Three Approaches for Mining Definitions from Relational Data in the Web of Data

Justine Reynaud, Yannick Toussaint, Amedeo Napoli

Three Approaches for Mining Definitions from Relational Data in the Web of Data

Justine Reynaud, Yannick Toussaint, and Amedeo Napoli

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
firstname.lastname@loria.fr

Abstract. In this paper we study a classification process on relational data that can be applied to the web of data. We start with a set of objects and relations between objects, and extensional classes of objects. We then study how to provide a definition to classes, i.e. to build an intensional description of the class, w.r.t. the relations involving class objects. To this end, we propose three different approaches based on Formal Concept Analysis (FCA), redescription mining and Minimum Description Length (MDL). Relying on some experiments on RDF data from DBpedia, where objects correspond to resources, relations to predicates and classes to categories, we compare the capabilities and the complementarity of the three approaches. This research work is a contribution to understanding the connections existing between FCA and other data mining formalisms which are gaining importance in knowledge discovery, namely redescription mining and MDL.

Keywords: Relational data, Formal Concept Analysis, Redescription mining, DBpedia.

1 Introduction

In this work, we are interested in checking the completeness and the quality of RDF data in the linked open data (LOD), and the potential to discover definitions from these sets of linked data. Such definitions can be reused in the design of Knowledge Bases (KBs). This challenge is of main importance when we consider the masses of data which are currently published in LOD.

At an abstract level, we can view the current problem as follows. We have at hand a set of interconnected objects –objects connected by relations– just as an ABox in a description logics (DL) framework [2], and the objective is to classify the objects with respect to and in compliance with the connections they are involved in. Objects are classified in the same class, actually an extension, as soon as they share common elements. This sharing can be strict –elements are the same– or soft –elements are similar. Finally, we obtain a set of classes, possibly partially ordered, and their associated descriptions. These descriptions are important if not mandatory as they are a basis for building the definitions of classes. Definitions are considered as sets of necessary (NC) and sufficient conditions (SC) used for classifying new objects. If `x` is an instance of class `Red`

then x has color red (NC), and conversely, if x has color red then x is an instance of class Red (SC).

Continuing the analogy with DLs, the idea in this paper is to build and apply induction rules having the form: $\mathtt{r}(\mathtt{x},\mathtt{y})$ and $\mathtt{y} : \mathtt{C}$ then $\mathtt{x} : \exists \mathtt{r}.\mathtt{C}$. This means that given a relation such as $\mathtt{r}(\mathtt{x},\mathtt{y})$ between objects x and y, with y instance of class C, then we infer that x is an instance of a class say D whose description includes the expression $\exists \mathtt{r}.\mathtt{C}$, i.e. instances of D are related to instance(s) of C.

In this work, we aim to build definitions from RDF data. To this end, we use three approaches including Formal Concept Analysis (FCA), redescription mining and translation rule discovery. Then the main operations that we should perform are (i) the preparation of the data, (ii) the discovery of definitions, (iii) the evaluation of the quality of definitions. To compare the three algorithms, we run experiments on data extracted from DBpedia. This paper is in continuation of a line of research work on the discovery of definitions within RDF triples in the linked open data. The originality in this paper is to compare three approaches which are not based on the same principles but which can complement each other. Moreover, to the best of our knowledge, this is one of the first papers where such a study and comparison is drawn at a theoretical and practical level.

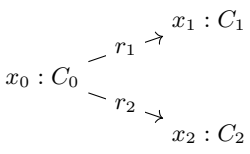
The paper is organized as follows. In the second section, we present the data on which we will be working and the basis of the classification process in the linked open data. The third section details the three classification approaches and their application. The following section is related to the experiments which have been carried out for evaluating the three approaches. Finally, a discussion, related and future work conclude the paper.

2 Data representation

In this section, we present basics of linked open data, and how we represent RDF triples as a formal context.

2.1 Linked Open Data

Linked open data (LOD) are relational data that can be seen as a set of interconnected *knowledge bases* (KB). A KB relies on two main components, a TBox which defines the *schema* of the KB and includes the concept definitions and the ABox which introduces individuals and the expressions in which individuals are involved. The basic units in a KB are RDF triples $\langle s, p, o\rangle$, which encode *subject–predicate–object* assertions. The elements of a triple can be a *resource* uniquely identified, a *literal* (values like strings, dates or integers) or a *blank node* (existential quantifier). For the sake of simplicity, in this paper we consider that $\langle s, p, o\rangle \in U \times U \times U$, where U is the set of all identified resources. Resources can refer to any object or abstraction and are identified by a URI (Uniform Resource Identifier). A URI is an address that is composed of two parts. The first part is the *namespace*, which indicates from which KB the resource comes from. The



(a) Relational data.

```
ex:x0 ex:r1 ex:x1 .
ex:x0 ex:r2 ex:x2 .
ex:x0 rdf:type ex:C0 .
ex:x1 rdf:type ex:C1 .
ex:x2 rdf:type ex:C2 .
```

(b) RDF data.

