# Redescription mining for learning definitions and disjointness axioms in Linked Open Data

Justine Reynaud, Yannick Toussaint, Amedeo Napoli

# Redescription mining for learning definitions and disjointness axioms in Linked Open 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 article, we present an original use of Redescription Mining (RM) for discovering definitions of classes and incompatibility (disjointness) axioms between classes of individuals in the web of data. RM is aimed at mining alternate descriptions from two datasets related to the same set of individuals. We reuse this process for providing definitions in terms of necessary and sufficient conditions to categories in DBpedia. Firstly, we recall the basics of redescription mining and make precise the principles of our definitional process. Then we detail experiments carried out on datasets extracted from DBpedia. Based on the output of the experiments, we discuss the strengths and the possible extensions of our approach.

**Keywords:** Redescription Mining - Linked Open Data - Definition of categories - Disjointness axioms - Formal Concept Analysis

## 1 Introduction

The Linked Open Data (LOD) cloud has become a very large reservoir of data over the last fifteen years. This data cloud is based on elementary triples (subject, predicate, object), denoted by $\langle s, p, o \rangle$, where $s$, $p$ and $o$ denote resources. These triples can be related to form a (huge) directed graph $G = (V, E)$ where vertices in $V$ correspond to resources –or individuals–, and edges in $E$ correspond to relations or predicate linking resources. Besides the graph structure, individuals can be grouped using the Resource Description Framework (RDF) thanks to the special predicate `rdf:type` in a class, and then individuals are "instances" of this class. In turn, using RDF Schema (RDFS), the set of classes can be organized within a poset thanks to the partial ordering `rdfs:subClassOf`.

A class can be defined through an *extension* by enumeration of all individuals composing this extension. For example, the extension of the `Smartphone` class would include the set of all "known" smartphones in a given universe. Dually, a class may also be defined through an *intension* by enumeration of all characteristics common to individuals in the class. For example, the intension of the `Smartphone` class could be described as "a small computer equipped with a cellular antenna".

* Supported by "Région Lorraine" and "Délégation Générale de l'Armement"

A standard classification problem is to provide a suitable definition to a class of individuals, i.e. a description based on a set of characteristics which are common to all individuals. This problem arises whenever there is a need for building classes for an ontology or a knowledge base related to a particular domain. Actually, such a classification process is related to clustering and to concept lattices in Formal Concept Analysis (FCA [8]).

Going back to the LOD cloud, there are classes defined by an extension but usually without any corresponding intension. More concretely, we may consider individuals as subjects $s$ whose description is composed of the set of available pairs $(p, o)$. A direct application of this classification problem is the mining of definitions of DBpedia categories, in the line of the work in [1]. Actually, DBpedia categories are automatically extracted from Wikipedia. In Wikipedia, a category is a specific page which lists all the pages related to itself, as is the case for example for the page `Category:Smartphones`[1]. In DBpedia, a category is a resource appearing in the range of the predicate `dct:subject`, thanks to the Dublin Core Metadata terms (DCT). Moreover, categories are widespread as there are more than one million of categories but, most of the time, a category does not have any "processable" description and there does not exist any ordering or structure among categories.

Accordingly, we can formulate our classification problem as follows: given a class defined by a set of instances, is it possible to find a corresponding definition in terms of a description made of set of characteristics or properties related to all these instances. Then, the class could be defined in terms of necessary and sufficient conditions for an individual to be a member of the class. The necessary condition means that all instances share the characteristics of the description while the sufficient condition means that any individual having those characteristics should be an instance of the class. In this work, we aim at defining the classes in two complementary ways: (i) by building a description shared by all the instances of a given class, (ii) by finding potential incompatible classes, i.e. classes which do not share any instance.

Actually, the present work is a continuation of a work initiated in [1] and in [12]. In [1], authors rely on FCA [8] and implication between concepts for discovering definitions in LOD. These definitions are based on pairs of implications, i.e. $C \Longrightarrow D$ and $D \Longrightarrow C$, which stand for necessary and sufficient conditions. A double implication is considered as a definition $C \equiv D$, but most of the time $C \Longrightarrow D$ is an implication (i.e. the confidence is 1) while $D \longrightarrow C$ is an association rule whose confidence is less than 1, meaning that the data at hand are incomplete but that the definition is plausible. In [12], we ran a preliminary comparison between several approaches for mining definitions in LOD, based on FCA, redescription mining and translation rule mining.

In the present paper, we focus on Redescription Mining (RM) [5,6]. RM aims at discovering alternate characterizations of a set of individuals from two sets of characteristics. The characterizations can be expressed thanks to Boolean connectors within propositional logic formulas. Thus, it appears that RM is a

[1] https://en.wikipedia.org/wiki/Category:SmartPhones

valuable and challenging candidate approach for discovering definitions in LOD. This research work is original and one of the first attempts to reuse redescription mining for mining definitions in LOD. Moreover, the fact that negation can be taken into account allows us to extend the FCA-based approach and to mine not only definitions but also disjointness axioms, as we have in a notation borrowed from Description Logics[2] [2]: $C \equiv D \iff C \sqsubseteq D$ and $D \sqsubseteq C$ and $C \sqcap D \equiv \bot \iff C \sqsubseteq \neg D$ or $D \sqsubseteq \neg C$. This approach was applied on various datasets from DBpedia [10] running the so-called `ReReMi` algorithm [5], and has shown a very good practical behavior.

