Detecting Synonymous Properties by Shared Data-driven Definitions

Jan-Christoph Kalo, Stephan Mennicke, Philipp Ehler, Wolf-Tilo Balke

Institut für Informationssysteme
Technische Universität Braunschweig
Mühlenpfordtstraße 23
38106 Braunschweig, Germany
{kalo,mennicke,balke}@ifis.cs.tu-bs.de
p.ehler@tu-bs.de

Abstract. Knowledge graphs have become an essential source of entity-centric information for modern applications. Today's KGs have reached a size of billions of RDF triples extracted from a variety of sources, including structured sources and text. While this definitely improves completeness, the inherent variety of sources leads to severe heterogeneity, negatively affecting data quality by introducing duplicate information. We present a novel technique for detecting synonymous properties in large knowledge graphs by mining interpretable definitions of properties using association rule mining. Relying on such shared definitions, our technique is able to mine even synonym rules that have only little support in the data. In particular, our extensive experiments on DBpedia and Wikidata show that our rule-based approach can outperform state-of-the-art knowledge graph embedding techniques, while offering good interpretability through shared logical rules.

Keywords: Synonym Detection · Association Rule Mining · Knowledge Graphs

1 Introduction

In recent years, knowledge graphs have gained more attention because of the popularity of projects like the Google Knowledge Graph [5], Wikidata [25], DBpedia [2], Freebase [3], and YAGO [22]. The size of these knowledge graphs nowadays comprises hundreds of millions of entities associated by ten thousands of properties, providing a comprehensive knowledge repository for several modern applications, e. g., semantic search, question answering and natural language understanding.

The size of these knowledge graphs has steadily been growing over the last years, due to advances in relation extraction and open information extraction. Often large knowledge graphs are created manually in collaborative knowledge graph projects [25], automatically by extracting information from text or tables [2], by integrating existing knowledge into a single ontology, or by a combination of these three methods. However, integrating knowledge from various

sources and by different curators into a single knowledge graph comes with serious heterogeneity issues in practice. Particularly, duplicate concepts, either entities, classes or properties, may cause problems in subsequent querying. As an example, DBpedia contains at least 19 different IRIs for the property `birthplace` ranging from synonymous properties as `placeOfBirth` to French-named properties like `lieuDeNaissance`. Some of them are can be found in thousands of triples, whereas others are very rare, only being used in a couple of triples. But all of them hamper applications working with the data and may lead to incorrect and incomplete query results.

These *synonyms* may either be prevented by controlled vocabularies or strict manual supervision mechanisms as for example seen in Wikidata, or by data cleaning methods that are able to automatically identify synonyms from the data in an efficient way. Previous work has shown that property synonyms can automatically be identified by either frequent item set mining-based techniques [1] or by knowledge graph embedding-based techniques as in one of our previous works [14]. Whereas frequent item set mining lacks in precision, embedding-based techniques usually show a high quality, but are not interpretable. Furthermore, knowledge graph embeddings have been shown to have difficulties in correctly representing closely related properties: most embeddings for example will identify `north` and `south` as synonymous. Also the lack of interpretability is problematic, when the approach is used in a semi-automatic manner to support manual data cleaning.

In this work, we present an interpretable and scalable method for data-driven synonym detection in large-scale knowledge graphs that mines equivalent property definitions using rule mining techniques. We have developed a procedure that mines logical rules in the form of $\texttt{birthplace}(x, y) \Leftrightarrow \texttt{placeOfBirth}(x, y)$ indirectly, such that the rule does not need to be directly supported by triples. That means `birthplace` and `placeOfBirth` do not need to occur for the same entity pairs, but the respective represented concepts need to have a shared definition. Our mining technique is thus able to find synonyms that occur in thousands of triples as well as very rare synonyms with quite high quality. In fact, our synonym detection quality outperforms existing embedding-based techniques, while offering good explainability. For every synonym pair that has been found, our method provides similarities and dissimilarities of the properties in the form of logical rules.

The contributions of this work can be shortly summarized as follows:

- We develop a novel technique for synonym detection based on rule mining, finding and matching property definitions in a data-driven fashion.
- We perform extensive experiments on Wikidata and DBpedia outperforming state-of-the-art techniques for synonym detection, while offering explainable results in the form of shared definitions.
- For reproducibility, we provide all our source code, datasets, and results in a publicly available Github repository [1].

[1] `https://github.com/JanKalo/RuleAlign`

2 Related Work

Synonym Detection in Knowledge Graphs So far there is only little research on detecting synonyms in knowledge graphs or RDF data. An early work, on synonymous predicates for query expansion uses frequent item set mining [1]. Given a knowledge graph, for each property, they mine frequent item sets, consisting of object entities. Properties with high overlap with regard to their objects, but low overlap in their subjects are identified as synonym. However, another work has shown that synonyms often cannot be identified by this approach, because they have no overlap in their extension [14].

