Trois approches pour classifier les données du web des données

Justine Reynaud Yannick Toussaint Amedeo Napoli

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

prénom.nom@loria.fr

Résumé

Dans cet article, nous nous intéressons au processus de classification de données relationnelles issues du web des données. Nous disposons d'un ensemble d'objets entre lesquels il existe des relations. Ces objets appartiennent à une ou plusieurs classes. Celles-ci sont définies en extension*, et nous cherchons à construire une description en* intension *en s'appuyant sur les relations des objets qui les composent. Pour cela, nous employons trois approches : les règles d'association qui s'appuient sur l'Analyse de Concepts Formels (FCA), les redescriptions et les règles de traduction qui s'appuient sur la Longueur de Description Minimale (MDL). À partir d'expérimentations sur DBpedia, nous discutons les spécificités et la complémentarité de ces trois approches. Nous montrons que les règles d'association sont les plus exhaustives tandis que les règles de traduction ont une meilleure couverture des données. Les redescriptions pour leur part, sont les règles les plus faciles à appréhender et interpréter.*

Mots Clef

Données relationnelles, Analyse de Concepts Formels, Fouille de redescriptions, DBpedia.

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.

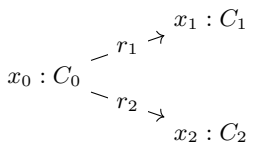
1 Introduction

Dans cet article, nous nous intéressons à la possibilité de découvrir des définitions dans les données RDF issues du web des données. Ces définitions peuvent être réutilisées dans la conception de bases de connaissances (KBs) ou dans l'enrichissement de KBs préexistantes. Étant donné l'immense quantité de données du web des données, il s'agit d'un enjeu majeur.

Le problème est le suivant : nous disposons d'un ensemble d'objets connectés par des relations, comme une ABox en logique de descriptions (DL) (Baader *et al.*, 2003). L'objectif est de classifier ces objets en fonction des relations dans lesquelles ils sont impliqués.

Les objets qui partagent des éléments communs appartiennent à une même classe. Ce partage peut être exact — les éléments sont identiques — ou approximatif — les éléments sont similaires. Finalement, nous obtenons un ensemble de classes organisées selon un ordre partiel, et les descriptions associées à ces classes. Ces descriptions sont nécessaires afin de construire les définitions des différentes classes. Les définitions sont considérées comme des conditions nécessaires (NC) et suffisantes (SC) pour classifier de nouveaux objets. Si $\mathtt{x}$ est une instance de la classe `Rouge` alors $\mathtt{x}$ a la couleur `rouge` (NC), et inversement, si $\mathtt{x}$ à la couleur `rouge` alors $\mathtt{x}$ est une instance de la classe `Rouge` (SC).

Pour poursuivre l'analogie avec les DLs, l'idée dans cet article est de construire et d'appliquer des règles d'induction de la forme « $\mathtt{si}\ \mathtt{r}(\mathtt{x},\mathtt{y})\ \mathtt{et}\ \mathtt{y}{:}\mathtt{C}\ \mathtt{alors}\ \mathtt{x}{:}\exists\mathtt{r}.\mathtt{C}$ ». Cela signifie que, étant données $\mathtt{y}$ une instance de la classe $\mathtt{C}$ et $\mathtt{r}$ une relation telle que $\mathtt{r}(\mathtt{x},\mathtt{y})$, alors $\mathtt{x}$ appartient à une classe, disons $\mathtt{D}$, dont la description comprend $\exists\mathtt{r}.\mathtt{C}$. C'est-à-dire que les instances de $\mathtt{D}$ sont reliées à au moins une instance de $\mathtt{C}$ par la relation $\mathtt{r}$. Nous utilisons ce type de règles pour construire les définitions des classes. Ces définitions sont de la forme $\mathtt{C_i} \equiv \mathtt{e_1} \sqcap \mathtt{e_2} \sqcap \ldots \sqcap \mathtt{e_n}$ où $\mathtt{e_j}$ est une expression de la forme $\exists\mathtt{r}.\mathtt{C_j}$. Notre travail se divise en trois tâches principales, à savoir (i) préparer les données, (ii) découvrir les définitions, (iii) évaluer la qualité de ces



ex : x_0 ex : r_1 ex : x_1.
ex : x_0 ex : r_2 ex : x_2.
ex : x_0 rdf : type ex : C_0.
ex : x_1 rdf : type ex : C_1.
ex : x_2 rdf : type ex : C_2.

FIGURE 1 – Exemple de données relationnelles et les triplets associés.

définitions. Nous nous appuyons ici sur trois approches, les règles d'association, la fouille de redescriptions et la découverte de règles de traduction.

Cet article est dans la continuité de travaux de recherche sur la découverte de définitions dans le web des données. Son originalité est de comparer trois approches qui ne s'appuient pas sur les mêmes principes mais qui s'avèrent être complémentaires au vu des résultats de nos expérimentations. À notre connaissance, il s'agit du premier article où une telle comparaison est proposée à la fois à un niveau théorique et à un niveau expérimental.