The paper is organized as follows. Section 2 presents the problem statement while Redescription Mining is introduced in Section 3. Section 4 is related to experiments which are conducted to evaluate the approach, and a discussion on the quality of the results and the possible improvements. Finally, Section 5 includes related work preceding Section 6 with future work and conclusions.

## 2 Problem Statement in FCA

### 2.1 Basics of FCA and implications

Formal Concept Analysis (FCA) is a mathematical framework mainly used for classification and knowledge discovery [8]. FCA starts with a formal context $(G, M, I)$ where $G$ is a set of objects, $M$ a set of attributes, and $I \subseteq G \times M$ a binary relation, with $gIm$ meaning that object $g$ has attribute $m$. Two dual derivation operators, denoted by $'$, are defined as follows:

$$A' = \{m \in M / \forall g \in A, gIm\} \text{ for } A \subseteq G \text{ and}$$
$$B' = \{g \in G / \forall m \in B, gIm\} \text{ for } B \subseteq M$$

The two compositions of the both derivation operators, denoted by $''$, are closure operators. In particular, for $A \subseteq G$ and $B \subseteq M$, we have $A \subseteq A''$ and $B \subseteq B''$. Then $A$ and $B$ are closed sets when $A = A''$ and $B = B''$ respectively. Moreover, a pair $(A, B)$ is a "concept" whenever $A' = B$ and $B' = A$, where $A$ is closed and called the "extent" of $(A, B)$, and $B$ is closed and is called the intent of $(A, B)$. The set of concepts is organized within a "concept lattice" thanks to the partial ordering defined by $(A_1, B_1) \leq (A_2, B_2)$ when $A_1 \subseteq A_2$ or dually $B_2 \subseteq B_1$.

Two types of rules can be extracted from concepts, namely "association rules" and "implications". An implication $B_1 \Longrightarrow B_2$ states that all objects having all attributes in $B_1$ have all attributes in $B_2$, i.e. $B_1^{'} \subseteq B_2^{'}$. The implication $B_1 \Longrightarrow B_2$ has a *support* defined as the cardinality of the set $B_1^{'} \cap B_2^{'}/G$, and a *confidence* defined as the cardinality of the set $B_1^{'} \cap B_2^{'}/B_1^{'}$. The confidence can be interpreted as a conditional probability:

$$\text{support}(B_1 \Longrightarrow B_2) = \frac{|B_1' \cap B_2'|}{|G|} \quad \text{and} \quad \text{conf}(B_1 \Longrightarrow B_2) = \frac{|B_1' \cap B_2'|}{|B_2'|}.$$

[2] We adopt this formalism for the readers of this paper. Finding the good representation of the rules for domain experts is out of the scope of this paper.

The confidence is used for measuring the quality of a rule. The confidence of an implication is always 1, and which is not the case for an "association rule" $B_1 \longrightarrow B_2$. Then, an association rule is "valid" if its confidence is above a given threshold $\theta$.

Finally, if both $B_1 \Longrightarrow B_2$ and $B_2 \Longrightarrow B_1$, then the definition $B_1 \equiv B_2$ or $B'_1 = B'_2$ can be inferred.

### 2.2 Defining Categories in DBpedia

The content of *DBpedia* is built with information extracted from *Wikipedia*, an online encyclopedia. In *Wikipedia*, a category say $X$ is a specific kind of Wikipedia page listing all pages related to $X$ (see page `Category:Smartphones`[3] for example). In *DBpedia*, a category appears in RDF triples in the range of the relation `dct:subject`. For example, the triple $\langle \mathtt{x}, \mathtt{dct:subject}, \mathtt{Smartphones} \rangle$ states that the `x` subject belongs to the `Smartphones` "category".

Moreover, speaking in terms of knowledge representation and reasoning, the name of a category is a purely syntactic expression, and thus a category does not have any formal definition as one could expect (see discussion in [1] on this aspect). Then it is impossible to perform any classification within the set of categories as the latter are not defined in terms of necessary and sufficient conditions. This is precisely what we want to deal with, i.e. providing a definition to a category. This amounts to finding pairs of the form $(C, \{d_1, \ldots, d_n\})$ where $C$ denotes a category, such as `Nokia_Mobile_Phone` for example, and $d_i$ denotes a pair $(p, o)$, such as `(manufacturer,Nokia)` for example. Then the whole set of $d_i$ will stand for a possible description of $C$. A parallel can be drawn with concept definitions in Description Logics [2], where a form of definition is given by $C \equiv d_1 \sqcap \cdots \sqcap d_n$, such as:

$$\mathtt{Nokia\_Mobile\_Phone} \equiv \mathtt{Phone} \sqcap \exists \mathtt{manufacturer.Nokia}$$

Following the same line, we aim also at finding "incompatible categories", i.e. pairs of categories $(C_i, C_j)$ such as there does not exist any subject $s$ verifying both $\langle \mathtt{s}, \mathtt{dct:subject}, \mathtt{C_i} \rangle$ and $\langle \mathtt{s}, \mathtt{dct:subject}, \mathtt{C_j} \rangle$. In terms of Description Logics, this is written as $C_i \sqcap C_j \equiv \bot$.