	M_C			M_D	
	C_0	C_1	C_2	$\exists r1{:}x_1$	$\exists r2{:}x_2$
x_0	×			×	×
x_1		×			
x_2			×		

(c) Formal context.

Fig. 1: Relational data, the associated set of RDF triples and the formal context built from RDF triples.

second part names the resource in this KB. The relation `rdf:type` is a specific relation of RDF which corresponds to the relation of instanciation.

LOD can be queried thanks to SPARQL queries. For example, the query `SELECT ?x WHERE {?x rdf:type ex:C0}` returns all the instances of C_0. Considering the example in Figure 1b, only $\texttt{ex} : \texttt{x}_0$ is returned.

2.2 Formal Concept Analysis and RDF data

We rely on Formal Concept Analysis (FCA) from [6] in order to compare the approaches. Given G a set of objects, M a set of attributes and $I \subseteq G \times M$ a binary relation between G and M, (G, M, I) is a formal context. Derivation operators (denoted $.'$) for a set of entities $X \subseteq G$ and a set of attributes $Y \subseteq M$ are $X' = \{m \in M \mid \forall x \in X, xIm\}$ and $Y' = \{g \in G \mid \forall y \in Y, gIy\}$.

From RDF data describing a KB, we build a formal context where G is the set of subjects of the triples (i.e. $G = \{s \mid \langle s, p, o\rangle \in KB\}$) and M is the set of pairs (predicate, object) that appear in the RDF data (i.e. $M = \{(p, o) \mid \langle s, p, o\rangle \in KB\}$). The incidence relation is defined as $sI(p, o) \Leftrightarrow \langle s, p, o\rangle \in KB$.

The set of attributes is a partition of two sets: $M = M_C \cup M_D$ and $M_C \cap M_D = \emptyset$. The set M_C is the set of all attributes (p, o) such that $\texttt{p} = \texttt{rdf:type}$. Since all the resources in the range of `rdf:type` are classes, M_C corresponds to the set of all the classes we are trying to define. Hereafter, an attribute $(rdf : type, C)$ will simply be denoted C. The set M_D is the set of all attributes (p, o) such that $\texttt{p} \neq \texttt{rdf:type}$. Hereafter, an attribute $(p, o) \in M_D$ will be referred as a *description* and denoted $\exists p : o$ where o is an abbreviation of an abstract class containing only o. Considering the example Figure 1b, the associated context is presented Figure 1c.

Our goal is to build definitions of classes of the form $\texttt{C} \equiv \texttt{e}_1 \sqcap \texttt{e}_2 \sqcap \ldots \sqcap \texttt{e}_\texttt{n}$, where the $\texttt{e}_\texttt{i}$ is an expression of the form $\exists \texttt{r.x}$. To this end, we are searching for two sets of attributes $C \subseteq M_C$ and $D \subseteq M_D$ such that their derivations are the same ($C' = D'$). For example, in Figure 1, we have $\{C_0\}' = x_0$ and $\{\exists r_1{:}x_1, \exists r_2{:}x_2\}' = x_0$. Thus, the definition $C_0 \equiv \exists r_1{:}x_1 \sqcap \exists r_2{:}x_2$ can be constructed. Since data may be incomplete, it is possible that there is no equality between the derivations of a class and the derivation of its description, i.e. $\{\texttt{C}_\texttt{i}\}' \neq \{\exists \texttt{r} : \texttt{x}_\texttt{j}\}'$. Therefore, we need to find some kind of approximation. This is allowed by the three algorithms presented in next section.

3 Rule mining algorithms

In this section, we briefly present the three approaches we are interested in, namely association rule mining, redescription mining and translation rule mining. Interested reader may refer to the original publications for further explanations.

3.1 Association rules

The goal of association rule mining [7] is to find dependencies between attributes. An association rule between two sets of attributes A and B, denoted $A \rightarrow B$ means that $A' \subseteq B'$. This rule has a *confidence* which can be considered as a conditional probability:

$$\text{conf}(A \rightarrow B) = \frac{|A' \cap B'|}{|A'|}$$

where $(.)'$ corresponds to the derivation operator. Confidence is used as a quality measure of the rule. An association rule is valid if its confidence is superior to a given threshold θ. When $conf(A \rightarrow B) = 1$, the rule is an implication, denoted by$A \Rightarrow B$. If $B \Rightarrow A$, then A and B form a definition, denoted by $A \equiv B$.