L'article est organisé comme suit : dans la section 2, nous présentons la nature des données sur lesquelles nous travaillons et les processus de classification dans le web des données. Dans la section 3, nous détaillons les trois approches de classification et leurs applications. La section 5 présente les expériences qui ont été menées ainsi que l'évaluation des règles obtenues. Enfin, la section 6 présente une discussion ainsi que les pistes de recherches futures avant la conclusion section 7.

2 Représentation des données

2.1 Web des Données

Dans cette section, nous présentons les données issues du Web des Données (LOD) que nous considérons. Le LOD peut être vu comme un ensemble de KBs interconnectées. Une KB est composée de deux éléments : une TBox qui définit son *schema* et une ABox qui introduit les individus et leurs relations. L'unité de base d'une KB est le triplet RDF, noté $\langle s, p, o\rangle$, qui encode une assertion sous la forme sujet–prédicat–objet. Les différentes composantes d'un triplet peuvent être des *ressources* U identifiées de manière unique, des *littéraux* L (chaîne de caractères, numérique, date, ...) ou des *nœuds anonymes* B (assimilables à une variable existentiellement quantifiée), de telle sorte que $\langle s, p, o\rangle \in (B \cup U) \times U \times (B \cup U \cup L)$. Dans cet article, nous nous restreignons aux triplets composés uniquement de ressources, c'est-à-dire tels que $\langle s, p, o\rangle \in U \times U \times U$. Les ressources peuvent faire référence à n'importe quel objet ou abstraction et sont identifiées par une URI (*Uniform Resource Identifier*). Une URI est une adresse composée de deux parties. La première partie est l'espace de nom (*namespace*) qui indique de quelle KB viennent les ressources. La seconde partie donne un nom à la ressource au sein de cette KB.

La figure 1, présente un exemple de données relationnelles et l'ensemble de triplets correspondant. Le préfixe `ex:` correspond à un *namespace* créé pour notre exemple, tandis que le préfix `rdf:` correspond à un *namespace* préexistant.`rdf:type` est la relation d'instanciation. Ainsi le triplet $\langle$`ex:`x_0 `rdf:type` `ex` : $C_0\rangle$ indique que x_0 est une instance de la classe C_0.

Le LOD peut être interrogé grâce aux requêtes SPARQL. Par exemple, la requête `SELECT ?x WHERE { ?x rdf:type ex:`C_0`}` retourne toutes les instances de C_0. Si l'on prend les données de la figure 1, seul `ex:`x_0 est retourné.

2.2 Analyse de Concepts Formels

Nous utilisons ici l'Analyse de Concepts Formels (FCA) de Ganter et Wille (1999) pour présenter et comparer les différentes approches. Étant donné un ensemble G d'entités [1], un ensemble M d'attributs et une relation binaire $I \subseteq G \times M$, (G, M, I) est un contexte formel. L'expression gIm s'interprète comme « l'entité g possède l'attribut m». Les correspondances de Galois (notées $.'$) pour un ensemble d'entités $X \subseteq G$ et un ensemble d'attributs $Y \subseteq M$ sont définies comme suit :

$$X' = \{m \in M \mid \forall x \in X, xIm\} \quad \text{et}$$
$$Y' = \{g \in G \mid \forall y \in Y, gIy\}.$$

À partir des données RDF, nous construisons un contexte formel dont les entités sont les sujets des triplets. Les attributs sont les paires (prédicat,objet) issues des triplets. Nous distinguons deux types d'attributs : des descriptions et des classes. Le premier ensemble d'attributs, dénoté $\mathcal{C}$, est composée des paires de la forme (`rdf:type`, `C`), qui font d'un sujet une instance de la classe `C`. Ce sont ces classes que nous cherchons à définir. Le second ensemble, dénoté $\mathcal{D}$ est composé de paires (`p`, `o`) telles que `p` $\neq$ `rdf:type`. Les attributs du contexte sont donc $M = \mathcal{C} \cup \mathcal{D}$. Si l'on prend l'exemple de la figure 1, la figure **??** présente le contexte associé.

		Attributs				
		Descriptions		Classes		
		$\exists r1{:}C_1$	$\exists r2{:}C_2$	C_0	C_1	C_2
	x_0	×	×	×		
Objets	x_1				×	
	x_2					×

FIGURE 2 – Contexte formel associé aux données représentées figure 1.

À partir de là, il s'agit de trouver un ensemble de catégories et un ensemble de descriptions de manière à ce que leurs correspondances soient les mêmes. Sur la figure **??** par exemple, on a $\{C_0\}' = x_0$ et $\{\exists r_1{:}C_1, \exists r_2{:}C_2\}' = x_0$. On peut alors construire la définition suivante :

$$C_0 \equiv \exists r_1{:}C_1 \sqcap \exists r_2{:}C_2$$

1. En FCA, G est l'ensemble des *objets*, renommés *entités* afin de ne pas confondre avec les objets en RDF.

Comme les données peuvent être incomplètes, il se peut qu'on ne retrouve pas une égalité entre une classe $\texttt{C}_\texttt{i}$ et sa description $\exists\texttt{r:C}_\texttt{j}$. Autrement dit $\{\texttt{C}_\texttt{i}\}' \neq \{\exists\texttt{r:C}_\texttt{j}\}'$. Il nous faut donc des approches qui tolèrent une forme d'approximation. C'est ce que permettent les trois algorithmes utilisés, et nous allons les détailler plus précisément dans la section suivante.