For example, $\mathtt{Nokia\_Mobile\_Phone} \sqcap \mathtt{Turing\_Award\_laureate} \equiv \bot$ states that two categories are disjoint or incompatible, which is a particular type of definition, meaning in terms of sets of instances that `Nokia_Mobile_Phone` is in the complementary of `Turing_Award_laureate`.

Both types of definitions are useful for a practitioner aiming at contributing to *DBpedia*. Indeed, providing descriptions and then definitions to categories allows to be in agreement with knowledge representation principles, i.e. building sound and complete definitions of individual classes, as categories should be. In particular, this would help to find missing triples. For example, suppose that the definition $\mathtt{Nokia\_Mobile\_Phone} \equiv \mathtt{Phone} \sqcap \exists \mathtt{manufacturer.Nokia}$ is

[3] https://en.wikipedia.org/wiki/Category:Smartphones

lying in *DBpedia*. Then, if an element `x` belongs to `Nokia_Mobile_Phone`, then this element should be a phone with manufacturer Nokia, i.e. `x` is an instance of $\texttt{Phone} \sqcap \exists \texttt{manufacturer}.\texttt{Nokia}$ ("necessary condition"). Conversely, if an element is an instance of $\texttt{Phone} \sqcap \exists \texttt{manufacturer}.\texttt{Nokia}$, then `x` should be an instance of `Nokia_Mobile_Phone` ("sufficient condition"). This allows to complete incomplete triples if required.

In addition, specifying incompatible categories enables to track inconsistencies. Indeed, suppose there exits a triple in *DBpedia* asserting that `Smartphones` and `Sports_cars` are incompatible. Whenever a practitioner tries to associate the category `Sports_cars` to a resource related to the category `Smartphones`, she/he could be warned that both categories are incompatible. This will guide practitioners and help them having better practices.

### 2.3 A Practical Approach in FCA

Following the lines of [1] in the FCA framework, the discovery of category definitions relies on the construction of a context $(G, M, I)$ from a set of triples denoted by $ST$. Given $ST$, $G$ is the set of subjects, i.e. $G = \{s / \langle s, p, o\rangle \in ST\})$ and $M$ is a set of pairs predicate-objects, i.e. $M = \{(p, o) / \langle s, p, o\rangle \in ST\})$. The incidence relation is defined as $sI(p, o) \Longleftrightarrow \langle s, p, o\rangle \in ST$.

Then the discovery process is based on a search for implications of the form $B_1 \Longrightarrow B_2$ where $B_1, B_2 \subseteq M$. Whenever an implication $B_1 \Longrightarrow B_2$ is discovered, the converse rule is checked. If $B_2 \Longrightarrow B_1$ is also an implication, then we have the definition $B_1 \equiv B_2$. If this is not the case, the set of triples involved in the context should be checked for potential incompleteness.

In the following, we present an alternative search for category definition based on "Redescription Mining", where the name of the category appears on the left hand side of the $\equiv$ symbol and a set of characteristics (composed of $\exists predicate.object$ expressions) appears on the right hand side.

## 3 Redescription Mining

### 3.1 Definitions

Redescription mining aims at searching for data subsets with multiple descriptions, as different views on the same set of objects [5,6]. Redescription mining takes as input a set of objects $G$ and a set of attributes $M$ partitioned into *views* $V_i$ such as $M = V_1 \cup \cdots \cup V_n$ and $V_i \cap V_j = \emptyset$ if $i \neq j$. For example, the attributes can be partitioned w.r.t. the sources of the data or w.r.t. some criteria defined by a user. A value is associated to each pair $(object, attribute)$, which can be Boolean, numerical or nominal, and which depends on the domain of the attribute. An example of such a dataset is provided in Figure 1.

Given a set of objects $G$, a partition of a set of attributes $M$, redescription mining aims at finding a pair of "queries" $(q_1, q_2)$, where $q_1$ and $q_2$ correspond to logical statements involving attributes and their values. These statements are

| Views | $V_1$ | | | $V_2$ |
|---|---|---|---|---|
| Attributes | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| $f_1$ | | 2 | 3 | Triangle |
| $f_2$ | | 3 | 3 | Triangle |
| $f_3$ | × | 0 | 3 | Triangle |
| $f_4$ | × | 2 | 3 | Triangle |
| $f_5$ | × | 2 | 4 | Rectangle |

$a_1$: Has a right angle (Boolean)
$a_2$: Max number of equal sides (numerical)
$a_3$: Total number of sides (numerical)
$a_4$: Type (nominal)

Fig. 1: An example of dataset for redescription mining, with objects $\{f_1, \ldots, f_5\}$ and attributes $\{a_1, a_2, a_3, a_4\}$.

expressed in propositional logic with the conjunction, disjunction and negation connectors. Below, a redescription say $RD$ based on the pair $(q_1, q_2)$ is denoted by $RD = q_1 \longleftrightarrow q_2$ or $RD = (q_1, q_2)$.

