, 2020.

[5] N. Ahmadi, J. Lee, P. Papotti, and M. Saeed. Explainable fact checking with probabilistic answer set programming. In *Conference for Truth and Trust Online (TTO)*, 2019.

[6] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web. In *WWW*, pages 1265–1266, 2008.

[7] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia-A crystallization point for the web of data. *Web Semantics: science, services and agents on the WWW*, 7(3):154–165, 2009.

[8] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[10] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr, and T. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313, 2010.

[11] Y. Chen, S. Goldberg, D. Z. Wang, and S. S. Johri. Ontological pathfinding. In *SIGMOD*, pages 835–846, 2016.

[12] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013.

[13] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data mining and knowledge discovery*, 3(1):7–36, 1999.

[14] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610, 2014.

[15] M. Duc Tran, C. d'Amato, B. T. Nguyen, and A. G. B. Tettamanzi. Comparing rule evaluation metrics for the evolutionary discovery of multi-relational association rules in the semantic web. In *Genetic Programming*, 2018.

[16] W. Fan and F. Geerts. *Foundations of Data Quality Management.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[17] W. Fan, C. Hu, X. Liu, and P. Lu. Discovering graph functional dependencies. In *SIGMOD*, pages 427–439, 2018.

[18] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. Welty. Building Watson: An overview of the DeepQA project. *AI Mag.*, 31(3):59–79, 2010.

[19] M. H. Gad-Elrab, D. Stepanova, J. Urbani, and G. Weikum. Exception-enriched rule learning from knowledge graphs. In *ISWC*, 2016.

[20] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal*, 24(6):707–730, 2015.

[21] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Jointly embedding knowledge graphs and logical rules. In *EMNLP*, pages 192–202, 2016.
[22] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999.
[23] V. Huynh and P. Papotti. A benchmark for fact checking algorithms built on knowledge bases. In *Conference on Information and Knowledge Management CIKM*, pages 689–698, 2019.
[24] Internet Engineering Task Force (IETF). The javascript object notation (JSON) data interchange format. https://tools.ietf.org/html/std90.
[25] M. G. Kendall. The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251, 1945.
[26] S. Kruse and F. Naumann. Efficient discovery of approximate dependencies. *PVLDB*, 11(7):759–772, 2018.
[27] C. Mangold. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23–34, 2007.
[28] S. Ortona, V. V. Meduri, and P. Papotti. Robust discovery of positive and negative rules in knowledge bases. In *34th IEEE International Conference on Data Engineering, ICDE*, pages 1168–1179, 2018.
[29] S. Ortona, V. V. Meduri, and P. Papotti. RuDiK: Rule discovery in knowledge bases. *PVLDB*, 11(12):1946–1949, 2018.
[30] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. In *International semantic web conference*, pages 30–43, 2006.
[31] M. Schmidt, M. Meier, and G. Lausen. Foundations of sparql query optimization. In *ICDT*, pages 4–33, 2010.
[32] S. Schoenmackers, O. Etzioni, D. S. Weld, and J. Davis. Learning first-order horn clauses from web text. In *Empirical Methods in Natural Language Processing*, pages 1088–1098, 2010.
[33] B. Shi and T. Weninger. ProjE: Embedding projection for knowledge graph completion. In *AAAI*, 2017.
[34] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. Knowl. Data Eng.*, 8(6):970–974, 1996.
[35] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge unifying wordnet and wikipedia. In *WWW*, pages 697–706, 2007.
[36] F. M. Suchanek, J. Lajus, A. Boschin, and G. Weikum. Knowledge representation and rule mining in entity-centric knowledge bases. In *Reasoning Web Summer School*, pages 110–152, 2019.
[37] T. P. Tanon, C. Bourgaux, and F. M. Suchanek. Learning how to correct a knowledge base from the edit history. In *WWW*, pages 1465–1475, 2019.
[38] G. Töpper, M. Knuth, and H. Sack. DBpedia ontology enrichment for inconsistency detection. In *I-SEMANTICS*, 2012.
[39] D. Vrandečić and M. Krötzsch. Wikidata: A free collaborative knowledge base. *Comm. of the ACM*, 57(10):78–85, 2014.
[40] Z. Wang and J. Li. RDF2Rules: Learning rules from RDF knowledge bases by mining frequent predicate cycles. *CoRR*, abs/1512.07734, 2015.
[41] D. Wienand and H. Paulheim. Detecting incorrect numerical data in DBpedia. In *ESWC*, 2014.
[42] V. Zeman, T. Kliegr, and V. Svátek. Rdfrules preview: Towards an analytics engine for rule mining in RDF knowledge graphs. In *RuleML+ RR (Supplement)*, 2018.

