Découverte de définitions dans le web des données

Justine Reynaud

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Découverte de définitions dans le web des données

THÈSE

présentée et soutenue publiquement le 10 décembre 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Justine Reynaud

Composition du jury

Présidente :	Claire Gardent	DR CNRS, LORIA, Nancy
Rapporteurs :	Catherine Faron-Zucker	MCf HDR, Université de Nice Sophia Antipolis
	Fatiha Saïs	MCf HDR, Université Paris Sud
Examinateur :	Luis Galàrraga	CR INRIA, IRISA Rennes
Invitée :	Nathalie Pernelle	MCf HDR, Université Paris Sud
Directeurs :	Amedeo Napoli	DR CNRS, LORIA, Nancy
	Yannick Toussaint	Pr. Université de Lorraine, Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je tiens en premier lieu à remercier mes encadrants de thèse, Amedeo Napoli et Yannick Toussaint, pour le temps qu'ils m'ont consacré pendant ces quatre années. Ils ont su me guider dans mes recherches et me canaliser quand cela était nécessaire. Cette thèse n'aurait pas pu être ce qu'elle est sans leur aide précieuse. Dans un second temps je remercie Catherie Faron-Zucker et Fathia Saïs d'avoir accepté d'être les rapporteures de cette thèse et de m'avoir fait des retours pertinents et constructifs. Je remercie également Luis Galàrraga et Claire Gardent de leur présence en tant qu'examinateurs, ainsi que Natalie Pernelle pour avoir accepté l'invitation à participer à mon jury.

Mon aventure dans l'équipe Orpailleur a débuté bien avant ma thèse. Je tiens donc à remercier ici Jean Lieber de m'avoir fait découvrir le monde (fermé/ouvert) des logiques de descriptions (et des tartelettes) au cours de mon stage de master Informatique. Merci également à Chedy Raïssi pour m'avoir initié à la fouille de données au cours de mon stage de master de Sciences Cognitives, et de m'avoir donné envie de poursuivre en thèse par son enthousiasme et sa capacité à comprendre ce que je n'arrivais pas toujours à formuler.

Je tiens également à remercier toutes les personnes, actuelles ou anciennes, de l'équipe et plus généralement du laboratoire avec qui j'ai pu avoir des discussions enrichissantes. Je ne peux les citer toutes ici par manque de place, mais je leur fait confiance pour se reconnaître. Je tiens tout de même à remercier quelques personnes en particulier. Tout d'abord Esther Galbrun, sans qui cette thèse aurait probablement pris une toute autre tournure : Esther, merci à toi d'avoir débarqué à Nancy avec les redescriptions dans tes bagages ! Ensuite Laurine Huber, qui m'a notamment aidée à relire ce manuscrit, et Alexandre Bazin, qui m'a aidée à gérer le stress de pré-soutenance.

Merci également aux collègues avec qui j'ai enseigné, ce qui a souvent donné lieu à des discussions très enrichissantes, que ce soit autour de l'enseignement ou de la recherche. Merci donc, par ordre chronologique, à Armelle Brun, Anne Boyer, Viviane Vaillard, Fabien Pierre, Jean-Luc Husson, Marie Slaghuis, et toute l'équipe de St-Dié.

Merci aux personnes qui m'ont accompagné administrativement et matériellement. J'inclus dans ces personnes les assistantes d'équipes, et tout particulièrement Emmanuelle Deschamps ainsi que les secrétaires de l'école doctorale IAEM, Vanessa Binet et Souad Boutaguermouchet. Merci également à la région Grand-Est et à la Direction Générale de l'Armement d'avoir financé cette thèse.

Je tiens à remercier mes amis, qui ont accepté mes silences radio, et qui m'ont supportée dans les moments de doutes. Une fois encore, je ne peux citer tout le monde ici. Je tiens cependant à remercier en particulier Alexandrine Fuhrer et Cédric Georgel, qui m'ont été d'un immense soutient tout au long de cette thèse. Merci également à Sarah Blind. Nous étions ensemble en Sciences Cognitives, nous avons décidé ensemble de faire une thèse, et nous avons soutenu « ensemble ». Je n'aurais peut-être pas osé poursuivre sans ta motivation comme exemple.

Finalement, je remercie ma famille, qui m'a permis de faire les études qui m'ont menées à la rédaction de ce manuscrit.

‘Serpent!’ screamed the Pigeon.

[. . .]

‘But I’m NOT a serpent, I tell you!’ said Alice. ‘I’m a–I’m a–’

‘Well! WHAT are you?’ said the Pigeon. ‘I can see you’re trying to invent something!’

‘I–I’m a little girl,’ said Alice, rather doubtfully, as she remembered the number of changes she had gone through that day.

‘A likely story indeed!’ said the Pigeon in a tone of the deepest contempt. ‘I’ve seen a good many little girls in my time, but never ONE with such a neck as that! No, no! You’re a serpent; and there’s no use denying it. I suppose you’ll be telling me next that you never tasted an egg!’

‘I HAVE tasted eggs, certainly,’ said Alice, who was a very truthful child; ‘but little girls eat eggs quite as much as serpents do, you know.’