Given a redescription $RD = q_1 \longleftrightarrow q_2$, the set of objects $G$ can be partitioned w.r.t. the queries which are satisfied by a subset of objects. There are four possible partitions, denoted by $E_{ij}$ with $i, j \in \{0, 1\}$, depending on the partition $q_1$ or $q_2$ which is satisfied. For example, $E_{10}(R)$ denotes the set of objects satisfying $q_1$ but not $q_2$.

Redescriptions are mined w.r.t. a support, the Jaccard coefficient, and a p-value. The support of a redescription $RD = (q_1, q_2)$ is the proportion of objects in the dataset satisfying both queries $q_1$ and $q_2$, i.e. $\text{support}(R) = \frac{|E_{11}(R)|}{|G|}$.

The similarity between two datasets corresponding to two queries $q_1$ and $q_2$ is measured thanks to the Jaccard coefficient:

$$\text{jacc}(q_1 \leftrightarrow q_2) = \frac{|E_{11}(R)|}{|E_{11}(R)| + |E_{10}(R)| + |E_{01}(R)|}$$

Let us consider for example the redescription $RD = (a_2 = 2) \longleftrightarrow (a_4 = Triangle)$ which is based on $q_1 = (a_2 = 2)$ and $q_2 = (a_4 = Triangle)$ w.r.t. the dataset in Figure 1. We have that: $|E_{11}(R)| = |\{f_1, f_4\}| = 2$, $|E_{10}(R)| = |\{f_5\}| = 3$, $|E_{01}(R)| = |\{f_2, f_3\}| = 4$ and $|E_{00}(R)| = 0$. Then it comes that $\text{support}(RD) = \frac{2}{5}$ and $\text{jacc}(RD) = \frac{2}{2+3+4} = \frac{2}{9}$. This means that the redescription $RD$ is not of very good quality.

By contrast, the redescription $(a_3 = 3) \longleftrightarrow \neg(a_4 = Rectangle)$ returns a Jaccard coefficient of 1 which is maximal, meaning this time that we have a very good redescription.

### 3.2 A Redescription Mining Algorithm

In this paper, we reuse the `ReReMi` algorithm to mine redescriptions [5]. `ReReMi` takes two files $D_1$ and $D_2$ as input, which correspond to two subsets of attributes or "views" $V_1$ and $V_2$ in the dataset, and returns a set of redescriptions.

Firstly, a "candidate redescription" based on a given set of pairs $(q_1, q_2)$, where $q_1$ contains only one attribute $\{a_1\} \subseteq V_1$ and $q_2$ only one attribute $\{a_2\} \subseteq$

$V_2$, is checked. The checking is not necessarily systematic for all possible pairs or combinations of pairs of attributes, as a set of initial pairs can be specified by an analyst. Doing so, the set of candidate redescriptions is progressively extended, i.e. one attribute is added at a time to one of the queries of the candidate redescription.

A query $q$ can be extended with a new attribute $a$ in four possible ways: $q_1 \wedge a$, $q_1 \vee a$, $q_1 \wedge \neg a$ or $q_1 \vee \neg a$. The redescription with the best Jaccard coefficient is added to the candidate redescriptions. However, this extension can be customized using for example only one of the possibilities, e.g. $q_1 \wedge a$. The algorithm continues until there is no more candidate available, i.e. until there is no way to increase the Jaccard coefficient of the current candidate redescription. Finally, the set of the candidate redescriptions is returned to the analyst.

### 3.3 Redescription mining in Linked Open Data

For applying redescription mining to a set of linked data, i.e. a set of related RDF triples, we need first to transform this set of triples into a format that can be processed by the `ReReMi` algorithm. This operation is similar to the building of a context in the FCA framework. The attributes correspond to the predicates of the triples and they are separated into views.

Given a set of triples $ST$, we build an input "context" $(G, M, I)$ where objects correspond to subjects of the RDF triples, i.e. $G = \{s/\langle s, p, o\rangle \in ST\})$ and attributes to the set of pairs "(predicate, object)", i.e. $M = \{(p, o)/\langle s, p, o\rangle \in ST\})$. The relation $I \subseteq G \times M$ is Boolean and we have $sI(p, o)$ is `true` whenever $\langle s, p, o\rangle \in ST$.

Next, the set of attributes is partitioned into two views as follows. $M = M_{subj} \cup M_{desc}$ and $M_{subj} \cap M_{desc} = \emptyset$. $M_{subj}$ is the set of attributes $(p, o)$ such that $p =$ `dct:subject` and the set $M_{desc}$ is the complementary set in $M$ (i.e. pairs $(p, o)$ where $p \neq$ `dct:subject`. Based on that, searching for a category definition car be achieved in two complementary ways:

(i) by providing a description to the category: in this case, there is a search for redescriptions $(q_1, q_2)$ where $q_1 = a$ with $a \in M_{subj}$ and $q_2$ is a query based on a set of one or more attributes from $M_{desc}$. Actually, this search should output a definition based on characteristics shared by all the resources of the category, actually a set of necessary and sufficient conditions for being a member of the category.