To tackle this problem, in a previous work, we have proposed a technique based on knowledge graph embeddings [14]. We trained knowledge graph embeddings and compute similarities between properties and use an outlier detection to separate synonyms from only similar properties. This approach is highly dependent on the quality of the embeddings, which varies massively from property to property, from knowledge graph to knowledge graph and from embedding model to embedding model. Furthermore, the results are not interpretable and therefore it is hardly foreseeable for properties, whether synonym detection works well and where it does not. To overcome these drawbacks, we have developed a technique going back to using a symbolic approach based on explicit feature representations in the form of logical rules.

Both approaches, frequent item set mining [1] and knowledge graph embeddings [14] are evaluated and compared to our technique in the experimental section of the paper.

Ontology Matching is about identifying corresponding concepts in two (or more) ontologies or knowledge graphs. Particularly the Ontology Alignment Evaluation Initiative (OAEI) at the Ontology Matching Workshop co-located with the International Semantic Web Conference plays an important role in advancing ontology matching research [2]. Ontology matching systems are primarily concerned with matching corresponding entities and classes from two or more distinct RDF datasets [13,10,12]. Some systems are also capable of matching properties [21,9]. Techniques often heavily rely on string metrics between URLs and labels, but also on structural graph measures.

In contrast to synonym detection, ontology matching systems usually can only align two distinct knowledge graphs and heavily rely on some existing correspondences between these two [21]. Finding duplicate information (e.g. property synonyms) within a single knowledge graph is therefore often not possible. Furthermore, several techniques are relying on manually built ontologies in OWL [13,12]. Heterogeneous real-world knowledge graphs however, often do not provide high quality ontological information.

Open Knowledge Graph Canonicalization Building knowledge graphs from text is a well researched topic in the natural language processing community. One approach is to rely on open information extraction techniques that extract triples

[2] `http://om2019.ontologymatching.org/`

directly from text, without sticking to some fixed vocabulary provided by an ontology or knowledge graph [17]. However, this often leads to heterogeneity issues like duplicate entities and paraphrased properties. Cleaning the extracted entity and property mentions from text is known under the term knowledge graph canonicalization [6,24]. In [6], synonym property mentions from text are identified by equivalence rules among these properties. But in contrast to our approach, their technique only mines rules that are supported by the data. This indeed works well for canonicalization where triple stem directly from text, but not for synonym detection in existing knowledge graphs. Also CESI [24] mines this kind of synonym rules as side information. Their main method however uses knowledge graph embeddings on the textual mentions of properties to canonicalize them. A comparable technique has been explored in our previous work for synonym detection in knowledge graphs and is evaluated in our experimental section [14].

Relational Learning Representing properties in some feature space is an important topic in the relational learning domain. Relational learning in general is about machine learning from relational data. In context of knowledge graphs, relational learning techniques are usually used for knowledge base completion, i.e. predicting triples from existing knowledge [18].

Recent works in knowledge graph completion rely on so called knowledge graph embeddings [19,23,4]. Entities and properties are represented as vectors/-matrices satisfying mathematical expressions given by a model. They are usually used to predict new triples. Furthermore, the semantic similarity between properties may be measured by computing similarity metrics between these embeddings. As previously mentioned, these embeddings may be used for synonym detection, but have some problems [14].

Other techniques for knowledge graph completion rely on symbolic representations, usually logical rules or graph features [8,15]. Here, it has been shown that logical rules, in particular Horn rules, can compete with embedding based techniques for knowledge graph completion. In this work, we analyze whether these logical representations of properties are also well suited for detecting synonyms.

3 Preliminaries

Without limiting the generality of our approach, we assume an arbitrary knowledge graph (KG) to be represented in the Resource Description Framework (RDF) [20]. Thus, a KG consists of a set of facts being subject, predicate, object *triples*: $(s, p, o) \in E \times R \times (E \cup L)$. A subject is an entity or concept from E, a predicate from a universe of properties R and an object is either an entity (i. e., from E) or a literal value from L. Although entities and predicates are technically represented through IRIs, we use suggestive identifiers for the sake of readability.

Our notation of logical rules over KGs stems from Galárraga et al. [8].