3 Algorithmes de fouille de règles

3.1 Règles d'association

Le but de la fouille de règles d'association (Agrawal *et al.*, 1993; Klemettinen *et al.*, 1994) est de trouver des dépendances entre les attributs. Une règle d'association entre deux ensembles d'attributs X et Y, notée $X \to Y$, signifie « si un objet a tous les attributs de X, alors il a tous les attributs de Y ». À cette règle est associée une *confiance*, qui peut être vue comme une probabilité conditionnelle :

$$\text{conf}(X \to Y) = \frac{|X' \cap Y'|}{|X'|},$$

où $.'$ correspond à la correspondance de Galois. La confiance est une mesure de qualité des règles d'association. Une règle d'association est valide si sa confiance est supérieure à un seuil θ défini par l'utilisateur. La confiance n'est pas symétrique : $X \to Y$ peut être valide sans que $Y \to X$ le soit. Si la confiance vaut 1, on dit qu'il s'agit d'une *implication* et l'on note $X \Rightarrow Y$. Si on a également $Y \Rightarrow X$, alors X et Y forment une définition, notée $X \equiv Y$.
Nous nous intéressons ici à des définitions. Nous considérons donc conjointement $X \to Y$ et sa réciproque $Y \to X$ et cherchons à estimer à quel point ces règles s'approchent d'une implication. Pour cela, nous introduisons la notion de *quasi-définition* qui est à la définition ce que la règle d'association est à l'implication.

Définition 1 (Quasi-définition)**.** Étant donné deux ensembles d'attributs X et Y, une quasi-définition $X \leftrightarrow Y$ est valide si, pour un seuil θ donné,

$$\min(\text{conf}(X \to Y), \text{conf}(Y \to X)) \geqslant \theta.$$

L'algorithme `Eclat` (Zaki, 2000) est l'un des nombreux algorithmes de fouille de règles d'association existant. Il énumère de manière exhaustive toutes les règles d'association valides pour un seuil donné. Nous utilisons cet algorithme pour notre comparaison.
Une règle d'association $r_0 : x_1, \ldots, x_n \to y_1, \ldots, y_m$ peut se décomposer sous la forme de plusieurs règles d'association $r_i : x_1, \ldots, x_n \to y_i$ pour $i \in \{1, \ldots, m\}$. Si r_0 est valide, l'ensemble des r_i est valide. Pour que les règles obtenues définissent des classes comme nous le souhaitons, il faut que les règles d'association soient de la forme «Classes $\to$ Descriptions» ou «Descriptions $\to$ Classes». C'est à dire que la règle $X \to Y$ est telle que $X \subseteq \mathcal{C}, Y \subseteq \mathcal{D}$ ou $X \subseteq \mathcal{D}, Y \subseteq \mathcal{C}$. Il nous faut donc utiliser un post-traitement afin de s'assurer que la règle ne contient que des catégories d'un côté et que des descriptions de l'autre. Pour cela, seules les règles $X \to Y$ qui satisfont l'une des deux contraintes alternatives suivantes sont conservées : (i) l'antécédent ne contient que des classes ($X \subseteq \mathcal{C}$) et il y a au moins une description dans la conséquence ($Y \cap \mathcal{D} \neq \emptyset$) ; (ii) l'antécédent ne contient que des descriptions ($X \subseteq \mathcal{D}$) et il y a au moins une classe dans la conséquence ($Y \cap \mathcal{C} \neq \emptyset$). Les règles conservées sont décomposées de manière à ne garder que des classes (resp. des descriptions) dans la conséquence si l'antécédent ne contient que des descriptions (resp. des classes).
Par exemple, $\exists r_1{:}C_1, C_0 \to \exists r2{:}C_2$ n'est pas conservée parce que l'antécédent contient à la fois une catégorie et une description. En revanche, la règle $\exists r_1{:}C_1 \to \exists r_2{:}C_2, C_0$ peut être décomposée en deux règles r_1: $\exists r_1{:}C_1 \to \exists r_2{:}C_2$ et r_2: $\exists r_1{:}C_1 \to C_0$. La règle r_2 est conservée. Si sa réciproque est valide, la quasi-définition obtenue est $C_0 \leftrightarrow \exists r_1{:}C_1$.

3.2 Redescriptions

La fouille de redescriptions, introduite par Ramakrishnan *et al.* (2004), a pour but de trouver des caractérisations multiples d'un même ensemble d'entités. À la différence des règles d'association, les redescriptions s'appuient sur une séparation de l'ensemble des attributs en *vues*. Une vue est un sous-ensemble d'attributs. L'ensemble des vues forme une partition des attributs. Nous utilisons ici deux vues, qui correspondent aux deux types d'attributs — classes et descriptions.
La similarité entre deux ensembles d'attributs, provenant de deux vues différentes, est mesurée avec le coefficient de Jaccard :

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

où $.'$ correspond à la correspondance de Galois. Une paire d'ensembles d'attributs (A, B) est une redescription si le coefficient de Jaccard $\text{jacc}(A, B)$ est supérieur à un seuil donné. Le coefficient de Jaccard est symétrique, contrairement à la confiance. Une redescription dont le coefficient de Jaccard vaut 1 est une définition. Toutes les redescriptions sont nécessairement des quasi-définitions. En effet,