A RULE MINING SYSTEMS

We briefly describe the two systems that we used to generate most of the rules in our corpus.

AMIE mines positive rules from KGs and it works based on the assumption that the target KG fits into memory. AMIE lists all the predicates in the KG and inserts each of them as head of the rule. Once the head is filled, the system tries to expand the rule by pivoting on one of the variables of the current predicate and looks for predicates sharing the same variable with high coverage in the KG. The coverage of a rule is penalized with the partial closed world assumption, where the set of negative examples for a given pair (x, y) and a target predicate p is all those pairs where x is connected through p to an entity different from y. It then ranks rules according to a support function.

RuDiK is a disk-based system that mines positive and negative rules over KGs. RuDiK extracts both positive and negative rules and finds the smallest set of rules that cover most of the positive examples and few negative examples. RuDiK uses positive and negative example as Generation and Validation sets. Next, it retrieves all valid paths that start from every node in those sets. It finally computes the coverage for each path and traverses only paths that can generate promising rules.

We generate rules using AMIE and RuDiK with their default settings as suggested from the authors. These settings imply a maximum number of atoms in the body of a rule equals to 3 for AMIE and 5 with RuDiK.

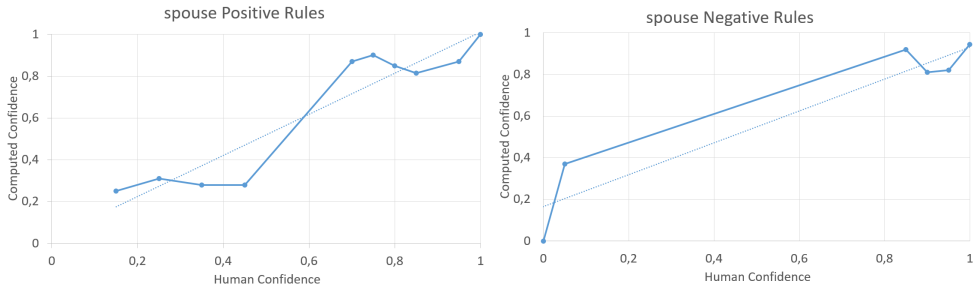


Fig. 8. Confidence measures of the rules for DBpedia:spouse.

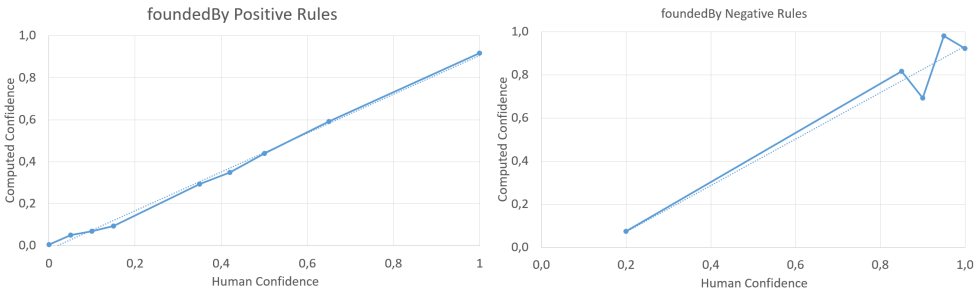


Fig. 9. Confidence measures of the rules for DBpedia:foundedBy.