‘I don’t believe it,’ said the Pigeon; ‘but if they do, why then they’re a kind of serpent, that’s all I can say.’

Lewis Carroll, Alice’s Adventures in Wonderland

Table des matières

Table des figures

Chapitre 1

Introduction

Sommaire

En mars 1989, un rapport qui sera qualifié de « *Vague, but exciting.* » voit le jour au CERN. Celui-ci détaille une proposition de système d'information reposant sur un modèle client-serveur et utilisant les liens hypertextes. Il s'agit du premier document introduisant le web tel que nous le connaissons, et dont nous célébrons les 30 ans cette année. Durant ces trois décennies, le paysage a bien évolué. L'évolution matérielle, tout d'abord, a permis une augmentation exponentielle des capacités de stockage et de la gestion des données. L'accès à Internet par le grand public et l'émergence du commerce en ligne ont été des facteurs de développement pour les systèmes informatisés et l'automatisation de nombreuses tâches. En conséquence de nombreuses données se retrouvent stockées sur de supports informatiques. Par la suite, les pages se font plus dynamiques et facilitent la création de contenu par les internautes, qui n'ont plus à avoir leur propre hébergeur et site web pour s'exprimer en ligne. C'est notamment l'émergence des blogs et des wikis, qui permettent l'édition collaborative. Ces nouvelles pratiques participent à la massification des données accessibles sur le web. En 2001, les prémisses du web des données, aussi appelé web sémantique [1], sont esquissés (Berners-Lee et Hendler, 2001). Jusqu'alors, les données sur le web étaient simplement traitées par les machines. L'objectif du web des données est de permettre à ces machines de « comprendre » ces données, c'est-à-dire de pouvoir faire des inférences pour en extraire de l'information qui n'est pas contenue de manière explicite.

1.1 Le web des données

Les données ouvertes liées (*Linked Open Data*, LOD) se matérialisent par un ensemble de ressources en relation les unes avec les autres. Elles peuvent être représentées sous forme de graphe où les nœuds

1. Le terme originel, web sémantique, tend à être remplacé par le terme web des données : « The term "Semantic Web" refers to W3C's vision of the Web of linked data. » (`https://www.w3.org/standards/semanticweb/`, le 10 octobre 2019)

sont correspondent à des ressources et les arêtes des relations entre ces ressources. Le terme *knowledge graph* est parfois employé pour désigner un sous-ensemble de ces ressources.

L'objectif du web des données est d'encoder des connaissances qui soient interprétables et manipulables aussi bien par des humains que par des machines, ce qui implique (i) un format d'encodage des connaissances, (ii) la possibilité d'accéder à ces connaissances et (iii) la possibilité de raisonner sur ces connaissances. Il y a un hiatus entre les données et les connaissances, termes qui n'ont pas le même sens. Toutefois, les éléments de base constituant le web des données, les triplets RDF, peuvent être considérés à la fois comme des données et comme des unités de connaissance.

Concernant l'encodage des connaissances (i), le web des données repose sur des ontologies. L'ontologie est originellement une branche de la philosophie qui s'intéresse à « l'être en tant qu'être » : qu'est-ce qui le définit, quelles sont ses propriétés, ... Ce sont des questions abordées dès l'Antiquité, notamment par Aristote. Cependant, ce n'est qu'au XVIIe, que le terme ontologie est introduit pour désigner cette discipline. Dans le domaine de l'informatique, le terme a été introduit par McCarthy (1980) dans le champ de l'intelligence artificielle, pour désigner un ensemble de concepts et leur sens dans un contexte donné. Par la suite, la définition de Gruber (1993), qui désigne une ontologie comme une « spécification explicite d'une conceptualisation » a été largement adoptée. La notion de conceptualisation fait référence à une représentation, tandis que le terme « spécification explicite » implique que cette conceptualisation est accessible aussi bien à des humains qu'à des machines. Dans le cadre du web des données,le terme ontologie désigne aussi bien l'ensemble des connaissances représentées que la structure des concepts de cette représentation. Nous préférerons ici parler de bases de connaissances pour désigner l'ensemble des connaissances encodées. L'encodage est standardisé par le *World Wide Web Consortium* (W3C) qui a définit plusieurs langages à cette fin, tels que RDF et RDFS. Ceux-ci seront présentés dans le prochain chapitre.

L'accès aux connaissances encodées (ii) et la possibilité de raisonner sur ces connaissances (iii) nécessitent un moteur d'inférence. Celui-ci permet d'appliquer des règles d'inférences, et de dériver de nouvelles connaissances à partir des connaissances encodées dans la base. Par exemple, sachant que « Socrate est un Homme » et que « les Hommes sont des Mortels », le moteur d'inférence peut en déduire que » Socrate est Mortel ». Ces règles sont spécifiées par les différents standards qui régissent le web des données tels que RDFS et OWL. Le moteur d'inférence permet également de retrouver des informations présentes implicitement ou explicitement dans la base de connaissances.

Plus haut, nous avons déjà parlé de la différence entre données et connaissances. Nous allons voir ci-dessous comment peut se matérialiser le passage de l'une à l'autre.