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

L'algorithme `ReReMi` (Galbrun et Miettinen, 2012) est utilisé ici. En plus des données binaires, `ReReMi` permet de tenir compte de données numériques et catégorielles. Les redescriptions obtenues peuvent être des fonctions booléennes contenant des disjonctions et des négations d'attributs. Afin de comparer les trois algorithmes, `ReReMi` est ici restreint à des conjonctions d'attributs, raison pour laquelle nous ne parlons pas de fonctions booléennes.
Dans un premier temps, `ReReMi` cherche des paires d'attributs — un attribut de chaque vue — susceptibles de prendre part à une définition, c'est-à-dire celles qui ont le coefficient de Jaccard le plus élevé. Ces paires sont ensuite

étendues tour à tour, en ajoutant à chaque fois un attribut à l'un des côtés de la redescription, jusqu'à ce que le nombre d'attributs maximum soit atteint ou que le coefficient de Jaccard n'augmente plus. Les redescriptions ainsi obtenues qui satisfont les critères de sélection sont retournées à l'utilisateur.

3.3 Règles de traduction

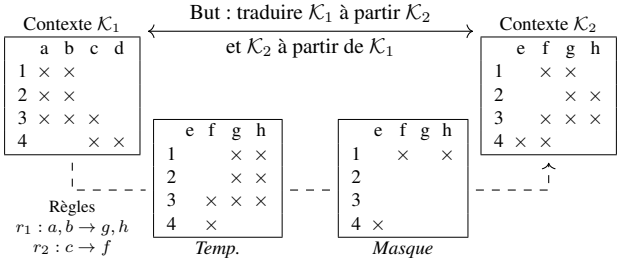


FIGURE 3 – Intuition du fonctionnement de `Translator`. Ici, seule la traduction de $\mathcal{K}_1$ vers $\mathcal{K}_2$ est représentée. À chaque étape, le but est de trouver la règle qui va minimiser le nombre de croix dans le masque.

L'algorithme `Translator` (van Leeuwen et Galbrun, 2015) s'appuie également sur une séparation du contexte en deux vues, et cherche un ensemble d'associations entre ces deux vues. Ces associations ont la même forme que des règles d'association, la différence se situant au niveau de la constitution de l'ensemble de règles.
Cet ensemble doit être compact et représentatif. D'une part, il doit couvrir la majorité des données. D'autre part, les règles doivent être aussi petites que possible en terme d'attributs. Afin de trouver un équilibre entre ces deux contraintes, `Translator` s'appuie sur le concept de Longueur de Description Minimum (MDL). Étant donné un contexte $\mathcal{K}$ et $X \subseteq M$ un ensemble d'attributs, la *longueur* de X est

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

Cette longueur correspond au nombre minimal de bits requis pour encoder X.
Pour construire l'ensemble des règles, les auteurs font l'analogie avec la notion de traduction. Une règle est une traduction d'une vue vers une autre. L'idée sous-jacente est la suivante : on souhaite construire un ensemble de règles qui permettent, connaissant une des deux vues, de reconstruire la seconde et *vice versa*. L'idée générale est représentée figure 2. Les erreurs entre le contexte cible et le contexte reconstruit sont corrigées à l'aide d'un masque. La taille de ce masque indique donc le nombre d'erreurs introduites.
L'ensemble de règles est construit de manière itérative, en prenant, à chaque étape, la règle $X \to Y$ qui maximise Δ

$$\Delta(X \to Y) = \underbrace{L(\mathit{Mask}^-) - L(\mathit{Mask}^+)}_{\text{Gain d'information}} - \underbrace{L(X \to Y)}_{\text{Longueur de la règle}}$$

où Mask^+ correspond aux éléments ajoutés au masque (erreurs introduites par la règle) et Mask^- correspond aux éléments retirés du masque (erreurs corrigées par la règle). Des règles sont ajoutées tant que Δ est positif.
L'algorithme `Translator` est le seul dont la mesure de qualité tient compte des règles déjà extraites. Le masque étant mis à jour à chaque étape, la qualité d'une règle dépend des règles déjà extraites. Ainsi, `Translator` favorise un *bon ensemble de règles* plutôt qu'un *ensemble de bonnes règles*. Chaque règle apporte une information qui n'est pas contenue dans les autres, ce qui limite la redondance d'information.