An *atom* for the triple (s, p, o) is written as $p(s, o)$. Beyond the RDF format for subject s and object o, rule atoms allow for variables $x, y, z, z_1, z_2, \ldots$ from

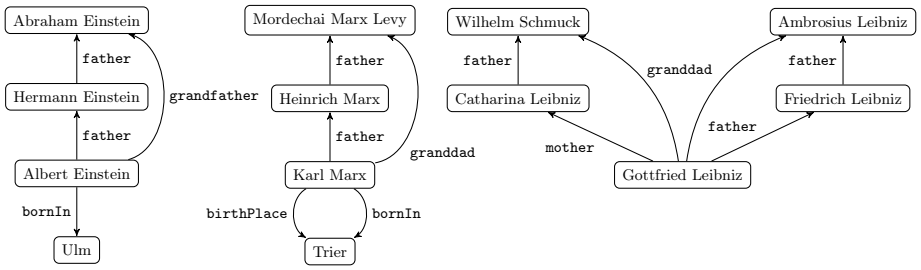

Fig. 1: An example knowledge graph about persons and their ancestors. Nodes are entities and edges are relationships.

a universe of variables V. A rule is a logical implication from a *body* term to a *head* term, where the body is a conjunction of multiple atoms b_i ($i \in \{1, ..., k\}$) while the head is a single atom: $b_1 \wedge ... \wedge b_k \Rightarrow r(s, o)$. Throughout this paper, our rules follow strictly this format, i. e., they are *Horn rules*. A rule in which every variable occurs at least twice is a *closed rule*. As an example, consider the following rule:

$$\texttt{father}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{granddad}(x,z) \tag{1}$$

The meaning of such a rule w. r. t. a KG is whatever matches the body of the rule (i. e., an assignment of actual KG subjects/objects to the variables) also matches the head. Regarding (1), if y is the *father of* x and z is the *father of* y, then z is the *granddad of* x. We can use this closed rule to predict new facts or to justify existing ones in a KG, like the one depicted in Fig. 1. The KG delivers the facts `father`(Karl M., Heinreich M.) and `father`(Heinrich M., Mordechai M.L.), which implies the fact `granddad`(Karl M., Mordechai M.L.) according to (1). Similarly, we can infer the granddad property between Gottfried Leibniz and Ambrosius Leibniz.

Since rules are usually mined for prediction purposes from an incomplete real-world KG, we need a way to assess their quality. Galárraga et al. use two measures for assessing the quality of a rule $B \Rightarrow r(x, y)$ [8]. First, the *support* of a rule is the absolute number of instances the rule is correct in the KG:

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

where $z_1, \ldots, z_n$ are the variables occurring in B distinct from x and y. Thus, the support quantifies the number of predictions that are already instances of the KG (*true positives*). For our example rule (1) and KG in Fig. 1, the support is 2 because the rule can only be instantiated by Karl Marx, Gottfried Leibniz and their ancestors. We count `granddad`(Karl M., Mordechai M. L.) and `granddad`(Gottfried L., Ambrosius L.) exactly once. The absolute support value is difficult to interpret if the frequency of a property in the KG and the size of the KG itself is unknown [8]. A support of 1 has a totally different meaning if there are thousands of properties in the KG, as opposed to only a single one

in our example. For $\texttt{father}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{grandfather}(x,z)$, the support is only 1 but `grandfather` also only occurs once in the KG.

To become independent of the size of the KG and the frequency of property occurrences, the *head coverage* was introduced as a relative support. It measures the support of a rule relative to the number of occurrences of the respective head relation in the given KG:

$$hc(B \Rightarrow r(x,y)) = \frac{supp(B \Rightarrow r(x,y))}{\#(x',y') : r(x',y')} \tag{3}$$

The head coverage for (1) is thus 0.66 because its support is 2 and the granddad relation occurs three times in Fig. 1.

Since rules are usually mined from the data, we need a second measure assessing the prediction quality of a rule by means of its *standard confidence*:

$$conf(B \Rightarrow r(x,y)) = \frac{supp(B \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, \ldots, z_n : B} \tag{4}$$

The number of true positives relative to the number of all predictions (due to the rule) shows us how many of the predictions are part of the current knowledge graph. Hence, a high confidence entails that the rule is justified by the data in the KG. Regarding (1), the confidence w. r. t. Fig. 1 is 0.66 because the rule's body matches three times while the whole rule comes with a support of 2.

We restrict ourselves to closed Horn rules because this allows the mining process to finish in reasonable time [8].

4 Rule Mining for Synonym Detection

From an RDF point of view, two distinct properties refer to two distinct concepts [20], as described by distinct resources. However, as KGs grow at an enormous pace, extraction and/or human error bring forth properties, such as `birthPlace`, `born`, or `placeOfBirth`, which refer to the same real-world concept. Therefore, we qualify such properties as *synonymous*. Even at this informal stage, *synonymity* is recognized as an equivalence relation. Hence, if two or more properties $r_1, \ldots, r_m \in R$ are synonymous, they can be united to a single URI.