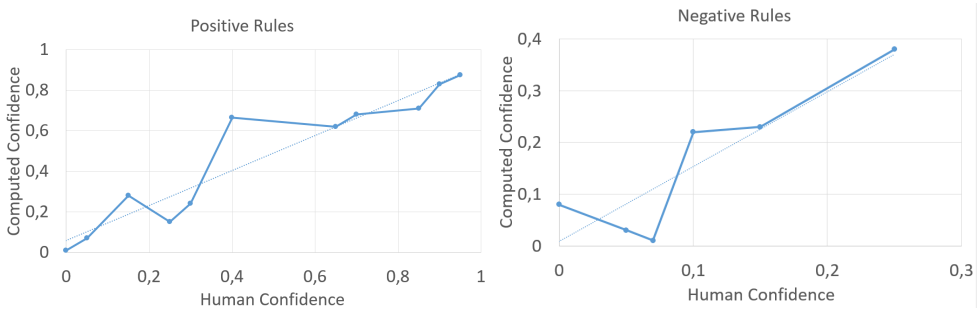


Fig. 10. Confidence measures of the rules for DBpedia:relative and DBpedia:publisher (union of the two rule sets).

B MORE EXPERIMENTAL RESULTS

B.1 Other Predicates Confidence Results

In this section, we report more results about our experiments. Figure 8 shows a comparison between *human confidence* and *computed confidence* for the rules involving DBpedia:spouse. Figure 9 shows the results for the rules involving the DBpedia:foundedBy predicate. Figure 10 reports the results for the union of the rules for two DBpedia predicates: *relative* and *publisher*. The mean error for these predicates is 8.36% for positive rules and 8.71% for negative rules. The correlation between *human confidence* and *computed confidence* for both positive and negative rules for these predicates is evident from the plots.

B.2 A Sample of Rules Used for Evaluations

We report in Table 3 the negative rules that we used for evaluating our confidence measures. Figure 5a shows the computed confidence of these rules for different κ values.

In Table 4 we report the positive rules that we used for evaluating our confidence measures.

B.3 Quality Evaluation Measure Experiment

In this experiment, we assess the *quality evaluation* measure (Section 3.1) by comparing it with the human computed confidence. We randomly chose 20 annotated positive and negative rules from the RuleHub database and asked three non-experts to assign quality evaluation score to each one of them. We provided three annotators with some general guidelines about logical rules and KGs, as well as with a brief description in English for every rule to help them understand the semantic meaning of the rule. Each annotator assigned individually a score from 1 to 5 to every rule, where 5 is a completely incorrect rule and 1 is a correct one.

In Figure 11, we report on the horizontal axis the average of annotators quality evaluation score for the rules (between 1 and 5) and in the vertical axis their human confidences (based on triple annotation). The results show a clear correlation between the two measures, with the exception of the two rules marked with red points. The first exception rule (red point on the right) is $predecessor(object, v0) \wedge spouse(v0, v1) \wedge predecessor(subject, v1) \rightarrow spouse(subject, object)$, which has 0.35 human confidence and the average of annotators score is 4.7. This rule is a positive rule which states that if the job predecessors of two given persons are married then these two persons are married. Even though it was difficult for annotators to find supports for this rule, there are 547 support triples for this rule in DBpedia. In fact, this is true in most of the cases when kings or presidents are replaced. For example, given that $predecessor(Donald\_Trump, Barak\_Obama)$ and