Since the confidence is not symmetric, $A \rightarrow B$ can be valid but $B \rightarrow A$ not valid. Potentially, an association rule $A \rightarrow B$ can be considered together with its reverse $B \rightarrow A$, and we can wonder how far they are from being implications. Accordingly, we introduce the notion of a quasi-definition which is to definition what association rule is to implication.

Definition 1 (Quasi-definition). *Given two sets of attributes A, B and a user-defined threshold θ, a quasi-definition $A \leftrightarrow B$ holds if $A \rightarrow B, B \rightarrow A$ and*

$$\min(\text{conf}(A \rightarrow B), \text{conf}(B \rightarrow A)) \geqslant \theta$$

The algorithm `Eclat` [11] is one of the existing algorithms for enumerating frequent itemsets. From frequent itemsets, association rules can be enumerated. Here, we use `Eclat` as implemented in the Coron system[1] for computing association rules. It exhaustively enumerates all the association rules that hold w.r.t. a given threshold. Here, we rely on `Eclat` to mine association rules.

Since we want to provide definitions of classes, we are interested in rules $X \rightarrow Y$ such that $X \subseteq M_C$ and $Y \subseteq M_D$ or, conversely, $X \subseteq M_D$ and $Y \subseteq M_C$. Given a rule R: $X \rightarrow Y$, the consequent can be decomposed into two rules R_C: $X \rightarrow Y_C$ and R_D: $X \rightarrow Y_D$ where $Y_C = Y \cap M_C$ and $Y_D = Y \cap M_D$ respectively. Since $Y_C \subseteq Y$, $Y' \subseteq Y'_C$, thus $|X' \cap Y'| \leqslant |X' \cap Y'_C|$, which means that if R holds, then R_C holds. Similarly, if R holds, then R_D holds.

We take advantage of this property to keep the quasi-definitions we are interested in. For example, $\exists r_1{:}x_1, C_0 \rightarrow \exists r_2{:}x_2$ is not kept because the antecedent include both categories and descriptions. On the other hand, $\exists r_1{:}x_1 \rightarrow$

[1] http://coron.loria.fr/

$\{\exists r_2{:}x_2, C_0\}$ can be decomposed into R_1: $\exists r_1{:}x_1 \to \exists r_2{:}x_2$ and R_2: $\exists r_1{:}x_1 \to C_0$. The rule R_2 is kept. If its converse is valid, we obtain the quasi-definition $C_0 \leftrightarrow \exists r_1{:}C_1$.

3.2 Redescriptions

Redescription mining [8] provides multiple characterizations of a given set of entities. Contrasting association rules, redescriptions rely on the separation of the set of attributes into *views*. The set of all views corresponds to a partition of the set of attributes. We work here with two views, corresponding to the two kinds of attributes we distinguished: M_C and M_D.

The similarity between the sets of attributes, coming from two different views, is measured thanks to the Jaccard coefficient:

$$\text{jacc}(A, B) = \frac{|A' \cap B'|}{|A' \cup B'|}$$

where $(.)'$ corresponds to the derivation operator. We say that the redescription holds if the Jaccard coefficient is above a given threshold. Contrary to confidence, the Jaccard coefficient is symmetric. A redescription with a Jaccard coefficient equal to 1 corresponds to a definition as introduced in the previous section. A redescription is necessarily a quasi-definition. Indeed,

$$\min(\text{conf}(A \to B), \text{conf}(B \to A)) \geqslant \text{jacc}(A, B).$$

Example 1. Given the context Figure 1, the two views are distinguished by the vertical line in gray. From this context, $\{C_0\} \leftrightarrow \{\exists r_1{:}x_1, \exists r_2{:}x_2\}$ is a redescription with a Jaccard coefficient of 1.

The algorithm `ReReMi` [5] is used in this work to mine redescriptions. It searches for a pair of attributes–one in each view–that may constitute a definition and tries to extend it by adding one attribute at each step. More than binary data, `ReReMi` also handles numerical and categorical data. It also enables to consider Boolean functions including conjunctions, disjunctions and negations over the attributes. Here, we only use a binary dataset and conjunctions of attributes in order to compare the results with the other algorithms.