4 Travaux connexes

La FCA est un outil de conceptualisation qui permet de structurer une ontologie par une approche *bottom-up*. Sertkaya (2011) fait une synthèse des différentes contributions qui s'intéressent au lien entre la FCA et les ontologies. La plupart des ontologies sont construites avec une approche *top-down*, c'est à dire en construisant d'abord le schéma. L'une des façons de tirer parti de la FCA est donc de consuire des connaissances qui vont enrichir une ontologie préexistante. C'est ce que nous faisons dans notre approche, en partant de triplets pour trouver des définitions.
Dans (Ferré et Cellier, 2016), les auteurs proposent une extension de la FCA pour les graphes conceptuels. Contrairement aux graphes RDF qui reposent sur des relations binaires, les graphes conceptuels permettent de tenir compte de relation n-aires. Les concepts construits correspondent à une association entre une requête SPARQL et l'ensemble des solutions. Dans notre approche, nous cherchons à mettre en relation deux *graph patterns* et nous ne connaissons pas *a priori* la requête SPARQL qui correspond à la description d'une catégorie.
Notre travail poursuit un travail initié dans (Alam *et al.*, 2015). Les auteurs s'appuient sur la fouille de règles d'association pour fournir un espace de navigation sur des données RDF. Pour cela, les auteurs recherchent des implications et les ordonnent en fonction de la confiance de leur réciproque. Dans notre approche, nous introduisons la séparation des attributs et nous comparons les règles d'association avec les redescriptions et les règles de traduction.
L'algorithme AMIE, proposé par Galárraga *et al.* (2013), est une référence en matière de fouille de règles sur le web des données. Ces règles sont de la forme $B_1, B_2, \ldots B_{n-1} \to B_n$ où B_i correspond à une relation entre deux variables, que l'on peut exprimer sous forme de triplet $<?x$ `r` $?y>$. Par exemple, $manufacturer(x, Samsung), \quad successor(y, x) \quad \to manufacturer(y, Samsung)$ est une règle qui peut être trouvée par AMIE. Les auteurs ajoutent une condition, à savoir que toutes les variables doivent apparaître deux fois dans la règle. Si l'on considère les triplets sous forme de graphe, il en résulte qu'une règle est un motif cyclique entre des variables (indépendamment de l'orientation des arcs). Dans notre cas, les motifs recherchés

caractérisent une seule ressource identifiée (motif en étoile) et l'association est bidirectionnelle. On trouve par exemple la règle $Samsung\_Mobile\_Phone(x) \leftrightarrow manufacturer(x, Samsung)$.

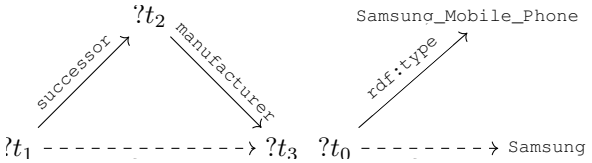


FIGURE 4 – Motifs recherchés dans la base de connaissances par AMIE et par notre approche. Les traits pleins concernent la partie gauche de la règle, et les pointillés la partie droite. Les $?t_i$ sont des variables existentielles.

5 Expérimentations

Nous avons réalisé nos expérimentations sur des données issues de *DBpedia*, qui est une KB construite à partir de *Wikipédia*. Nous cherchons à définir des *catégories*, c'est-à-dire les ressources qui sont dans le co-domaine de la relation `dct:subject`. Pour cela, nous extrayons un sous-ensemble de *DBpedia* à l'aide d'une requête SPARQL avant de transformer les triplets obtenus en contexte comme mentionné en Section 2.2. Enfin, nous utilisons les trois algorithmes présentés pour comparer et évaluer les résultats obtenus.

TABLE 1 – Statistiques sur les jeux de données extraits.

	Triplets	Objets	Attributs	
			Cat.	Descr.
`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 Méthodologie

Nous avons extrait quatre sous-ensembles de triplets de *DBpedia*, de différents domaines [2]. Afin d'isoler un sous-ensemble de triplets, nous avons récupéré tous les triplets dont le sujet est lié à une catégorie donnée. Pour cela, nous avons utilisé la requête SPARQL suivante :

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

Les quatres domaines utilisés (`X`) sont `French_films`, `Turing_Award_laureates`, `Smartphones` et `Sports_cars`. Le triplet `?p a owl:objectProperty` assure que $?o$ n'est pas un littéral.

Les triplets extraits sont divisés en deux groupes. L'ensemble des triplets qui ont pour prédicat `dct:subject` correspond aux attributs de classe ($\mathcal{C}$). L'ensemble des triplets dont le prédicat n'est pas `dct:subject` correspond aux attributs de description ($\mathcal{D}$).

2. Les jeux de données ainsi que les résultats obtenus sont disponibles à l'adresse `https://gitlab.inria.fr/jreynaud/DefinitionMiningComparison`.

5.2 Résultats et évaluation

TABLE 2 – Évaluation des résultats pour chaque jeu de données. $\overline{|C_i|}$ (resp. $\overline{|D_i|}$) désigne le nombre moyen de catégories (resp. de descriptions) dans une règle.

`Turing_Award_laureates`

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

`Sports_cars`

X	`Eclat`	`ReReMi`	`Translator`
$\lvert\mathcal{B}^X_{cand}\rvert$	132	52	31
$\lvert\mathcal{B}^X_{def}\rvert$	95	30	23
Précision	.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

`Smartphones`

X	`Eclat`	`ReReMi`	`Translator`
$\lvert\mathcal{B}^X_{cand}\rvert$	810	98	41
$\lvert\mathcal{B}^X_{def}\rvert$	521	57	31
Précision	.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

`French_films`

X	`Eclat`	`ReReMi`	`Translator`
$\lvert\mathcal{B}^X_{cand}\rvert$	546	36	93
$\lvert\mathcal{B}^X_{def}\rvert$	371	12	89
Précision	.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