	Rule
1	$owningCompany(object, v0) \wedge manufacturer(v1, v0) \wedge computingPlatform(subject, v1) \wedge publisher(subject, object) \rightarrow \bot$
2	$relative(v0, object) \wedge child(v0, subject) \wedge parent(subject, v0) \wedge relative(subject, object) \rightarrow \bot$
3	$country(subject, v0) \wedge location(v1, v0) \wedge successor(v1, object) \wedge publisher(subject, object) \rightarrow \bot$
4	$almaMater(object, v0) \wedge country(v0, v1) \wedge birthPlace(subject, v1) \wedge relative(subject, object) \rightarrow \bot$
5	$employer(v0, object) \wedge birthYear(v0, v1) \wedge deathYear(subject, v1) \wedge employer(subject, object) \rightarrow \bot$
6	$publisher(subject, v0) \wedge product(object, v0) \wedge publisher(subject, object) \rightarrow \bot$
7	$owningCompany(subject, object) \wedge owner(v0, object) \wedge author(v0, object) \wedge type(object, Organisation) \wedge type(subject, Organisation) \wedge foundedBy(subject, object) \rightarrow \bot$
8	$publisher(subject, v0) \wedge parentCompany(v1, v0) \wedge successor(object, v1) \wedge publisher(subject, object) \rightarrow \bot$
9	$birthYear(object, v0) \wedge birthYear(subject, v1) \wedge > (v0, v1) \wedge influencedBy(subject, object) \rightarrow \bot$
10	$spouse(v0, object) \wedge parent(subject, v0) \wedge spouse(subject, object) \rightarrow \bot$
11	$writer(v0, object) \wedge artist(v0, subject) \wedge foundedBy(subject, object) \rightarrow \bot$
12	$predecessor(v0, object) \wedge predecessor(subject, v0) \wedge spouse(subject, object) \rightarrow \bot$
13	$associatedMusicalArtist(object, subject) \wedge associatedBand(object, v0) \wedge associatedMusicalArtist(v0, subject) \wedge foundedBy(subject, object) \rightarrow \bot$
14	$deathYear(subject, v0) \wedge activeYearsStartYear(object, v1) \wedge < (v0, v1) \wedge spouse(subject, object) \rightarrow \bot$
15	$spouse(v0, object) \wedge child(v0, subject) \wedge spouse(subject, object) \rightarrow \bot$
16	$deathPlace(object, v0) \wedge region(v0, v1) \wedge birthPlace(subject, v1) \wedge type(subject, Royalty) \wedge type(object, Royalty) \wedge spouse(subject, object) \rightarrow \bot$
17	$birthPlace(object, v0) \wedge deathPlace(object, v0) \wedge predecessor(object, subject) \wedge spouse(subject, object) \rightarrow \bot$
18	$spouse(subject, v0) \wedge parent(object, v0) \wedge parent(object, subject) \wedge spouse(subject, object) \rightarrow \bot$
19	$deathDate(object, v1) \wedge birthYear(subject, v0) \wedge > (v0, v1) \wedge spouse(subject, object) \rightarrow \bot$
20	$spouse(v0, object) \wedge parent(subject, v0) \wedge predecessor(subject, object) \wedge spouse(subject, object) \rightarrow \bot$
21	$successor(object, subject) \wedge parent(subject, v0) \wedge spouse(object, v0) \wedge spouse(subject, object) \rightarrow \bot$
22	$parent(object, v0) \wedge spouse(subject, v0) \wedge successor(subject, object) \wedge spouse(subject, object) \rightarrow \bot$
23	$artist(v0, subject) \wedge recordedIn(v0, v1) \wedge birthPlace(object, v1) \wedge foundedBy(subject, object) \rightarrow \bot$
24	$foundedBy(subject, v0) \wedge deathPlace(v0, v1) \wedge locationCountry(object, v1) \wedge type(object, Company) \wedge type(subject, Company) \wedge foundedBy(subject, object) \rightarrow \bot$
25	$foundingYear(subject, v0) \wedge activeYearsStartYear(object, v1) \wedge < (v0, v1) \wedge founder(subject, object) \rightarrow \bot$

Table 3. Examples of negative rules.

$predecessor(Melanie\_Trump, Michelle\_Obama)$, since $spouse(Barak\_Obama, Michelle\_Obama)$ then $spouse(Donald\_Trump, Melanie\_Trump)$. The second rule $deathDate(subject, v0) \wedge < (v0, v1) \wedge activeYearsStartYear(object, v1) \wedge spouse(subject, object) \rightarrow \bot$ has 0.9 human confidence but a lower evaluation score of 2.7. This rule states that a person (object) can not be in spouse relationship with a person (subject) who died before object has started his career. In both cases, the triple annotation based method seem to be closer to the correct evaluation. We observe that in these two cases there was always one non-expert making the evaluation very different from the one based on triple annotation, thus skewing the results. In fact, the Kendall's tau-b correlation [25] score for annotating the 20 rules is 0.49, which is a fair agreement but also shows that the annotators had different understandings of some of the non-trivial rules. These results confirm that the triple based annotation is more time-consuming but also more reliable than the qualitative analysis of the rules.