(ii) by determining which categories are incompatible: in this case, there is a search for categories which do not share any common resources, i.e. $C_i \sqcap C_j \equiv \bot$. Then the redescriptions are only based on $M_{subj}$ and the Jaccard coefficient of the categories in the output should be close to 0 instead of 1.

Table 1: Statistics on the datasets extracted.

| D | Triples | Objects | $\lvert M_{subj}\rvert$ | $\lvert M_{desc}\rvert$ | Density |
|---|---|---|---|---|---|
| `Turing_Award` | 2 642 | 65 | 503 | 857 | 3.9e−2 |
| `Smartphones` | 8 418 | 598 | 359 | 1 730 | 6.7e−3 |
| `Sports_cars` | 9 047 | 604 | 435 | 2 295 | 5.5e−3 |
| `French_films` | 121 496 | 6 039 | 6 028 | 19 459 | 7.9e−4 |

## 4 Experiments

### 4.1 Datasets

We extracted four different subsets of triples[4], corresponding to the domains `Turing_Award_laureates`, `Smartphones`, `Sports_cars`, and `French_films` in *DBpedia*, whose statistics are given in Table 1. The `Turing_Award_laureates` dataset is small with only 65 objects and less than 1500 attributes, meaning that there are less than 1500 unique pairs $(predicate, object)$ in the extracted triples. The dataset `French_films` is the largest, with more than 6000 objects and 25000 attributes. This dataset is rather sparse and the attributes have a weak support, and the density is very low as well. The datasets `Smartphones` and `Sports_cars` are similar in size, with roughly 600 objects and between 2000 and 2800 attributes.

For each dataset, the partition of the attributes is built as follows: $M_{subj}$ is constructed from the subset of triples whose predicate is `dct:subject` whereas $M_{desc}$ is the complementary set. Here, there are only Boolean attributes and only conjunction is used in RM. From $M_{subj}$ and $M_{desc}$, two tabular files compliant with `ReReMi` input are created, namely $D_{subj}$ which contains attributes of the view $M_{subj}$, $D_{desc}$ which contains attributes of the view $M_{desc}$. The thresholds used are 0.5 for Jaccard similarity (jacc $\geqslant 0.5$) and 3 for support (support $\geqslant 3$).

The discovery of incompatible categories relies only on the use of $M_{subj}$, and $D_{subj}$ is provided for both views. The thresholds used are 0.3 for Jaccard similarity with this time jacc $\leqslant 0.3$ and 5 for support (support $\geqslant 5$).

### 4.2 Extraction of Definitions

The `ReReMi` algorithm returns a set of redescriptions with their respective Jaccard coefficients. For measuring the precision of the algorithm, each redescription is manually evaluated by a domain expert. Hereafter, a redescription which is considered as "valid" by the expert is called a definition. This allows us to compute the precision as the ratio of definitions to redescriptions (see in section 4.4). Table 2 presents the redescriptions extracted along with their Jaccard coefficient.

In the `Turing_Award_laureates` dataset, most of the discovered definitions are about universities (redescriptions `R1` and `R2`), whereas definitions discovered

[4] The datasets and the results of the experiments are available online, see https://gitlab.inria.fr/jreynaud/iccs19-redescriptions.

Table 2: Definitions extracted by ReReMi for each dataset, along with their corresponding Jaccard coefficient, written in a Description Logics-like formalism. If the evaluator answered true to the question, the symbol $\equiv$ is used. Otherwise, the symbol $\not\equiv$ is used.

| N. | Redescription | jacc |
|---|---|---|
| | **Turing_Award_laureates** | |
| R1 | Harvard_University_alumni $\equiv$ ∃almaMater.Harvard_University | .89 |
| R2 | Stanford_University_alumni $\equiv$ ∃almaMater.Stanford_University | .56 |
| R3 | British_computer_scientists $\not\equiv$ ∃award.Fellow_of_the_Royal_Society | .63 |
| | **Sports_cars** | |
| R4 | McLaren_vehicles $\equiv$ ∃manufacturer.McLaren_Automotive | .86 |
| R5 | McLaren_vehicles $\equiv$ ∃assembly.Surrey | .75 |
| R6 | 2010_automobiles ⊓ Audi_Vehicles $\not\equiv$ ∃manufacturer.Audi | .55 |
| | **Smartphones** | |
| R7 | Nokia_mobile_phones $\equiv$ ∃manufacturer.Nokia | .82 |
| R8 | Samsung_Galaxy $\equiv$ ∃manufacturer.Samsung_Electronics ⊓ ∃operatingSystem.Android_OS | .66 |
| R9 | MeeGo_Devices $\not\equiv$ ∃operatingSystem.Sailfish_OS | .73 |
| | **French_films** | |
| R10 | Films_directed_by_Georges_Méliès $\equiv$ ∃director.Georges_Méliès | .98 |
| R11 | Film_scores_by_Georges_Delerue $\equiv$ ∃musicComposer.Georges_Delerue | .82 |
| R12 | Films_directed_by_Georges_Méliès $\not\equiv$ ∃director.Georges_Méliès ⊓ ∃language.Silent_Film | .50 |

in the two datasets Sports_cars and Smartphones are mostly about manufacturers. In the dataset French_films, all the redescriptions except one are related to "Georges Méliès". This means that such attributes have a support high enough to supporting redescriptions.