This section is devoted to characterizing synonymous properties in a way that enables us to use existing rule mining techniques, e. g., AMIE+ [7], to identify them as equivalence rules

$$r_1(x,y) \Leftrightarrow r_2(x,y), \tag{5}$$

i. e., r_1 may be replaced by r_2 and vice versa. For properties $r_1, r_2 \in R$, we call (5) a *synonym rule*. An obvious *rule mining*-based solution tries to find two rules $r_1(x,y) \Rightarrow r_2(x,y)$ and $r_2(x,y) \Rightarrow r_1(x,y)$ which culminates to synonymity of r_1 and r_2. Confidence values greater than 0 would require r_1 and r_2 to co-occur for the same subject-object pairs: In Fig. 1, rules $\texttt{bornIn}(x,y) \Rightarrow \texttt{birthPlace}(x,y)$

and $\texttt{birthPlace}(x,y) \Rightarrow \texttt{bornIn}(x,y)$ may be inferred with confidence values of 0.5 and 1.0. Since both rules have high confidence values, we could take them as being correct and therefore infer the synonym rule $\texttt{bornIn}(x,y) \Leftrightarrow \texttt{birthPlace}(x,y)$. Mining this kind of synonym rules is the classical approach to detect synonyms using rule mining [6,24].

However, the just stated scenario is quite artificial: In real-world KGs, synonyms often stem from integrated triples from multiple sources, e. g., different extraction tools or persons. Often these triples have totally different domains and share no entities at all. In such cases, rule mining solely relying on the data instances is not very helpful. As another example, we observe that we have no support for the rule $\texttt{grandfather}(x,y) \Rightarrow \texttt{granddad}(x,y)$ in our example. They simply occur for totally different entities, although a unification of both properties is appropriate.

4.1 Mining Property Definitions

Instead, we try to indirectly mine synonym rules by first mining *property definitions*. Intuitively speaking, a definition is a paraphrase of a property through other properties. Thus, it is an equivalent logical formula to some property. In case of `granddad`, we may find

$$\begin{aligned}\texttt{granddad}(x,z) \Leftrightarrow &(\texttt{father}(x,y) \wedge \texttt{father}(y,z)) \vee \\ &(\texttt{mother}(x,y) \wedge \texttt{father}(y,z))\end{aligned} \tag{6}$$

an appropriate definition. We identify synonymous properties r_1 and r_2 indirectly by mining their property definitions. More formally, we mine property definitions D such that

$$r_1(x,y) \Leftrightarrow D \Leftrightarrow r_2(x,y),$$

which lets us conclude (5) by transitivity of logical equivalence.

Since state-of-the-art rule induction systems usually are only able to mine Horn rules, due to performance reasons, we adapt our notion of property definitions accordingly. Applying a standard rule mining system on the KG from Fig. 1, using the `granddad` relation as a head relation, we mine two rules culminating to the definition given in (6): (a) the paternal granddad

$$\texttt{father}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{granddad}(x,z)$$

but also (b) the maternal granddad

$$\texttt{mother}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{granddad}(x,z).$$

The confidence of (a) is 0.66 and its head coverage is also 0.66. The confidence of rule (b) is 1.0, but the head coverage therefore is only 0.33. Both rules pretty much cover what the `granddad` property expresses. The hypothetical rule

$$\begin{aligned}\texttt{granddad}(x,z) \Leftarrow &(\texttt{father}(x,y) \wedge \texttt{father}(y,z)) \vee \\ &(\texttt{mother}(x,y) \wedge \texttt{father}(y,z))\end{aligned}$$

has a head coverage of 1.0 and a confidence 0.75. We observe that indeed the disjunction of the rule bodies of the mined Horn rules exceeds the head coverage values of the single rules. The higher the head coverage, the more likely it is to observe the body (or one of the bodies) whenever the head is matched. The extreme case of a head coverage of 1.0 means that whenever the head property is observed, the body can also be matched. In our example, the head coverage is even the sum of the head coverages of both rules because the bodies cover totally different entities. More generally, however, instantiations of different Horn clauses in a definition might overlap, which needs to be considered for head coverage computation by counting distinct instances.

Our example already suggests that the combined rule (6) is valid. This observation can be justified by the rule's head coverage and standard confidence. In general, confidence and head coverage have the rule support in the numerator. While confidence considers the number of matches of the body in the denominator, head coverage uses the size of the head relation. From this, we obtain

$$conf(B \Rightarrow r(x,y)) = hc(B \Leftarrow r(x,y)). \tag{7}$$

Thus, a rule having a high standard confidence and a high head coverage may imply that rule body and rule head are equivalent.

Driven by the interpretations and observations above, a *property definition* for $r \in R$ is a disjunction of Horn clauses, i. e., $D = b_1 \vee ... \vee b_k$, such that the rule $D \Leftrightarrow r$ holds. In the best case, confidence and head coverage of a definition are as close to 1.0 as possible. Note that a head coverage and confidence of 1.0 is only possible if the property and its definition share all their entities, which is rarely the case in KGs. If directly synonymous properties, sharing several entities, exist in the KG, this yields synonym rules with high confidence and high head coverage, being part of the respective definitions. In most cases, mining Horn rules on real-world KGs yields high confidence but a large number of rules with low head coverage values. Hence, a definition usually consists of a disjunction of hundreds of Horn clauses, covering a very diverse set of entities, and therefore achieving a high overall head coverage for the definition.