	Rule
1	$recordLabel(v0, subject) \wedge foundedBy(object, v0) \rightarrow foundedBy(subject, object)$
2	$parentCompany(subject, v0) \wedge successor(v1, v0) \wedge foundedBy(v1, object) \rightarrow foundedBy(subject, object)$
3	$producer(v0, object) \wedge producer(v0, subject) \rightarrow relative(subject, object)$
4	$predecessor(v0, object) \wedge spouse(v1, v0) \wedge successor(subject, v1) \rightarrow spouse(subject, object)$
5	$author(subject, v0) \wedge author(v1, v0) \wedge publisher(v1, object) \rightarrow publisher(subject, object)$
6	$relative(v0, object) \wedge spouse(v0, subject) \rightarrow relative(subject, object)$
7	$child(object, v0) \wedge spouse(v0, subject) \rightarrow relative(subject, object)$
8	$successor(subject, v0) \wedge parent(v0, object) \rightarrow spouse(subject, object)$
9	$relative(subject, v0) \wedge relative(v0, object) \wedge child(object, v0) \rightarrow relative(subject, object)$
10	$spouse(subject, v0) \wedge spouse(v1, v0) \wedge spouse(v1, object) \rightarrow spouse(subject, object)$
11	$developer(subject, object) \wedge developer(v0, object) \wedge composer(v0, v0) \rightarrow publisher(subject, object)$
12	$child(subject, v0) \wedge child(object, v0) \rightarrow spouse(subject, object)$
13	$relative(object, subject) \wedge birthPlace(object, v0) \wedge birthPlace(subject, v0) \rightarrow relative(subject, object)$
14	$spouse(object, subject) \rightarrow spouse(subject, object)$
15	$parent(v0, object) \wedge parent(v0, subject) \rightarrow spouse(subject, object)$

Table 4. Examples of positive rules.

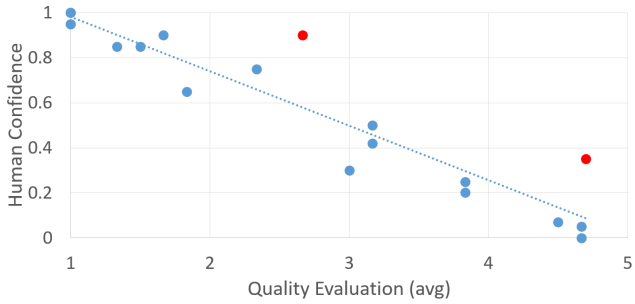


Fig. 11. Quality Evaluation (subjective value between 1 and 5) vs Human Confidence (derived from triple annotation).

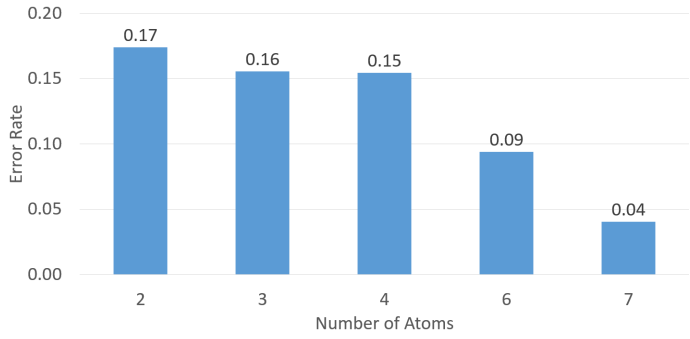


Fig. 12. Impact of number of atoms on the error rate for the computed confidence w.r.t. human confidence.

B.4 Important factors in Computing Rules Confidence

We report some observations from comparing computed rule confidence with human confidence. There are 105 manually annotated rules in the RuleHub corpus: 56 positive and 49 negative ones. Comparing computed confidence values with human confidence values shows that there is a lower error rate for positive rules (12.9%) in comparison to negative rules (16.5%). The number of atoms in a rule also affects the computed confidence. Figure 12 shows that the computed confidence has a better accuracy for rules with a higher number of atoms. Intuitively, the more a rule is specific, the less likely it is that it is satisfied with incorrect triples. In fact, assuming most of the errors are independent from each other (i.e., they are no systematic), it is very unlikely that multiple atoms show a pattern by chance.

Conditional rules are rules that apply only for a subset of data and therefore come with at least two extra atoms compared to the same rules without type constraints [4]. For example, $party(object, v0) \wedge party(subject, v1) \wedge type(subject, Politician) \wedge type(object, OfficeHolder) \wedge spouse(subject, object) \rightarrow \bot$ is a conditional rule that applies where subject has type *Politician* and object has type *OfficeHolder*. In accordance with our observation above, conditional rules show on average a lower error rate in the computation of their confidence (11.6%) in comparison to the average for rules that are not conditional (15.1%).