Chaque algorithme retourne un ensemble ordonné de quasi-définitions de la forme $C_0, \ldots, C_n \leftrightarrow D_0, \ldots, D_m$. Chacune des quasi-définitions est évaluée manuellement par trois personnes jouant le rôle d'experts. Étant donnée une quasi-définition $C_0, \ldots, C_n \leftrightarrow D_0, \ldots, D_m$ d'un jeu de données J, l'évaluateur répond à la question « Étant donné que nous parlons de *J*, est il correct de dire que *faire partie à la fois de* C_0*, de ... et de* C_n et *avoir les propriétés* $D_0, \ldots$*et* D_m est équivalent ? »

L'évaluation finale est l'évaluation majoritaire des trois experts. Si la règle est acceptée, elle rejoint l'ensemble des

définitions obtenues, dont 20 exemples sont présentés figure 4. Les experts ont été du même avis dans 95.4% des cas. La règle R5 de la figure 4 est une règle qui n'a pas fait l'unanimité.

Les algorithmes sont comparés sur la base des définitions extraites et des catégories définies. La figure 5 présente un diagramme de Venn pour chaque corpus, qui représente le nombre de définitions extraites par chaque algorithme. Par exemple, pour le corpus `Turing_Award_laureates`, il y a 22 définitions trouvées uniquement par `Eclat`, et 8 trouvées à la fois par `Eclat` et `Translator`. Au total, `Eclat` a extrait 30 définitions. La figure 6 présente également un diagramme de Venn pour chaque corpus, qui correspond au nombre de catégories définies par chaque algorithme. Une catégorie est considérée comme définie dès lors qu'elle apparaît dans au moins une définition. Il peut donc y avoir une ou plusieurs catégories considérées comme définies pour une seule définition. C'est notament le cas des règles R2, R8 – R10, R12, R14, R17 – R20 de la figure 4.

6 Discussion

Ci-dessous, $\mathcal{B}_{cand}[X]$ désigne l'ensemble de toutes les quasi-définitions extraites par l'algorithme X et $\mathcal{B}_{def}[X]$ l'ensemble des définitions de X, c'est-à-dire, l'ensemble des quasi-définitions de $\mathcal{B}_{cand}[X]$ évaluées vraies par les experts. L'ensemble $\mathcal{B}_{cand}$ désigne la totalité des quasi-définitions,indépendamment de l'algorithme qui les a extraites. De la même façon, $\mathcal{B}_{def}$ désigne l'ensemble de toutes les définitions extraites.

6.1 Précision et rappel

La précision d'un algorithme X est $\frac{|\mathcal{B}^X_{def}|}{|\mathcal{B}^X_{cand}|}$. La précision de `ReReMi` a une très forte variabilité (entre 33 et 75%) et est globalement la plus faible, tout particulièrement sur le jeu de données `French_films`. La précision d'`Eclat` est stable (entre 64 et 72%). La meilleure précision est obtenue par `Translator` : quel que soit le jeu de données, elle toujours supérieure à 74%.

Le rappel, défini comme $\frac{|\mathcal{B}^X_{def}|}{|\mathcal{B}_{def}|}$ ne peut pas être utilisé comme une mesure de performance. En effet, certaines règles se recouvrent (ont des attributs en commun des deux côtés). C'est le cas des règles R6 à R10 (figure 4) qui définissent toutes la catégorie `McLaren_vehicles`. Tandis que `Translator` n'extrait qu'une seule règle (R8), `ReReMi` extrait 2 règles (R6 et R7) et `Eclat` en extrait 9 (seules R8 à R10 sont reportées ici).

La table **??** indique le nombre de catégories pour chaque jeu de données. Si l'on se fie à ces valeurs, le rappel est très faible : dans le corpus `Turing_Award_laureates` par exemple, pour 503 catégories dans les données de départ, seules 19 font partie d'une règle. Cela s'explique principalement par la très faible densité des contextes ; une grande quantité de catégories ne concerne qu'une seule entité du jeu de données. Si l'on ne compte que les catégories qui ont un support d'au moins 3, elles ne sont plus que 105, et 47 catégories seulement on un support d'au moins 5. Nous considérons donc le rappel non pas par rapport au nombre de catégories présentes dans le jeu de données, mais par rapport aux catégories retrouvées par l'ensemble des algorithmes, comme présenté figure 6.

6.2 Forme et interprétation des règles

D'après la figure 6, 70% des catégories définies par `Eclat` ou `Translator` sont définies par les deux algorithmes. `Translator` extrait considérablement moins de règles que `Eclat` (jusqu'à 16 fois moins pour le corpus `Smartphones`). Cela s'explique par la façon dont sont générées les règles d'association. Si dans les données, la règle $A \rightarrow B$ a le même support que la règle $A \rightarrow B, C$, alors seule la règle $A \rightarrow B, C$ est conservée. En revanche si le support de $A \rightarrow B$ est strictement supérieur au support de $A \rightarrow B, C$, `Eclat` génère deux règles. En conséquence, on obtient avec `Eclat` des règles qui ne diffèrent que d'un attribut, comme R9 et R10 par exemple, alors que `Translator` ne génère qu'une seule règle, R8 dans l'exemple.