3.3 Translation rules

The algorithm `Translator` [10] also relies on two views and searches for a set of associations between these two views, but the construction of the associations is based on a different approach, that is *minimum description length* (MDL).

The associations consist in rules that enable building one context from the other, as shown in Figure 2. The set of rules has to be compact and representative. In one hand, it should cover most of the data. In the other hand, the rules have to be as small as possible in term of attributes. To check these two constraints, `Translator` relies on MDL. Given $K = (G, M, I)$ a context and $X \subseteq M$

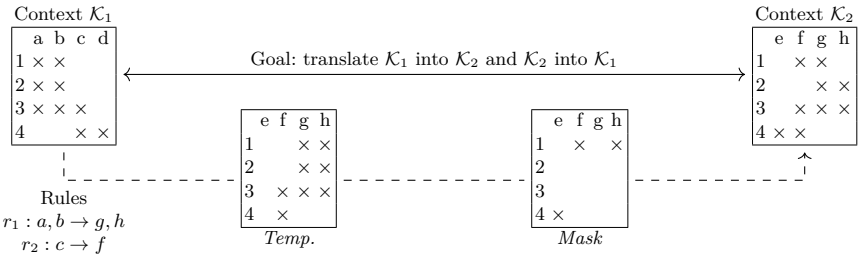


Fig. 2: `Translator` is searching for a set of rules that enables transforming $\mathcal{K}_2$ into $\mathcal{K}_1$ and $\mathcal{K}_1$ into $\mathcal{K}_2$. Here we represent only the construction of $\mathcal{K}_2$ from $\mathcal{K}_1$. For each object in $\mathcal{K}_1$ and for each rule, if the object has all the attributes of the condition, then all the attributes of the conclusion are added to $\mathcal{K}_2$.

a set of attributes, the *length* of X w.r.t. K corresponds to the minimum number of bits required in order to encode X. That is:

$$L(X) = -\sum_{x \in X} \log_2 P(x \mid K) \quad \text{where } P(x \mid K) = \frac{\mid x' \mid}{\mid G \mid}.$$

In [10], the authors compare the mining process to a translation task. A rule is considered as a *translation* from one context to an other. The underlying idea is that, with enough translation rules, one can build the first context from the other and *vice versa*. The general idea is depicted Figure 2. The errors introduced in the target context are fixed with a mask. Thus, the size of the mask corresponds to the number of errors added. The algorithm `Translator` compute rules step by step. At the beginning of the process, the mask corresponds to the target context. The algorithm searches for the rule which has the best trade-off between lowering density of the mask and not being too long, i.e. the rule which maximizes Δ:

$$\Delta(X \to Y) = \underbrace{L(Mask^-) - L(Mask^+)}_{\text{Information gain}} - \underbrace{L(X \cup Y)}_{\text{Rule length}}$$

where $Mask^+$ corresponds to the items added to the mask (errors introduced by the rule) and $Mask^-$ corresponds to items removed from the mask (errors fixed by the rule). Rules are added while $\Delta > 0$.

The mask is updated each time a rule is added. Since the information gain depends on the mask, the quality Δ of a rule depends on the rules that are already found. Thus, `Translator` is the only algorithm which takes into account rules already found to choose which rule is added.

4 Related work

In [1], authors rely on association rule mining to provide a navigation space over RDF resources. To this end, they search for implications and rank them

w.r.t. the confidence of their converse. Our work is in the continuity of this one: our purpose is the same, but here we use two other approaches and compare them.

The AMIE algorithmn, extended to AMIE+ [4], is a reference for mining rules in KBs. Those rules have the form $B_1 \wedge B_2 \wedge \ldots B_{n-1} \Rightarrow B_n$ where B_i is a relation between two objects $r(x_i, x_j)$. Authors add a constraint: all the variables have to appear twice in the rule, in different atoms. Our work is distinct from this one in two manners. We consider rules and their converse, and we do not focus on relations (i.e. predicates), but on the pair (predicate, object).

In a survey, Sertkaya [9] presents papers trying to bridge the gap between FCA and ontologies. In ontologies, the knowledge is constructed with top-down approaches (e.g experts who encode knowledge of a domain). At the contrary, in FCA, knowledge is discovered with a bottom-up approach, starting from facts and trying to generalize them. Thus, one way to take advantage of FCA is to allow bottom-up construction of ontologies and to complete existing ontologies. This is what is done in our approach: we start from RDF statements and try to find definitions of classes.