C RULEHUB WEB PAGE

In this section, we report more details about the web interface of RuleHub. A step-by-step user manual is available at http://rudik.eurecom.fr/user-manual.

C.1 Add new rules

To enrich the corpus, RuleHub allows users to easily add more rules into the system through a simple form with the following mandatory fields:

- Knowledge Base
- Predicate
- Rule Type
- Premise
- Human Confidence
- Quality Evaluation

When a rule is submitted, the system computes the hash code to check if the rule already exists in the corpus. If it is a new rule, it will automatically calculate the support score based on the number of matching cases and not-matching cases over the corresponding KG (see Section 2.3), then store it into the database as a rule to be validated. These rules will not be published until they are approved by administrators of the site. Figure 13 reports a screenshot of the main page of the *RuleHub* portal. Figure 14 reports a screenshot of the page to add rules. Figure 15 reports a screenshot of the rule management page.

C.2 Evaluate rules

Administrators can get samples of the output of a rule execution and label them through RuleHub. Figure 16 reports a screenshot of the evaluation page.

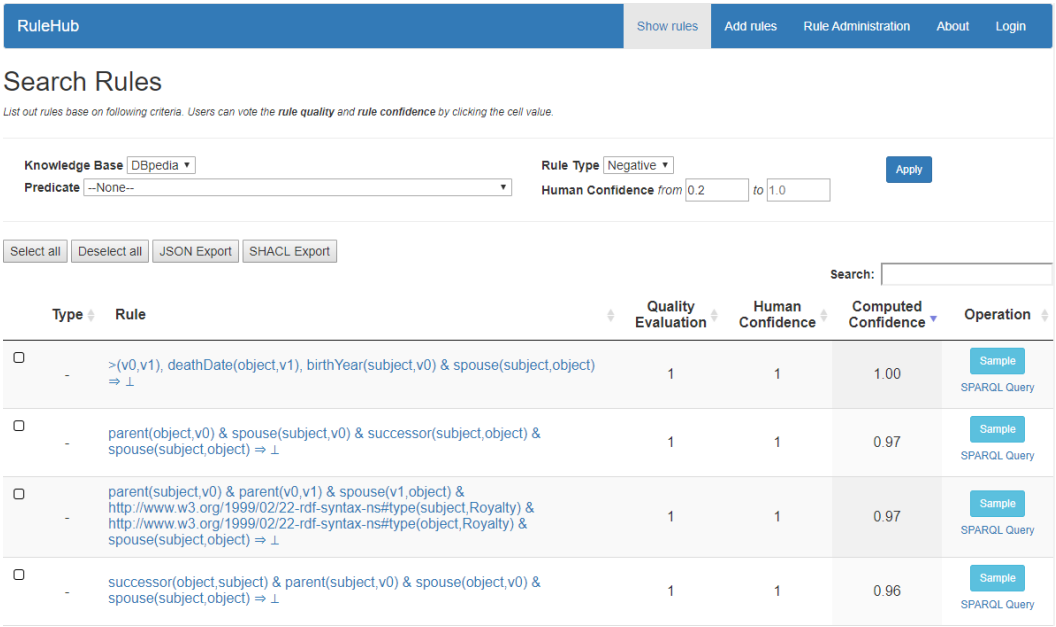


Fig. 13. Screenshot of RuleHub web portal - Search for rules.

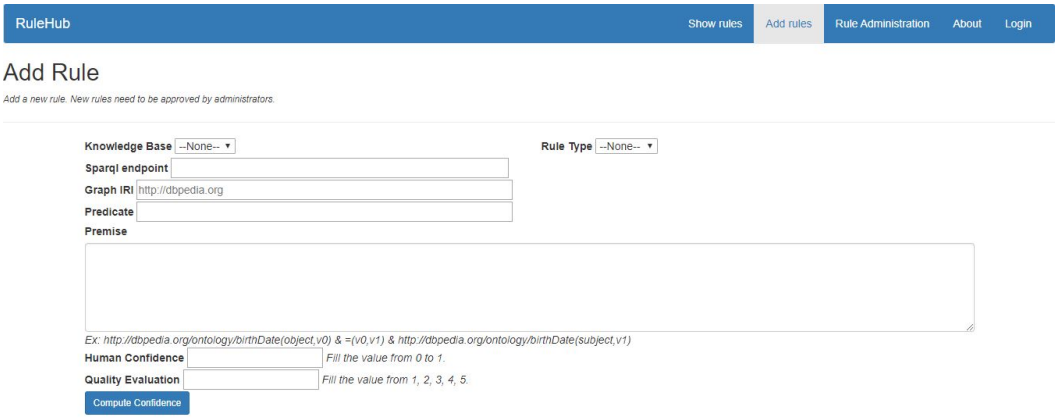


Fig. 14. Screenshot of RuleHub web portal - Add rule.