`ReReMi` n'extrait aucune définition en commun avec `Eclat` et `Translator`. Cette différence s'explique par l'heuristique employée par `ReReMi`. Si C est une catégorie, et que D_1 et D_2 sont deux descriptions telles que $C' = D'_1 = D'_2$, alors `ReReMi` privilégie deux définitions $C \equiv D_1$ et $C \equiv D_2$ tandis que `Eclat` ne génère qu'une seule définition $C \equiv D_1 \sqcap D_2$. C'est par exemple le cas des définitions R6, R7 générées par `ReReMi` et R8, générée par `Eclat` (figure 4). Si $C' = D'_1$ et $D'_1 \subset D'_2$, `ReReMi` génère la définition $C \equiv D_1$, tandis que `Eclat` génère la définition $C \equiv D_1 \sqcap D_2$. Ce cas a également été retrouvé dans nos résultats, comme le montrent les définitions R12 et R13.

La taille des définitions générées par `ReReMi` est inférieure à celle des définitions de `Translator` et `Eclat`. Cela s'explique par l'heuristique de `ReReMi` déaillée dans le paragraphe précédent. `ReReMi` a en moyenne entre 1 et 2 attributs de chaque côté de la définition tandis que pour `Eclat` et `Translator`, il y a en moyenne 3 catégories et 4 descriptions par définition.

Les conjonctions dans les définitions extraites par `ReReMi` n'ont pas le même sens que celles extraites par `Eclat` et `Translator`. Par exemple, si l'on considère la définition R15, l'attribut `(a Device)` peut être enlevé sans altérer la définition : parce que toutes les entités considérées sont des instances de `Device`. À l'inverse, dans la définition R14, aucun attribut ne peut être retiré sans que la définition ne devienne fausse : tous les attributs font partie de la condition nécessaire. Dans notre approche, il nous semble plus pertinent de n'intégrer que des attributs qui contribuent pleinement à la définition dont ils font partie. Ainsi, la définition R14 nous parait préférable à la définition R15, parce qu'elle est apparue plus facile à interpréter.

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

FIGURE 5 – Exemples de quasi-définitions obtenues par `Eclat`, `ReReMi` et `Translator` pour chaque corpus. Lorsque le préfixe d'un attribut de droite n'est pas spécifié, il s'agit de `dbo`. Le préfixe des catégories (`dbc`) n'est pas spécifié.

7 Conclusion

Dans cet article, nous nous sommes intéressés à trois algorithmes pour trouver des définitions de classes dans le web des données. Nous avons vu que chaque algorithme avait ses spécificités et nous avons vérifié empiriquement que les résultats extraits reflètent ces spécificités. Nous avons aussi montré que malgré des approches très différentes, les algorithmes `Eclat` et `Translator` extraient de nombreuses règles communes. À l'inverse, `ReReMi`, malgré une mesure de qualité semblable à `Eclat`, extrait des règles plus courtes. L'intérêt de ces algorithmes dépend de l'objectif de l'utilisateur. Dans le cas de notre expérimentation, `Eclat` est l'algorithme qui a qualifié le plus de catégories, au prix d'un nombre de règles extraites très important. `Translator` extrait significativement moins de règles avec un nombre de catégories définies légèrement inférieur. Enfin, `ReReMi`, malgré un nombre de catégories définies plus faible, offre des définitions plus simples en n'incluant pas les attributs qui ne participent pas pleinement à la règle. Par la suite, plusieurs directions de recherche sont envisagées, au niveau de la préparation des données, de la fouille et de l'évaluation.

Comme mentionné précédemment, les contextes construits à partir des données RDF sont très creux. Dans le cas d'`Eclat` et de `ReReMi`, un seuil de support étant utilisé, l'existence d'attributs à très faible support n'a pas d'impact sur les règles retrouvées. Pour `Translator` en revanche, les règles doivent permettre de reconstruire intégralement le contexte. Dans ce cas, il peut être pertinent de simplifier le contexte avant de procéder à la fouille de règles. Cela peut notamment passer par la suppression des attributs à très faible support, ou au contraire, des attributs ayant un support quasi maximal, et donc qui concernent toutes les entités. Nous pourrons également travailler sur le type des données en entrée, en intégrant notamment des valeurs numériques, ce qui nous permettra de prendre en compte des triplets laissés de côté pour cette expérimentation, tels que les informations de date, d'âge, de taille, ...

Pour améliorer le processus de fouille, il est possible d'ajouter des contraintes sur les règles obtenues afin de restreindre l'espace de recherche, mais aussi d'obtenir des règles plus faciles à interpréter. Dans notre cas, n'autoriser qu'une catégorie par règle serait une contrainte pertinente. Nous pouvons également étendre l'expressivité des règles recherchées en permettant par exemple des opérateurs logiques tels que les disjonctions ou les négations. Si `ReReMi` permet déjà d'obtenir de telles formules lo-

