Wikidata Logical Rules and Where to Find Them

Naser Ahmadi
naser.ahmadi@eurecom.fr
EURECOM
France

Paolo Papotti
papotti@eurecom.fr
EURECOM
France

ACM Reference Format:
Naser Ahmadi and Paolo Papotti. 2021. Wikidata Logical Rules and Where to Find Them. In *Companion Proceedings of the Web Conference 2021 (WWW '21 Companion), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3442442.3452343

1 INTRODUCTION

Wikidata is a knowledge base (KB) representing data with a large collections of interconnected entities. Statements describe the property value for any item, such as *London* is *an instance* of *city*. Wikidata supports all kinds of applications and it is the structured data storage for its Wikimedia sister projects.

A benefit of a KB is the ability to define *constraints* over it. Unfortunately, KBs in general do not come with a set of logical rules and Wikidata is no exception. Despite the positive impact of users contributing to the coverage and the quality of Wikidata information, rich logical rules are not part of this effort. Property constraints[1] have been defined for over 8K items[2], but these are mostly syntactic checks defined over the value of a property, such as the fact that an IMDb ID should follow a particular regular expression.

We are interested in soft (approximate) constraints expressed as dependencies (or *logical rules*), such as the constraint that "a person cannot be born after one of her children". Such rules have proven to be useful for error detection [4], adding missing facts [3], executing queries faster, and reasoning [1].

Not only these rules are not stated in Wikidata, but, to the best of our understanding, a way to express them as constraints is still to be defined in the repository. Moreover, even if primitives get exposed to the users for this task, manually crafting such rules is difficult as it requires both domain and technical expertise. Finally, once a set of semantically valid rules has been identified, these 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 exact, i.e., 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 has a very high confidence, but it is not exact. Confidence not only is key to rank rules for user validation and refinement, but it is also used in applications that rely on reasoning with soft constraints.

[1]https://www.wikidata.org/wiki/Wikidata:WikiProject_property_constraints
[2]https://w.wiki/YLS

ACM ISBN 978-1-4503-8313-4/21/04.
https://doi.org/10.1145/3442442.3452343

	Rule	*C*	*H*	*Q*
Positive	$P185[doctoralStudent](o,s) \rightarrow P184[doctoralAdvisor](s,o)$	1	1	1
	$P22[father](o,s) \rightarrow P40[child](s,o)$	.99	1	1
	$P26[spouse](o,s) \rightarrow P26(s,o)$	.99	1	1
	$P40[child](s,v) \wedge P40(o,v) \rightarrow P26(s,o)$	.88	.9	1
	$P800[notableWork](o,s) \rightarrow P170[creator](s,o)$	.72	.7	1
Negative	$P40(o,v) \wedge P25[mother](v,s) \wedge P40(s,o) \rightarrow \bot$	.94	1	1
	$P1038[relative](s,o) \wedge P40(s,o) \rightarrow \bot$	.93	1	1
	$P25(o,v_0) \wedge P26(v_1,v_0) \wedge P112[foundedBy](s,v_1) \wedge P112(s,o) \rightarrow \bot$	.89	1	3
	$P180[depicts](s,o) \wedge P170[creator](s,o) \rightarrow \bot$	.32	.2	3

Table 1: Examples of rules mined on Wikidata. We report between square brackets the label (e.g., spouse) for the Wikipedia item IDs (e.g., P26) to favor readability.

Collecting a set of high quality rules is an essential but challenging task to curate Wikidata and to improve the performance of the applications built on it. The goal of our work is to create a large collection of rules for Wikidata with their confidence measure. In this abstract, we report on two directions we have been exploring to obtain such rules, our results, and how we believe the Wikimedia community could benefit from this effort.

2 SEARCHING LOGICAL RULES

We first introduce logical rules and then describe two methods that we are evaluating for collecting Wikidata rules. The first one is a data mining approach, while the second one is based on the idea of translating rules from an existing corpus of DBpedia rules.

Logical Rules. A logical rule has the form $B \rightarrow h(x,y)$, where $h(x,y)$ is a single atom or the bottom type ($\bot$), while B is a conjunction of atoms $B_1(z_1,z_2) \wedge B_2(z_3,z_4) \wedge \cdots \wedge B_n(z_{2n-1},z_{2n})$. An atom is a predicate connecting two variables, two entities, an entity and a variable, or a variable and a constant (string or number).

We consider two kinds of rules. The first kind are *positive rules*, such as the first rule in Table 1, which identify relationships between entities, e.g., "if someone is the doctoral student of a second person, then the second person is her advisor", or "if two persons have a child in common, they are in the spouse relation". The second kind are *negative rules*, with $\bot$ in the conclusion, which identify data contradictions, e.g., "if two persons are in the relative relation, one cannot be the spouse of the other". A fact, or a contradiction, is derived from a rule if all the variables in the premise of the rule can be replaced with constants from the KB.

Rule Mining. Several mining methods have been proposed to identify rules in large KBs [3, 4]. These approaches are effective, but computationally expensive and leave to the user the selection and

	Rule	*C*	*H*	*Q*
Positive	$P40[child](o,v) \wedge P25[mother](v,s) \rightarrow P40(s,o)$	.94	1	1
	$P1038[relative](o,s) \rightarrow P1038(s,o)$	.6	1	1
	$P144[basedOn](v_0,o) \wedge P144(v_0,v_1)$ $\wedge P50[author](s,v_1) \rightarrow P144(s,o)$	.7	.75	2
	$P166[awardReceived](s,v_0)$ $\wedge P118[league](v_1,v_0) \wedge P166(v_1,o) \rightarrow P166(s,o)$	.25	.3	4
Negative	$P569[dateBirth](s,v_0) \wedge P570[dateDeath](o,v_1)$ $\wedge > (v_0,v_1) \wedge P26(s,o) \rightarrow \perp$	1	1	1
	$P26[spouse](o,v) \wedge P26(s,v) \wedge P26(s,o) \rightarrow \perp$	.99	1	1
	$P185[doctoralStudent](s,o) \wedge P185(o,s) \rightarrow \perp$	.95	1	1
	$P144(s,o) \wedge P86[composer](s,o) \rightarrow \perp$	.95	1	2

Table 2: Examples of DBpedia rules translated to Wikidata.