4.2 Mining Synonym Rules by Matching Definitions

In heterogeneous and large-scale knowledge graphs, only very few identical definitions can be found: Reconsidering the mined example definition and trying to find properties, such as `grandfather`, with an equal definition will almost surely fail. The mining process for `grandfather` returns a single Horn rule:

$$\texttt{grandfather}(x,z) \Leftarrow \texttt{father}(x,y) \wedge \texttt{father}(y,z) \tag{8}$$

This rule even has a head coverage of 1.0 and a confidence of 0.33. Due to the high head coverage and confidence, it follows that the body is a definition for grandfather (w. r. t. to the KG in Fig. 1). The mined definitions for `grandfather` and `granddad` are different but share the clause $\texttt{father}(x,y) \wedge \texttt{father}(y,z)$. This is a typical situation for real-world definitions that have been created in a

purely data-driven fashion. To overcome this mismatch of definitions, we relax our indirect mining approach, such that also only partial matches can be used to find synonymous properties. For our example, this would imply that we find the following indirect synonym rule:

$$\begin{aligned} & \texttt{granddad}(x,z) \\ \Leftrightarrow & \texttt{father}(x,y) \wedge \texttt{father}(y,z) \\ \Leftrightarrow & \texttt{grandfather}(x,z) \end{aligned} \tag{9}$$

Since this rule leaves out parts of the definition of `granddad`, we obtain a lower head coverage for this definition, which negatively influences the definition's quality. Since the matched definition only covers a restricted proportion of the entities that are taking part in the `granddad` relation, also the quality of the synonym rule may be affected negatively. Therefore, in our mining process, we aim at maximizing the overlap of the definitions of two properties, in order to classify them as synonymous. Here, the Jaccard coefficient of the definitions $\frac{D_1 \cap D_2}{D_1 \cup D_2}$ determines the quality of the overlap. Bodies from the definitions are thereby identical if they are structurally identical (isomorphic), respecting the head properties' direction. As a result, we obtain a Jaccard coefficient between 0 and 1 for each property pair which can be interpreted as a confidence for the indirect rule mining. In our granddad and grandfather example above, the Jaccard coefficient is $\frac{1}{2} = 0.5$.

The overall matching process consists of two steps: (1) First of all, we start a rule mining process on a knowledge graph to obtain definitions for all properties. (2) A comparison of all definitions for all property pairs is performed to compute respective Jaccard coefficients. As a result, a ranked list of property pairs with confidence values is returned. If no definition could be mined for some property, all its confidence values are automatically set to 0.0 since no matching definition can be found.

5 Evaluation

In our experiments, we evaluate our rule-based technique against a frequent item set-based technique [1] and our previously published approach based on knowledge graph embeddings [14] on two large real-world knowledge graphs. Our implementation, a description on how to reproduce the experiments and the datasets are all available through our Github repository [3].

For all experiments we employ an existing tool for mining Horn rules: we use AMIE+ [7] with a minimum head coverage of 0.005, a minimum confidence of 0.05 and a minimum initial support to mine closed and connected Horn rules on the datasets. If the rule mining algorithm did not output new rules for more than 10 hours, we preliminary stopped the mining process and used the rules mined so far.

[3] `https://github.com/JanKalo/RuleAlign`

Overall, two experiments using 7 baseline approaches are performed: (1) To assess, whether the quality of synonym detection methods is ready for cleaning real-world knowledge graphs, we perform a manual evaluation of the systems quality on DBpedia. (2) In the other experiment, we want to analyze the recall and precision of synonym detection techniques on synthetically created synonyms in Wikidata.

Overall, we compare the approaches on two very large real-world datasets Wikidata and DBpedia. Since both datasets have several hundred millions of triples which is unfeasible for training knowledge graph embeddings as well as for mining rules in a feasible time, we stick to the sampled datasets that have been built in [14]. This also allows for a better comparison of our results to previous works. In their work, the authors have presented a sampling technique that keeps triples with every existing property in the respective knowledge graph, while reducing the overall number of triples. Our gold standard datasets containing our manually labeled synonyms for DBpedia and the synthetic synonyms for Wikidata are available online [4].

Frequent Item Set Baseline The approach presented in [1] uses frequent item set mining to detect synonymously used predicates to perform query expansion. In this work, we used the implementation and results of this baseline from [14]. In that work, we re-implemented the approach using *Range Content Filtering* and *Reversed Correlation Coefficient* as described in the original paper using Python and Spark. The implementation of the approach is also openly available on Github. As an input parameter for frequent item set mining, the approach requires the user to provide a minimum support value. For both experiments, a grid search optimizing for optimal F1-measures was performed.