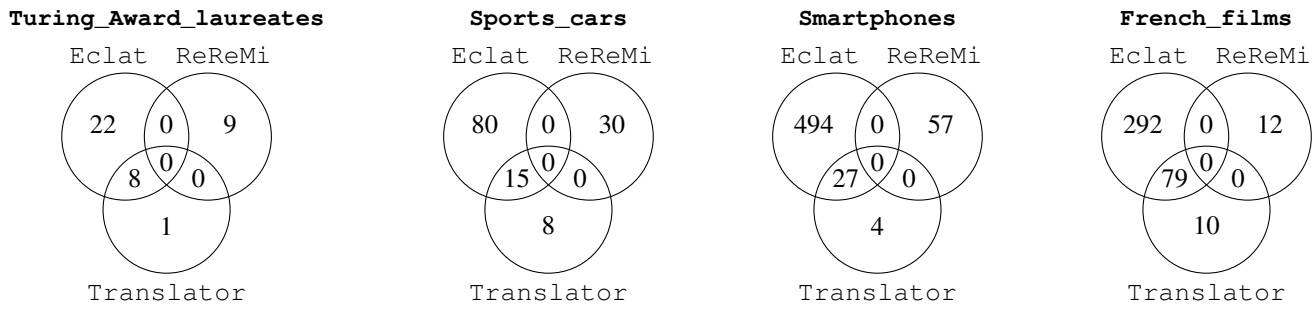


FIGURE 6 – Règles extraites par `Eclat`, `ReReMi` et `Translator` pour chaque jeu de données.

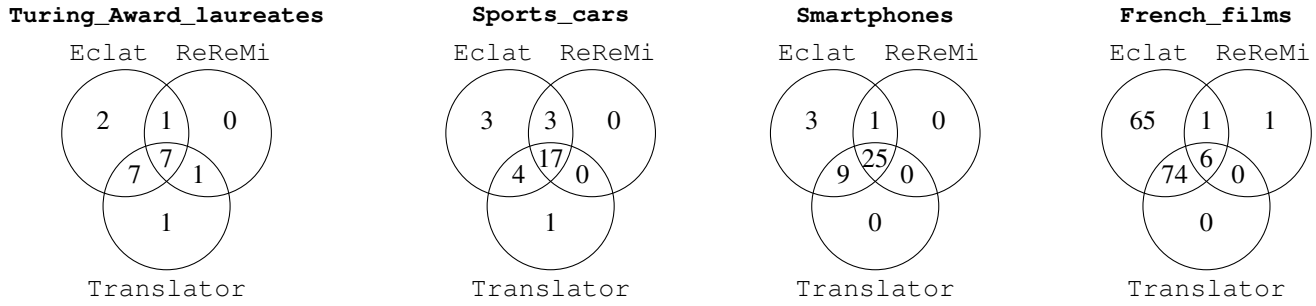


FIGURE 7 – Catégories définies par `Eclat`, `ReReMi` et `Translator` pour chaque jeu de données.

giques, et que des travaux dans ce sens existent pour les règles d'association, ce n'est pas le cas pour les règles de traduction. Cela implique de revoir le calcul de la longueur d'une règle.

Concernant l'évaluation, il nous faudra un moyen de comparer plus formellement les règles entre elles, peut-être *via* l'introduction d'une notion de redondance, qui permettrait de définir une base de règles.

Remerciements

Ce travail est financé avec le support de la Direction Générale de l'Armement et de la Région Lorraine. Merci à Esther Galbrun pour ses conseils ainsi que Quentin Brabant et Jérémie Nevin pour leur aide à l'évaluation des règles.

Références

R. AGRAWAL, T. IMIELIŃSKI et A. SWAMI : Mining association rules between sets of items in large databases. *In ACM SIGMOD Rec.*, vol. 22, p. 207–216. ACM, 1993.

M. ALAM, A. BUZMAKOV, V. CODOCEDO et A. NAPOLI : Mining definitions from rdf annotations using formal concept analysis. *In IJCAI*, p. 823–829, 2015.

F. BAADER, D. CALVANESE, D. MCGUINNESS, D. NARDI et P. PATEL-SCHNEIDER, éds. *The Description Logic Handbook*. Cambridge University Press, Cambridge, UK, 2003.

S. FERRÉ et P. CELLIER : Graph-FCA in Practice. *In Proceedings of 22nd ICCS*, p. 107–121, 2016.

L. A. GALÁRRAGA, C. TEFLIOUDI, K. HOSE et F. M. SUCHANEK : AMIE : association rule mining under incomplete evidence in ontological knowledge bases. *In WWW'13*, p. 413–422, 2013.

E. GALBRUN et 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.

B. GANTER et R. WILLE : *Formal concept analysis - mathematical foundations*. Springer, 1999.

M. KLEMETTINEN, H. MANNILA, P. RONKAINEN, H. TOIVONEN et A. I. VERKAMO : Finding interesting rules from large sets of discovered association rules. *In CIKM'94*, p. 401–407, 1994.

N. RAMAKRISHNAN, D. KUMAR, B. MISHRA, M. POTTS et R. F. HELM : Turning CARTwheels : an Alternating Algorithm for Mining Redescriptions. *In KDD'04*, p. 266–275, 2004.

B. SERTKAYA : A survey on how description logic ontologies benefit from formal concept analysis. *CoRR*, abs/1107.2822, 2011.

M. van LEEUWEN et E. GALBRUN : Association Discovery in Two-View Data. *TKDE*, 27(12):3190–3202, déc. 2015.

M. J. ZAKI : Scalable algorithms for association mining. *TKDE*, 12(3):372–390, May 2000.