In [3], an extension to FCA for conceptual graphs, called G-FCA, is proposed. Compared to RDF graphs, conceptual graphs (CG) are oriented bipartite graphs. The two kinds of nodes are classes and relations. Contrasting RDF graphs which only consider binary relations, CGs handle n-ary relations. The approach enables to find *projected graph patterns*. A projected graph pattern is a pair containing a graph query and a set of candidate solutions. It is similar to a SPARQL query where the graph query is the intent and the candidate solutions are the extent. This work is complementary to our work in the sense that, instead of dealing with rule mining, it considers the full lattice.

5 Experiments

We run our experiments on *DBpedia* data, which is one of the most important knowledge bases of the linked open data, built from Wikipedia. We are interested in *categories* of *DBpedia*, that is, resources in the range of the relation `dct:subject`. Categories are a specific kind of classes. They are built from specific Wikipedia pages which lists other pages (for example the page Category:Smartphones[2]). The advantage of considering these categories instead of common classes is that there are much more categories than classes, and the only information about them provided in DBpedia is which resources belong to each category. Thus, finding why some resources are gathered together (for example, "because they all are smartphones") is an interesting challenge.

To this end, we extracted a subset of *DBpedia* thanks to a SPARQL query. The triples extracted are transformed in a context as presented in section 2.2. We run algorithms which are introduced above, then, we compare and evaluate the extracted quasi-definitions. Both data and results are available online[3].

[2] https://en.wikipedia.org/wiki/Category:Smartphones
[3] https://gitlab.inria.fr/jreynaud/DefinitionMiningComparison

Table 1: Statistics of the datasets extracted with the SPARQL query. `D` is one of the four domains, whereas the predicate `owl:objectProperty` ensures that $?o$ is a resource, and not a literal nor a blank node.

```
SELECT DISTINCT * WHERE {
 ?s ?p ?o .
 ?s dct:subject dbc:C .
 ?p a owl:ObjectProperty .
}
```

D	Triples	Objects	$\lvert M_C \rvert$	$\lvert M_D \rvert$
`Turing_Award`	2 642	65	503	857
`Smartphones`	8 418	598	359	1 730
`Sports_cars`	9 047	604	435	2 295
`French_films`	121 496	6 039	6 028	19 459

5.1 Methodology

We extracted four different subsets of triples, of different size and different domains, from *DBpedia*, with SPARQL queries. All the queries follow the same pattern. The datasets correspond to the categories `Smartphones`, `Sports_cars`, `Turing_Award_laureates` and `French_films`. Statistics of the datasets are provided Table 1.

For each dataset, the partition of the attributes is constructed as follows: M_C is the subset of attributes whose predicate is `dct:subject` whereas M_D is the set of attributes whose predicate differs from `dct:subject`. For `Eclat`, since both attributes of classes and descriptions are in the same context, the input data is one file which contains the context in a tabular format. For `ReReMi` and `Translator`, the input data are two tabular files.

5.2 Results

Each algorithm returns an ordered set of quasi-definitions. Each quasi-definition is manually evaluated by three phD students familiar with linked open data, playing the role of experts. Given a definition $C_0, \ldots, C_n \leftrightarrow D_0, \ldots, D_m$ from a dataset X, each evaluator answers the question " Taking X as a reference, is it true that *belonging to* C_0 *and* C_1 *... and* C_n and *having the properties* D_0 *and* D_1 *... and* D_m is equivalent ? " The final evaluation is the majority between the experts. Experts gave the same answer in 95.4% of the cases. If evaluated true, the quasi-definition is added to the set of definitions (see Fig. 3).

The comparison between the algorithms is based on definitions (i.e. quasi-definition evaluated as true by at least 2 experts) that have been extracted and categories that have been defined. Figure 4 shows two Venn diagrams for each dataset: one for the number of definitions extracted and one for the number of categories defined. In the dataset `Turing_Award_laureates`, for example, there are 22 definitions only extracted by `Eclat` and 8 definitions extracted by both `Eclat` and `Translator`. `Eclat` extracted 30 definitions in total. A category is considered as defined as soon as it appears in a definition. Therefore, in one definition, there can be one or more categories considered as defined.