Most of the "invalid" mined redescriptions are based on a description which is too "approximate", i.e. there are possibly too many exceptions to the rule. For example, a large proportion of British computer scientists are also fellows of the Royal Society, but not all are award winners (see rule R3). In some other cases, there are not enough counter-examples in the dataset. For example, in redescription R9, there are too few Meego smartphones which are not running Sailfish in the dataset.

### 4.3 Extraction of the Incompatible Categories

The results about the extraction of incompatible categories are a bit disappointing for the dataset Turing_Award_laureates. Indeed, most of the categories discovered by ReReMi are not incompatible. This is maybe due to the fact that this dataset is about persons. Then the categories are characterizations of these persons w.r.t. a part of their life (e.g. where they studied and when they were born). Thus, most of the categories in this dataset cannot be incompatible, and there are too few objects to provide counter-examples.

Table 3: Incompatibilities discovered by ReReMi for each dataset. In all reported cases jacc = 0. The axioms are written in a Description Logics-like formalism.

| N. | Incompatible categories |
|---|---|
| | **Turing_Award_laureates** |
| R13 | Harvard_University_alumni ⊓ Scientist_from_California ≡ ⊥ |
| R14 | Fellows_of_the_British_Computer_Society ⊓ Jewish_American_scientists ≡ ⊥ |
| R15 | Massachusetts_Institute_of_Technology_faculty ⊓ IBM_Fellows ≡ ⊥ |
| | **Sports_cars** |
| R16 | 1970s_automobiles ⊓ Cars_introduced_in_1998 ≡ ⊥ |
| R17 | Kit_cars ⊓ Coupes ≡ ⊥ |
| R18 | 1960s_automobiles ⊓ Lotus_racing_cars ≡ ⊥ |
| | **Smartphones** |
| R19 | Blackberry ⊓ Nokia_mobile_phones ≡ ⊥ |
| R20 | Mobile_phones_introduced_in_2013 ⊓ Mobile_phones_introduced_in_2014 ≡ ⊥ |
| R21 | Touchscreen_mobile_phone ⊓ Nokia_platforms ≡ ⊥ |
| | **French_films** |
| R22 | 1980s_drama_films ⊓ 1970s_comedy_films ≡ ⊥ |

The other datasets provide better results. In the Sports_cars dataset, a lot of categories are incompatible because they denote cars from different time span, such as redescription R16 in Table 3. In the Smartphones dataset, a lot of categories are incompatible because they denote phones from different brands, such as redescription R19. For these two datasets, the ReReMi algorithm discovers a lot of incompatible categories. Finally, only one redescription is returned for the French_films dataset.

### 4.4 Discussion

Table 4: Results of the two experiments for each dataset. In the definition discovery settings, the number of extracted redescriptions ($|\mathcal{R}|$) and evaluated as true ($|\mathcal{D}|$) are reported, along with the precision ($\frac{|\mathcal{R}|}{|\mathcal{D}|}$). In the settings of incompatible categories, the number of disjunctions axioms extracted ($|\mathcal{N}|$) is reported.

| | Nb triples | $\|\mathcal{R}\|$ | $\|\mathcal{D}\|$ | Prec. | $\|\mathcal{N}\|$ |
|---|---|---|---|---|---|
| Turing_Award_laureates | 2642 | 12 | 9 | .75 | 30 |
| Smartphones | 8418 | 36 | 12 | .67 | 121 |
| Sports_cars | 9047 | 98 | 57 | .58 | 63 |
| French_films | 121496 | 52 | 30 | .58 | 1 |

The number of extracted category definitions along with the number of incompatible categories are reported in Table 4. These results are specific and

depends on the data domain, and thus cannot be generalized to the whole *DBpedia*. For discovering more general definitions, we probably need to process larger datasets, e.g. instead of `Turing_Award_laureates`, considering a dataset about `Person` . This would bring at the same time scalability issues that may be overcome with sampling or by using a pre-processing to select only a subset of predicates or by optimising the criteria used by the algorithm. Further experimentation in this direction could be considered in the future.

Discovered rules may look trivial, because DBpedia uses "explicit" labels for categories. However, in LOD, all categories are not labelled by an explicit name. In Wikidata, for example, the category corresponding to the French films is `Q393063`. Our approach do not use the semantics of the labels and can be generalised to other knowledge bases which do not use explicit labels.

The experiments also demonstrate the difficulty of data selection. Here, we use datasets of various sizes and domains. Finding a set of categories allowing to extract a good set of triples w.r.t. the constraints of the experiments is not straightforward. This calls for complex SPARQL queries and this underlines the interest of having better information about categories.