Knowledge Graph Embedding Baselines In our previous work [14], it was shown that knowledge graph embeddings may be used to detect synonymous properties, by using outlier detection techniques on the property representation in state-of-the-art embeddings. In the original paper 8 different embedding techniques have been presented using L1 metric as well as cosine similarity. For this work, we only take the top 6 embeddings with the metrics that worked best: TransH [26], TransD [11] ComplEx [23], DistMult [27], ANALOGY [16] and HolE [19]. All these techniques achieve very high quality in the top results, the recall however is problematic in some of the presented experiments. We will further analyze the differences of the fundamentally different approaches embeddings vs. logical rules in various settings here.

5.1 Manual Quality Evaluation in DBpedia

The DBpedia sample comprises 12 million triples with around 15,000 different properties with several natural synonyms, ranging from very rare synonyms only

[4] `https://doi.org/10.6084/m9.figshare.11343785.v1`

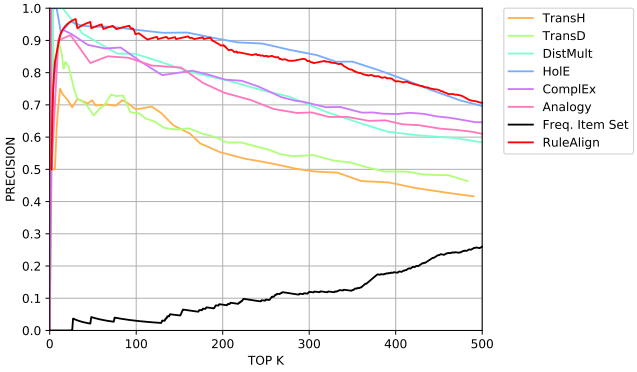

Fig. 2: Experimental results from our approach RuleAlign in red to several baselines on DBpedia manually evaluated with precision at k up to $k = 500$.

occurring in around 100 triples up to synonyms being part of hundreds of thousands triples. The evaluation on DBpedia is performed manually for the top 500 results of each of the approach classifying pairs of properties in either being synonyms or not. For the base line approaches, we rely on the datasets classified in [14] extended by a manual classification performed for our newly proposed approach.

In this experiment, we have performed a manual evaluation for the precision@k up to $k = 500$ on a DBpedia sample comparing 8 different approaches. The results are presented as line graphs in Figure 2.

The frequent item set-based baseline has an increasing precision for higher k values, due to a ranking function that assumes that synonymous properties are not occurring for similar subject entities. This assumption is not true for DBpedia. The precision for this baseline always is below 30% and also does not exceed 30% for k values above 500. The best embedding-based baseline is HolE, having a maximum precision of over 90% in the top 200 results and a precision around 70% at $k = 500$.

Our approach, presented as RuleAlign in red, shows the best results in this experiments together with the embedding model HolE finding at least 352 correct synonyms. Overall, the number should go into the thousands when we extended our manual evaluation. In comparison to a direct rule mining approach for equivalence rules, our indirect approach finds at least 77 correct synonym pairs on our DBpedia dataset which cannot be found by the other approach because they have no support.

But as an additional feature, our approach is able to propose explanations for the synonym predictions in form of property definitions. The top explanations are having a high head coverage, covering lots of entities and have a high confidence. In Table 1, we present some example definitions from DBpedia. Since for many properties around 100 Horn clauses are in the definition, we only present top matched Horn clauses. These explanations are very natural definitions of

Table 1: Matched property definitions mined from DBpedia as an explanation for the result.

Property	Definition
`grandsire`(x,z)	`sire`(x,y) ∧ `sire`(y,z)
`nationality`(x,y)	`stateOfOrigin`(x,y)
`nationality`(x,z)	`birthPlace`(x,y) ∧ `country`(y,z)
`north`(x,y)	`east`(y,z) ∧ `northeast`(x,z)

the respective properties that would also be used in the real-world. Note, that besides these human readable example definitions, many synonym pairs are entirely different in their respective URI labels, e.g. "dbp:ff" (father of the father) and "dbp:grandsire" and are therefore very difficult to be identified by humans without our automatic data-driven approach.

A closer look at our predictions reveal some shortcomings of our approach. First of all, our approach is not able to distinguish the gender within some properties. We classify for example `father` and `mother` as synonyms, because no rule is able to capture the gender correctly. One reason for that is, that the gender is only mentioned as a literal, which is ignored by the rule mining approach. A second problem are properties that hardly can be distinguished by their data instances, because they are extremely similar. As an example `firstDriver` and `secondDriver` representing a person's placement in a race, cannot be distinguished. Furthermore, false-positives in the form of hyponyms as for example `genre` and `musicGenre` are returned.

5.2 Precision-Recall Evaluation in Wikidata

The Wikidata sample has more than 11 million triples and more than 1,500 properties. In contrast to DBpedia, it is supposed to be free of synonyms due to intensive manual curation. Therefore, in ([14]) have introduced synthetic synonyms here by randomly re-naming existing properties. For the triple (Albert E., father, Hermann E.) we instead use (Albert E., father_synonym, Hermann E.). Thus, the properties `father` and `father_synonym` can be treated as synonyms, but never co-occur for the same subject-object pair. Overall, 343 synonymous properties have been introduced that need to be identified for the approach. A more detailed description on the creation of the dataset can be found in the original paper .