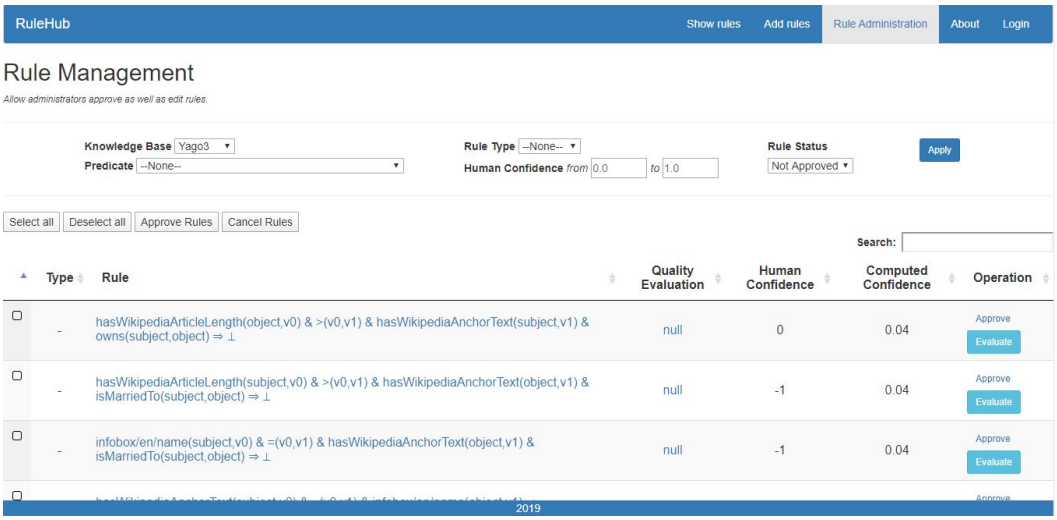


Fig. 15. Screenshot of RuleHub web portal - Rule Management.

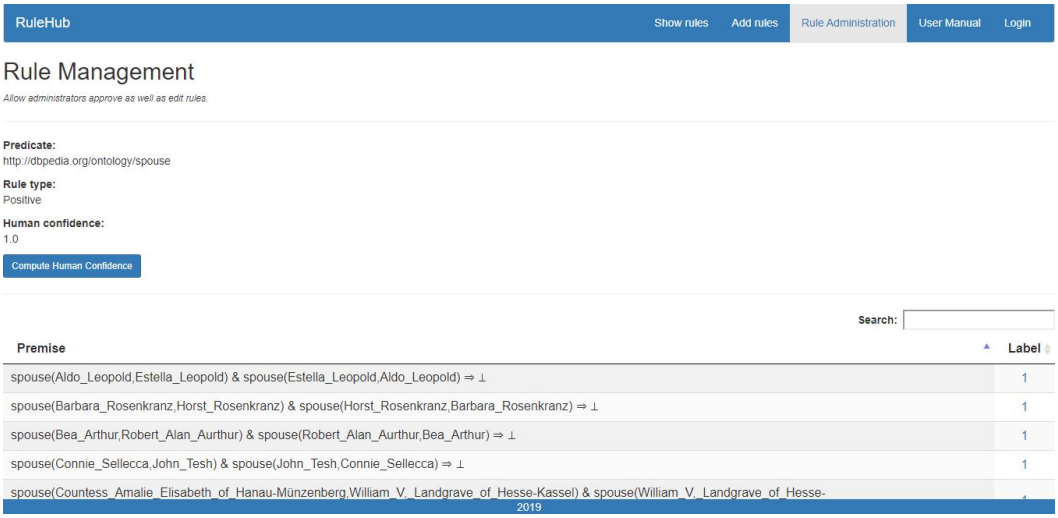


Fig. 16. Screenshot of RuleHub web portal - Rule Evaluation.