Turing_Award_laureates

R	**Harvard_University_alumni** ↔ (almaMater Harvard_University)	R1
ET	**Harvard_University_alumni, Turing_Award_laureates** ↔ (a Agent), (a Person), (a Scientist), (almaMater Harvard_University)	R2
E	**Turing_Award_laureates** ↔ (a Agent), (a Person), (award Turing_Award)	R3
ET	**Turing_Award_laureates** ↔ (a Agent), (a Person), (a Scientist), (award Turing_Award)	R4
E	**Modern_cryptographers** ↔ (field Cryptography)	R5

Sports_cars

R	**McLaren_vehicles** ↔ (manufacturer McLaren_Automotive)	R6
R	**McLaren_vehicles** ↔ (assembly Surrey)	R7
ET	**McLaren_vehicles, Sports_cars** ↔ (a Automobile), (a MeanOfTransportation), (assembly Woking), (assembly Surrey), (assembly England), (bodyStyle Coupé), (manufacturer McLaren_Automotive)	R8
E	**McLaren_vehicles, Sports_cars** ↔ (a Automobile), (a MeanOfTransportation), (assembly England), (assembly Surrey), (bodyStyle Coupé)	R9
E	**McLaren_vehicles, Sports_cars** ↔ (a Automobile), (a MeanOfTransportation), (assembly Surrey), (bodyStyle Coupé)	R10

Smartphones

ET	**Firefox_OS_devices, Open-source_mobile_phones, Smartphones, Touchscreen_mobile_phones** ↔ (a Device), (operatingSystem Firefox_OS)	R11
R	**Nokia_mobile_phones** ↔ (manufacturer Nokia)	R12
ET	**Nokia_mobile_phones, Smartphones** ↔ (a Device), (manufacturer Nokia)	R13
R	**Samsung_Galaxy** ↔ (manufacturer Samsung_Electronics), (operatingSystem Android_(operating_system))	R14
ET	**Samsung_Galaxy, Samsung_mobile_phones, Smartphones** ↔ (a Device), (manufacturer Samsung_Electronics), (operatingSystem Android_(operating_system))	R15

French_films

R	**Pathé_films** ↔ (distributor Pathé)	R16
R	**Films_directed_by_Georges_Méliès** ↔ (director Georges_Méliès)	R17
ET	**Films_directed_by_Georges_Méliès, French_films, French_silent_short_films** ↔ (a Film), (a Wikidata:Q11424), (a Work), (director Georges_Méliès)	R18
ET	**Films_directed_by_Jean_Rollin, French_films** ↔ (a Film), (a Wikidata:Q11424), (a Work), (director Jean_Rollin)	R19
ET	**Film_scores_by_Gabriel_Yared, French_films** ↔ (a Film), (a Wikidata:Q11424), (a Work), (musicComposer Gabriel_Yared)	R20

Fig. 3: Definitions extracted by `Eclat`, `ReReMi` and `Translator` for each dataset. In order to be more readable, namespaces have been removed.

6 Discussion

Hereafter, we will denote $\mathcal{B}^X_{cand}$ the set of all the quasi-definitions extracted by the algorithm X and $\mathcal{B}^X_{def}$ the set of quasi-definitions from $\mathcal{B}^X_{cand}$ evaluated true by the experts, i.e. the set of definitions extracted by X. The set $\mathcal{B}_{cand}$ denotes the set of all the quasi-definitions definitions extracted, regardless the algorithm. Similarly, $\mathcal{B}_{def}$ denotes the set of all the definitions extracted.

6.1 Precision, recall and completeness

The precision of an algorithm X is $\frac{|\mathcal{B}^X_{def}|}{|\mathcal{B}^X_{cand}|}$. The precision of `ReReMi` has a high variability (from 33% to 75%) and is overall the weakest, especially for the dataset `French_films`. The precision of `Eclat` is stable (from 64% to 72%). `Translator` has the best precision which is always over 74%.

Table 2: Evaluation of the results. For each dataset, the number of quasi-definitions extracted ($|\mathcal{B}_{cand}|$) and evaluated true ($|\mathcal{B}_{def}|$) are reported, along with the average number of categories ($\overline{|C_i|}$) and descriptions ($\overline{|D_i|}$) per rule.

(a) `Turing_Award_laureates`

X	`Eclat`	`ReReMi`	`Translator`
$\lvert\mathcal{B}_{cand}\rvert$	47	12	11
$\lvert\mathcal{B}_{def}\rvert$	30	9	9
$\lvert\mathcal{B}^X_{def}\rvert/\lvert\mathcal{B}^X_{cand}\rvert$	.64	.75	.85
$\overline{\lvert C_i\rvert}$–$\overline{\lvert D_i\rvert}$	2–4	1–1	3–5

(b) `French_films`

X	`Eclat`	`ReReMi`	`Translator`
$\lvert\mathcal{B}_{cand}\rvert$	132	52	31
$\lvert\mathcal{B}_{def}\rvert$	95	30	23
$\lvert\mathcal{B}^X_{def}\rvert/\lvert\mathcal{B}^X_{cand}\rvert$	.72	.68	.74
$\overline{\lvert C_i\rvert}$–$\overline{\lvert D_i\rvert}$	2.8–4.5	1.3–1.4	2.6–4.1