We again start in having a look at the frequent item set baseline in black. It starts with a very high precision for very low recall values and then drops sharply to under 20%. The maximum precision is at 21% at a recall value of around 35%. Due to the minimum support value that lead to best F1-measure, no higher recall value is achieved here. Embedding-based approaches achieve a very high precision up to a recall of 30%. The best approach is again HolE, starting at 90% precision for a recall of 10% and a precision of 10% for 70%

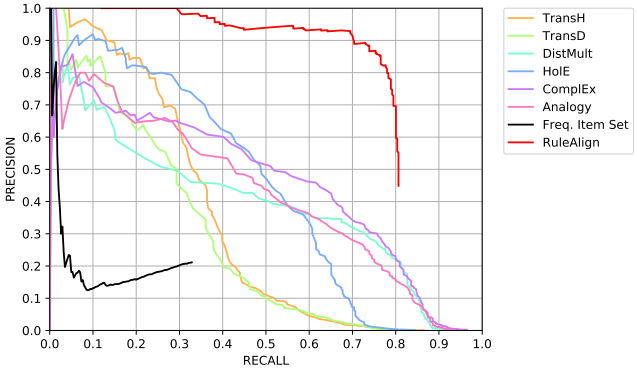

Fig. 3: Experimental results from our approach RuleAlign on Wikidata. We provide a precision-recall analysis for synthetic synonyms.

recall. In contrast, our approach (red) is having a perfect precision for recall values up to 30% and still a precision over 90% for a recall of 70%. The recall of our approach sharply drops never achieving 80% recall.

The second experiment measures precision and recall for 343 synonyms in a Wikidata sample. Our results regarding this experiment are presented as precision-recall curves in Figure 3. For Wikidata, our approach achieves an extremely high precision, but also has problems in recall due to two reasons: (1) For 32 properties, no rule could be mined due to the minimum head coverage in the rule mining process. (2) The other synonyms could not be found, since none of the mined rules fulfilled our minimum confidence threshold. The few false positives that have been returned by our approach often were hyponyms instead of synonyms.

5.3 Discussion

The rule-based approach matching data-driven property definitions for detecting synonymous properties achieves very high precision. In both datasets, we could observe that a high Jaccard coefficient often implies that the respective property pair is synonymous. In the Wikidata experiment, all pairs with a confidence above 0.9 are synonyms and also in DBpedia a high confidence leads to good results.

However, in DBpedia only very few synonyms with high confidence could be found. For lower Jaccard coefficients, a higher proportion of false positives is returned, because these properties often were in a hyponym relation. These could be solved by an improved matching process that also takes into account the head coverages of the rules when computing the Jaccard coefficient. However, this might further decrease the recall of our approach, which has already been observed as a problem for the Wikidata dataset. The simple Jaccard coefficient as used in this work, achieves very high precision with a reasonable recall.

A low recall could also be prevented by mining rules with lower head coverage, mining more expressive rules or by decreasing the minimum confidence threshold. In turn, this might further decrease the performance of the rule mining tool, resulting in enormous rule sets.

Several false positives that were returned in DBpedia, had a high overlap in their data instances and therefore also very similar definitions. These properties were very similar, but from the labels or IRIs, we observed that they were not synonym. These cases can hardly be identified in a data-driven fashion, because they often need detailed domain knowledge.

6 Conclusion

We have presented a novel approach adapting classical rule mining for knowledge graphs to detect synonymous properties in a data-driven way using property definitions. In two large-scale experiments on two real-world knowledge graphs, we have shown that our approach is able to identify a large proportion of existing synonyms with a precision of over 80% without making any assumptions on the data. In contrast to existing work in this area, our approach is providing human understandable explanations of its decisions in the form of logical Horn clauses, while achieving a higher precision in existing benchmark datasets.

This work shows that symbolic approaches, like rule mining in our case, are capable of competing with latent approaches (i. e., knowledge graph embeddings), when it comes to identifying synonymous properties. In particular with regard to precision and interpretability our rule-based approach is superior over existing systems. However, as shown in our evaluation, our rule-based approach is stretching a purely data-driven approach to its limits. Most false positives that have been produced by our system, cannot be detected purely automatically, because it cannot be observed from the triples nor the property label. Here, it seems promising to have a semi-automatic approach with humans manually checking the matched definitions.

With regard to scalability, however, both paradigms seem to have problems when it comes to real-world knowledge graphs. Knowledge graph embedding training needs powerful GPUs which currently are very restricted with regard to their memory, preventing the training for large datasets. Rule mining, on the other hand, requires the computation of huge joins for a possibly exponential number of rules. These joins sometimes comprise hundred thousands triples which already takes several minutes for a single rule candidate on state-of-the-art hardware, when working with large datasets like Wikidata.