	Rule	*# stms*
Posit.	$P1038[relative](s,o) \rightarrow P1038(o,s)$	13,690
	$P40[child](o,v_0) \wedge P25[mother](v_0,s) \rightarrow P40(s,o)$	226
	$P185[doctoralStud.](o,s) \rightarrow P184[doctoralAdv.](s,o)$	25
Negat.	$P569(s,v_0) \wedge P570(o,v_1) \wedge > (v_0,v_1) \wedge P26(s,o) \rightarrow \perp$	689
	$P22[father](o,s) \wedge P40(s,o) \rightarrow \perp$	41
	$P185(o,s) \wedge P185(s,o) \rightarrow \perp$	17

Table 3: Examples of rules with the # of missing (top) and incorrect (bottom) statements detected by every rule in Wikidata.

the refinement of the mined rules. For this approach, we use a state of the art method for mining declarative rules over RDF KBs [4]. We use a RDF dump of Wikidata, which has been stripped of metadata such as qualifiers and references to other KBs. We mined 80 positive and negative rules and report a sample in Table 1.

Translating DBpedia Rules. In this method, we convert the rules that have been mined over the DBpedia KB in previous work [2]. For every DBpedia rule, we translate its predicates into the equivalent Wikidata properties. For example, property $P184$ in Wikidata corresponds to predicate *DoctoralAdvisor* in DBpedia. We use the *owl:equivalentProperty* information to generate a mapping between the two KBs for *59* properties. With this method, we obtained *241* Wikidata rules. A sample of these rules is shown in Table 2.

3 EXPERIMENTS

Our experiments show how we (i) verify that we obtain rules of good quality and (ii) estimate effectively the confidence of the rules.

Metrics. For measuring rule confidence and quality, we use three metrics [2]. *Computed Confidence* (**C**) is estimated by measuring the rule support, i.e., we count the number of KB facts that support or violate the rule. *Human Confidence* (**H**) is computed by emanually annotating 20 random samples of a rule's output. The fraction of correct new facts or errors is used to estimate the confidence. *Quality Evaluation* (**Q**) is a subjective assessment of the correctness of the rule. The score varies between 1 (rule is always true) and 5 (rule is illogical), we report the average for two annotators.

Results. As reported in the examples in Tables 1 and 2, both methods identify rules with high confidence. We report also examples of

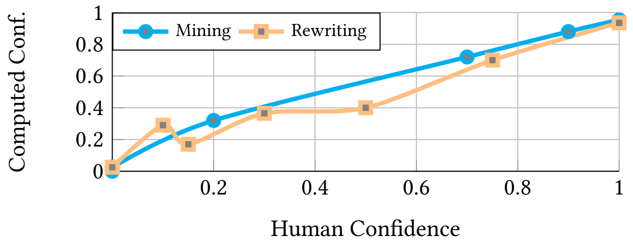

Figure 1: Human and computed confidence comparison.

rules with lower confidence values to show that the computed confidence correlates nicely with measures based on human annotation. We also report in Figure 1 the computed and human confidences for 40 rules over 18 properties. We grouped rules by their human confidence (H) 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 (C) for that group of rules. The plot shows similar correlations between computed and human confidences for rules obtained with both methods.

Finally, we report in Table 3 the number of missing and incorrect statements (stms) identified by a sample of exact positive and negative rules, respectively. For example, the first positive rule identifies almost 14k statements when the rule is instantiated but the conclusion is not in the KB. Similarly, the first negative identifies 689 statements in which the subject is in the spouse relationship with an object who died before the subject was born.

4 ENRICH WIKIDATA

While our effort shows that good rules can be gathered for Wikidata, we still need to make progress to be able to fully transfer this knowledge into the KB. Adding the building blocks to define such logical rules to the Wikidata infrastructure is a mandatory stepping stone, but the implementation is not obvious because of the uncertain nature of most rules. Even if a few hundreds exact rules can be defined, the majority of the rules are not exact. What is the right way to expose and enforce them in Wikidata?

One way to go from the logical form of a rule with high confidence to a fully implemented constraint is to identify and define the exceptions to the rule. Going back to the example about capitals and countries, it should be implemented as an exact rules with 15 exceptions. We are studying how to automatically go from non-exact rules to exact rules with either *more conditions* (that narrow their scope) or with *exceptions*. Discovering such rules is a challenging problem, but we believe our effort will equip the Wikidata community with the right tools to define and enforce these constraints, ultimately building a bigger and better KB.

REFERENCES

[1] Naser Ahmadi, Joohyung Lee, Paolo Papotti, and Mohammed Saeed. 2019. Explainable Fact Checking with Probabilistic Answer Set Programming. In *Conference for Truth and Trust Online (TTO)*.
[2] 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. *Journal of Data and Information Quality (JDIQ)* 12, 4 (2020), 1–22.
[3] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. 2020. Fast and exact rule mining with amie 3. In *European Semantic Web Conference*. Springer, 36–52.
[4] Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. 2018. Robust Discovery of Positive and Negative Rules in Knowledge Bases. In *34th IEEE International Conference on Data Engineering, ICDE*. 1168–1179.