Most of the time, there is only one attribute in the right side of a redescription, meaning that such an attribute is very discriminant and that redescriptions do not have any attribute in common. Then, it can be difficult to build a partial ordering between the defined categories. By contrast, in the `Smartphones` dataset, we have the redescription `R8` and

$$\texttt{Samsung\_Mobile\_Phone} \equiv \exists \texttt{manufacturer.Samsung\_Electronics}$$

In this case, from these two redescriptions, we can infer that

$$\texttt{Samsung\_Galaxy} \sqsubseteq \texttt{Samsung\_Mobile\_Phone}$$

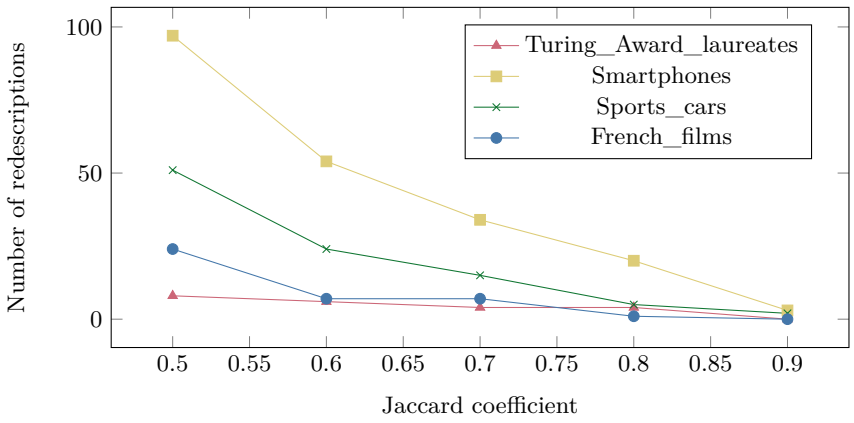


Fig. 2: Number of redescriptions extracted w.r.t. the Jaccard coefficient.

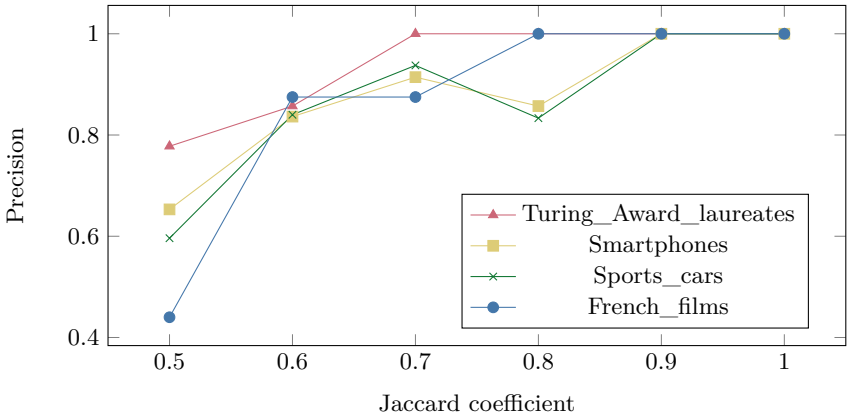


Fig. 3: Precision of the redescriptions w.r.t. the Jaccard coefficient.

Figure 2 shows the number of redescriptions found w.r.t. the Jaccard coefficient. Compared to association rules, the number of redescriptions is 2 to 10 times less [12]. The number of extracted redescriptions seems to be correlated with the density of the dataset, i.e. "the more dense the dataset is, the more redescriptions are extracted". This graduality becomes less important when the Jaccard coefficient increases.

Figure 3 shows the precision w.r.t. the Jaccard coefficient. The precision increases w.r.t. the threshold of the Jaccard coefficient, meaning that the Jaccard coefficient is a suitable measure for redescription mining in LOD. The precision depends on the datasets. It seems to be correlated to the size of the dataset and/or to the number of extracted redescriptions. However, further experiments should be performed to test this hypothesis. The low score of the `Turing_Award_laureates` dataset can be explained in two ways. Either the dataset is too small to mine definitions, or this is due to the nature of the dataset. Again, the fact that `Turing_Award_laureates` dataset is about persons could also explain the difference with the other datasets.

Finally, the results are interesting regarding both incompatible categories and definitions; Even with a low precision, the definitions which are obtained make sense and are quite easy to interpret for an analyst. However, the results obtained for the incompatible categories are a bit different. There are only a few incompatible categories and this fact is not due to the approximation of the Jaccard coefficient, since $\text{jacc} = 0$ for every pair of incompatible categories. This could mean that discovering category definitions and discovering incompatibles categories are not dual problems, although the main difference between the two tasks is based on the value of the Jaccard coefficient.

On a more semantic level, given a set $S$, it cannot be straightforwardly stated that any element which is not in $S$ is necessarily incompatible with elements in $S$, especially if we work in terms of open world assumption. This last point is also matter to future work.

## 5 Related Work

In [14], authors rely on *evidential terminological decision trees* (EDTD) to classify instances w.r.t. assertions in which they are involved. An EDTD is a decision tree where nodes are labeled with a logical formula and a value in $[0, 1]$ which can be interpreted as the probability of the logical formula to be true. This allows the authors to match a class with an assertion. The lower is the assertion in the tree, the more specific it is. To complete, the same authors in [13] search for a set of pairwise disjoint clusters in building a decision tree where each node corresponds to a concept description. Then, two concept descriptions at different leaf nodes are necessary disjoint.