(c) `Sports_cars`

X	`Eclat`	`ReReMi`	`Translator`
$\lvert\mathcal{B}_{cand}\rvert$	810	98	41
$\lvert\mathcal{B}_{def}\rvert$	521	57	31
$\lvert\mathcal{B}^X_{def}\rvert/\lvert\mathcal{B}^X_{cand}\rvert$	.64	.58	.76
$\overline{\lvert C_i\rvert}$–$\overline{\lvert D_i\rvert}$	4.3–7.8	1.6–1.8	3.1–3.1

(d) `Smartphones`

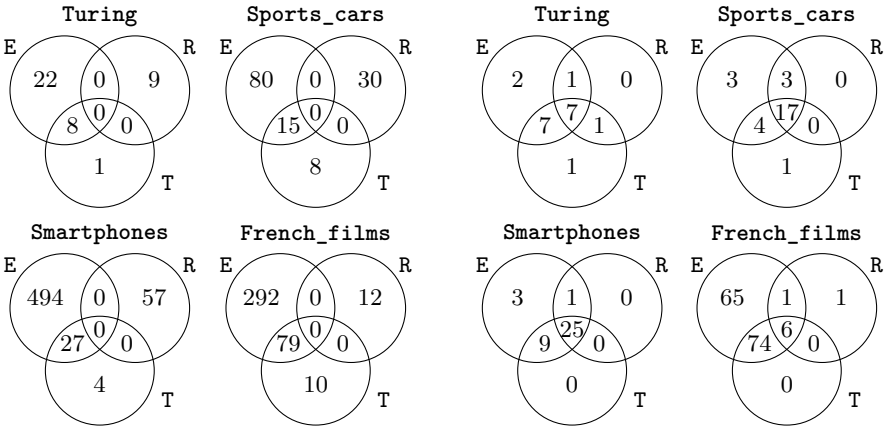
X	`Eclat`	`ReReMi`	`Translator`
$\lvert\mathcal{B}_{cand}\rvert$	546	36	93
$\lvert\mathcal{B}_{def}\rvert$	371	12	89
$\lvert\mathcal{B}^X_{def}\rvert/\lvert\mathcal{B}^X_{cand}\rvert$	.68	.33	.96
$\overline{\lvert C_i\rvert}$–$\overline{\lvert D_i\rvert}$	2.8–4.4	1.2–1.1	2.3–4.2

The recall could be defined as $\frac{|\mathcal{B}^X_{def}|}{|\mathcal{B}_{def}|}$. However, it cannot be used as a performance measure. Indeed, some of the definitions overlap (i.e. have attributes in common in both sides). This is the case for the rules R6 to R10 in Figure 3 : all the rules define the category `McLaren_vehicules`. Whereas `Translator` extracts only one rule (R8), `ReReMi` extracts 2 rules (R6 and R7) and `Eclat` extracts 9 rules (only 3 of them, R8 to R10, are reported here).

Given the valid quasi-definitions, the uncompleteness of the KB can be measured as the number of triples which can be inferred from the quasi-definitions and that are not already in the KB. For example, given the rule `Pathé_Films` $\leftrightarrow$ `(distributor Pathé)`, if a resource `r` belongs to `Pathé_Films` (i.e. $\langle$`r, subject, Pathé_Films`$\rangle \in KB$), then the triple $\langle$`r, distributor, Pathé`$\rangle$ is expected to be in the KB. Conversely, if the triple $\langle$`r, distributor, Pathé`$\rangle$ belongs to the KB, then $\langle$`r, subject, Pathé_Films`$\rangle$ is expected to be in the KB. Figure 4c counts, for each dataset, the number of inferred triples that were not in the KB.

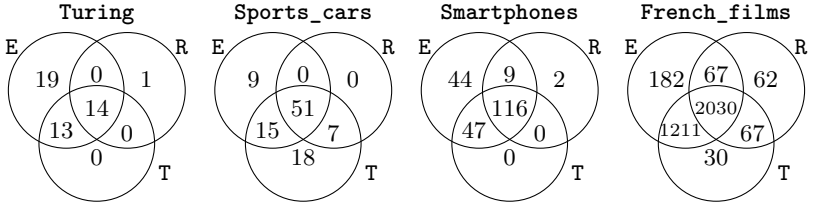
6.2 Shape and interpretation of the rules

From Figure 4, 70% of the categories defined by `Eclat` or `Translator` are defined by both algorithms. However, `Translator` extracts much less rules than `Eclat` (until 16 times less for the dataset `Smartphones`). This is due to the extraction process of association rules: if the rule $A \rightarrow B$ has the same support as the rule $A \rightarrow \{B, C\}$, then only the rule $A \rightarrow \{B, C\}$ is kept. However, if the support of $A \rightarrow B$ is higher, both rules are kept. Consequently, `Eclat` mines rule which can differ from only one attribute (R9 and R10), contrary to `Translator` (only R8).