As a future work, we plan to extend the approach to also detect hyponyms between properties and inverse properties. More importantly, we would like to use more expressive rules instead of just closed Horn clauses to improve precision even more. So far, existing rule mining approaches have major performance issues for these kinds of rules on larger datasets.

References

1. Abedjan, Z., Naumann, F.: Synonym analysis for predicate expansion. In: The Semantic Web: Semantics and Big Data. pp. 140–154. ESWC '13 (2013)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: Proc. of the 6th International Semantic Web Conf. pp. 722–735. ISWC '07 (2007)
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A collaboratively created graph database for structuring human knowledge. In: Proc. of the 2008 ACM SIGMOD Int. Conf. on Management of Data. pp. 1247–1250. SIGMOD '08 (2008)
4. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
5. Dong, X., 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 the 20th Int. Conf. on Knowledge Discovery and Data Mining. pp. 601–610. SIGKDD '14 (2014)
6. Galárraga, L., Heitz, G., Murphy, K., Suchanek, F.M.: Canonicalizing open knowledge bases. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 1679–1688. CIKM '14 (2014)
7. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. The VLDB Journal **24**(6), 707–730 (12 2015)
8. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.: Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In: Proceedings of the 22Nd International Conference on World Wide Web. pp. 413–422. WWW '13 (2013)
9. Hertling, S., Paulheim, H.: Dome results for oaei 2018. In: OM 2018 : Proc. of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conf. (ISWC 2018) Monterey, CA, USA, October 8, 2018. vol. 2288, pp. 144–151 (2018)
10. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology alignment for linked open data. In: Proc. of the 9th International Semantic Web Conf. on The Semantic Web - Volume Part I. pp. 402–417. ISWC '10 (2010)
11. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge Graph Embedding via Dynamic Mapping Matrix. In: Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conf. on Natural Language Processing. pp. 687–696. ACL '15 (2015)
12. Jiménez-Ruiz, E., Cuenca Grau, B.: Logmap: Logic-based and scalable ontology matching. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) The Semantic Web – ISWC 2011. pp. 273–288. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
13. Juanzi Li, J., Jie Tang, J., Yi Li, Y., Qiong Luo, Q.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Transactions on Knowledge and Data Engineering **21**(8), 1218–1232 (aug 2009)
14. Kalo, J., Ehler, P., Balke, W.: Knowledge graph consolidation by unifying synonymous relationships. In: The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand. pp. 276–292 (2019). https://doi.org/10.1007/978-3-030-30793-6_16

15. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 529–539. EMNLP '11 (2011)
16. Liu, H., Wu, Y., Yang, Y.: Analogical inference for multi-relational embeddings. In: Proc. of the 34th Int. Conf. on Machine Learning. pp. 2168–2178. ICML '17 (2017)
17. Mausam, Schmitz, M., Bart, R., Soderland, S., Etzioni, O.: Open language learning for information extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 523–534. EMNLP-CoNLL '12 (2012)
18. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A Review of Relational Machine Learning for Knowledge Graphs. Proc. of the IEEE **104**(1), 11–33 (1 2016)
19. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proc. of the 30. AAAI Conf. on Artificial Intelligence. pp. 1955–1961. AAAI'16, AAAI Press (2016)
20. Schreiber, A., Raimond, Y.: RDF 1.1 Primer. W3C Working Group Note, World-Wide Web Consortium (2014), `https://www.w3.org/TR/rdf11-primer/`
21. Suchanek, F.M., Abiteboul, S., Senellart, P.: Paris: Probabilistic alignment of relations, instances, and schema. Proc. of the VLDB Endowement **5**(3), 157–168 (Nov 2011)
22. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proc. of the 16th Int. Conf. on World Wide Web. p. 697. WWW '07 (2007)
23. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: Proc. of the 33rd Int. Conf. on Int. Conf. on Machine Learning - Volume 48. pp. 2071–2080. ICML'16 (2016)
24. Vashishth, S., Jain, P., Talukdar, P.: Cesi: Canonicalizing open knowledge bases using embeddings and side information. In: Proceedings of the 2018 World Wide Web Conference. pp. 1317–1327. WWW '18 (2018). https://doi.org/10.1145/3178876.3186030
25. Vrandečić, D., Denny: Wikidata: A New Platform for Collaborative Data Collection. In: Proc. of the 21st Int. Conf. companion on World Wide Web. p. 1063. WWW '12 Companion (2012)
26. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proc. of the Twenty-Eighth AAAI Conf. on Artificial Intelligence. pp. 1112–1119. AAAI'14 (2014)
27. Yang, Q., Wooldridge, M.J., Codocedo, V., Napoli, A.: Twenty-Fourth International Joint Conf. on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, 25-31 July 2015 (2015)