By contrast, in [9], authors rely on rule mining and search for *obligatory class attributes*. Given a class, an obligatory attribute denotes a relation that every individual of the class should be involved in, e.g. every person has a birthdate, and then `hasBirthdate` is an obligatory attribute of class `Person`. While in [9], authors are not interested in the range of relations, authors in [1] take into account both relations and their range. They rely on FCA [8] and extracted association rules to define classes. Attributes are based on pairs $\mathtt{A_i} = (\mathtt{predicate_i}, \mathtt{object_i})$ and implications $A_i \Rightarrow A_j$ are searched. Only implications whose converse has a high support w.r.t. are kept as candidate definitions.

FCA and association rules are also used in [16], where authors build different formal contexts in order to discover specific relations, such as subsumption between two classes or transitivity of a relation for example.

In [11], authors aim to classify resources from RDF data, focusing on the relations existing between resources. For a resource $s$ and a class $C$, they compute the probability of $s$ to belong to $C$ w.r.t. the relations in $s$. For example, resources with the relation `hasBirthdate` are instances of the class `Person`.

Contrasting other approaches, authors in [3,4] consider the RDF graph and propose the algorithm AMIE+, which mainly focuses on relations, without considering domain and range. AMIE+ searches for implications between relations. For example, *people married to a person who lives in some place P also live in P* is the kind of rule that can be extracted by AMIE+.

We position ourselves in the continuity of these works. However, while most of the approaches search for implications, we search for definition and disjunctions using redescription mining. Regarding disjunction, authors in [15] propose a gold standard for class disjointness in DBpedia and compare a supervised approach based on machine learning and a statistical approach based on schema induction for learning disjointness. Their approach shows some similarities with our own approach as detailed in [12].

## 6 Conclusion and Future Work

In this paper, we present an original use of redescription mining for discovering definitions and disjunctions of categories in DBpedia. The approach involves RM in a very original task; The experimental results show that the approach is well-founded and comparable to related work approaches.

In future work, we would like to make more usage of the expressiveness of RM, i.e. using $\neg$ and $\vee$ Boolean connectors, for discovering more complex redescriptions. However, one problem could be the scalability when processing large sparse datasets. Moreover, another improvement would be to consider datatype properties such as dates or distances. Since `ReReMi` handles numerical data, we could discover redescriptions including literals such as "cars manufactured in 1997". Finally, another research direction is related to attributes which are partially ordered. This is possible in FCA thanks to pattern structures [7]. However, this is not yet integrated in RM. Such an extension would allow to discover $\texttt{SongWriter} \equiv \exists \texttt{isCreating}.\texttt{Song}$ from the triples $\langle \texttt{x}, \texttt{a}, \texttt{dbo} : \texttt{SongWriter} \rangle$, $\langle \texttt{x}, \texttt{isCreating}, \texttt{y} \rangle$, and $\langle \texttt{y}, \texttt{a}, \texttt{dbo} : \texttt{Song} \rangle$.

## References


1. Alam, M., Buzmakov, A., Codocedo, V., Napoli, A.: Mining definitions from RDF annotations using formal concept analysis. In: IJCAI. pp. 823–829 (2015)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
3. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.M.: AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In: WWW'13. pp. 413–422 (2013)
4. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB Journal **24**(6), 707–730 (2015)
5. Galbrun, E., Miettinen, P.: 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. Galbrun, E., Miettinen, P.: Redescription Mining. Springer Briefs in Computer Science, Springer (2017)
7. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: ICCS. pp. 129–142. LNCS 2120, Springer (2001)
8. Ganter, B., Wille, R.: Formal concept analysis - mathematical foundations. Springer (1999)
9. Lajus, J., Suchanek, F.M.: Are All People Married? Determining Obligatory Attributes in Knowledge Bases. In: International Conference WWW (2018)
10. Lehmann, J., Isele, R., Jakob, M., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
11. Paulheim, H., Bizer, C.: Type inference on noisy rdf data. In: International Semantic Web Conference. pp. 510–525 (2013)
12. Reynaud, J., Toussaint, Y., Napoli, A.: Three approaches for mining definitions from relational data in the web of data. In: Proceedings of the 6th International Workshop FCA4AI (IJCAI/ECAI). pp. 21–32 (2018)
13. Rizzo, G., d'Amato, C., Fanizzi, N., Esposito, F.: Terminological Cluster Trees for Disjointness Axiom Discovery. In: Proceedings of ESWC. pp. 184–201 (2017)
14. Rizzo, G., Fanizzi, N., d'Amato, C., Esposito, F.: Approximate classification with web ontologies through evidential terminological trees and forests. Int. J. Approx. Reasoning **92**, 340–362 (2018)
15. Völker, J., Fleischhacker, D., Stuckenschmidt, H.: Automatic acquisition of class disjointness. J. Web Sem. **35**, 124–139 (2015)
16. Völker, J., Niepert, M.: Statistical schema induction. In: Extended Semantic Web Conference. pp. 124–138 (2011)