(a) Definitions extracted.

(b) Categories defined.



(c) Triples inferred.

Fig. 4: Definitions extracted, categories defined, and triples inferred by `Eclat(E)`, `ReReMi(R)` and `Translator(T)` for each dataset.

None of the definitions mined by `ReReMi` are shared by `Eclat` or `Translator`. This is due to the heuristic used by `ReReMi`. If C is a category and D_1 and D_2 are two descriptions such that $C' = D'_1 = D'_2$, then `ReReMi` generates the two definitions $C \leftrightarrow D_1$ and $C \leftrightarrow D_2$ rather than one definition $C \leftrightarrow \{D_1, D_2\}$ as `Eclat` does. This is the case for definitions R6 and R7 mined by `ReReMi`, and R8 mined by `Eclat`. If $C' = D'_1$ and $D'_1 \subset D'_2$, `ReReMi` generates the definition $C \leftrightarrow D_1$ whereas `Eclat` generates $C \leftrightarrow \{D_1, D_2\}$, as shown with definitions R12 and R13 for example. Another consequence of the heuristic used by `ReReMi` is that it mines smaller definitions than definitions mined by `Eclat` and `Translator` wrt the number of attributes. On average, definitions mined by `ReReMi` have 1 or 2 attributes on each side whereas definitions mined by `Eclat` and `Translator` have 3 categories and 4 descriptions. These differences raise the question of the semantics of the conjunctions. Indeed, the semantics of the conjunctions in the definitions mined by `ReReMi` differs from the one in definition mined by `Eclat` and `Translator`. For example, in rule R15, the attribute `(a, Device)` can be removed without repercussion on the meaning. On the opposite side, in definition R14, no attribute can be removed without

changing the meaning of the definition. That is, all the attributes are necessary. In our approach, it seems more interesting to consider only attributes that are necessary in the definition. Thus, R14 is better than R15 according to the ease of interpretation.

7 Conclusion

In this paper, we compared three algorithms to find definitions in the linked open data. Each algorithm has its specificities and we verified that these specificities are reflected in the results of our experiments. We showed that, despite their very different approaches, `Eclat` and `Translator` extract a lot of identical rules. At the opposite, `ReReMi`, in spite of a quality measure very similar to `Eclat`, extracts shorter rules. The advantage of each algorithm depends on the goal of the user. In our experiments, `Eclat` is the algorithm which defines the most of the categories, at the cost of a huge number of quasi-definitions extracted. `Translator` extracts significantly less quasi-definitions but defines less categories. `ReReMi`, despite a low number of categories defined, offers definitions easier to understand which do not include attributes that do not contribute to the definition.

Acknowledgements

This work has been conducted with the support of "Région Lorraine" and "Délégation Générale de l'Armement".

References

1. M. Alam, A. Buzmakov, V. Codocedo, and A. Napoli. Mining definitions from RDF annotations using formal concept analysis. In *IJCAI*, pages 823–829, 2015.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
3. S. Ferré and P. Cellier. Graph-FCA in Practice. In *Proceedings of 22nd ICCS*, pages 107–121, 2016.
4. L. A. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB Journal*, 24(6):707–730, 2015.
5. E. Galbrun and P. Miettinen. From Black and White to Full Color: Extending Redescription Mining Outside the Boolean World. *Statistical Analysis and Data Mining*, 5(4):284–303, 2012.
6. B. Ganter and R. Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.
7. J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
8. N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R. F. Helm. Turning CARTwheels: an Alternating Algorithm for Mining Redescriptions. In *KDD'04*, pages 266–275, 2004.
9. B. Sertkaya. A survey on how description logic ontologies benefit from formal concept analysis. *CoRR*, abs/1107.2822, 2011.
10. M. van Leeuwen and E. Galbrun. Association Discovery in Two-View Data. *TKDE*, 27(12):3190–3202, Dec. 2015.
11. M. J. Zaki. Scalable algorithms for association mining. *TKDE*, 12(3):372–390, 2000.