1.2 Extraction de connaissances dans les bases de données

L'extraction de connaissances dans les bases de données (ECBD) a été définie par Fayyad, Piatetsky-Shapiro et Smyth (1996) comme un *processus non trivial d'identification de motifs valides, nouveaux, potentiellement utiles et compréhensibles dans des données*. Ce processus, illustré figure 1.1, se décompose en cinq étapes.

Tout d'abord, la *sélection* des données consiste à déterminer, parmi les données à disposition, lesquelles seront utilisées dans le processus. Ce choix dépend des hypothèses formulées et des connaissances *a priori* du domaine. Parfois, toutes les données peuvent être utilisées, d'autres fois il va s'agir d'échantillonner les données ou d'en isoler un sous-ensemble selon des critères choisis. Les données sélectionnées peuvent éventuellement être *pré-traitées* pour gérer le bruit et les données manquantes. Elles sont ensuite *transformées* dans une forme adéquate à la tâche de fouille à effectuer. Il peut par exemple s'agir d'une réduction du nombre de dimensions, ou simplement d'un changement de format. La *fouille de données* peut alors être effectuée. Il peut par exemple s'agir de classification, de classement, de régression, ... Les motifs obtenus sont *interprétés* puis *évalués* pour être finalement intégrés comme nouvelles

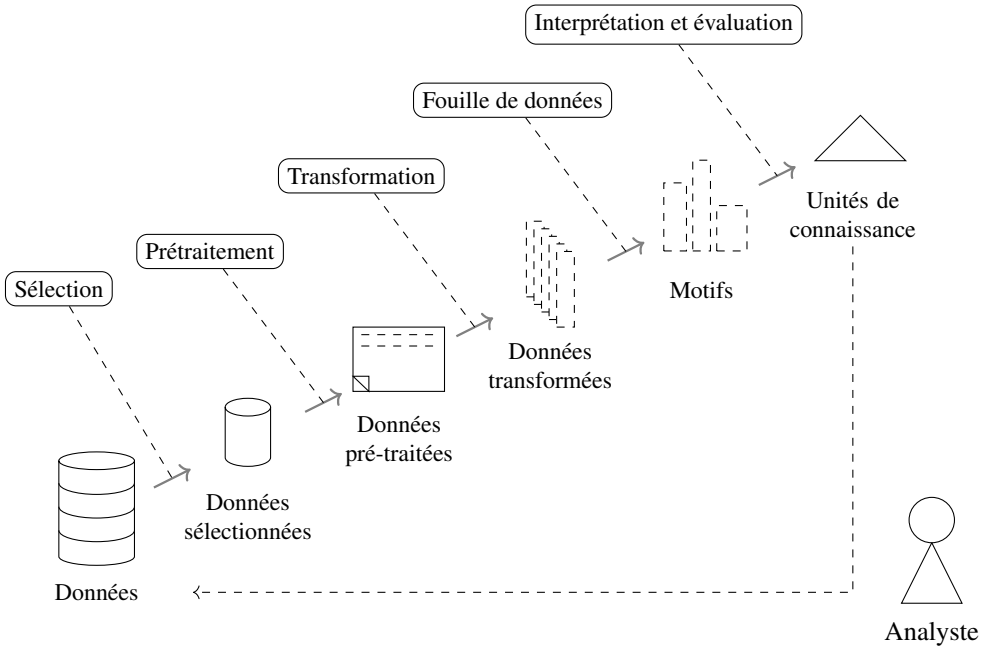


FIGURE 1.1 – Le processus d'extraction des connaissances. D'après Fayyad, Piatetsky-Shapiro et Smyth (1996).

unités de connaissance.

1.3 Problématique

L'un des mécanismes essentiels pour la construction et la manipulation de connaissances est la classification des entités. Grouper les choses qui nous entourent est un élément central du raisonnement humain que l'on retrouve dans de nombreux domaines, depuis les catégories d'Aristote jusqu'à la classification phylogénétique en passant par la classification de Mendeleïev. Ces catégorisations sont fondamentales, que ce soit, par exemple, pour l'indexation (ordonner et retrouver des livres dans une bibliothèque) ou pour le raisonnement (identifier les caractéristiques communes à un groupe d'entités). On les retrouve dans les nombreux travaux autour de la représentation des connaissances, et notamment dans les raisonnements logiques qui s'appuient sur la théorie des ensembles. En ce sens, une catégorie, ou une classe, est un ensemble d'objets qui partagent des caractéristiques. Il y a une dualité dans la définition d'une classe, selon que l'on considère l'ensemble des objets qui la composent – on parle alors de définition *en extension* – ou le sens qu'elle porte (c.-à-d. les caractéristiques que ses objets possèdent) – on parle dans ce cas de définition *en intension*.

Pour le web des données également, les classes sont des éléments clés. Considérons par exemple l'affirmation « Alan Turing est une voiture de course. » Pour détecter l'incongruité de cette affirmation, il est nécessaire de savoir qui est Alan Turing et ce qu'est une voiture de course. Dans une base de connaissances, il pourrait par exemple s'agir des connaissances suivantes : (1) Alan Turing est un scientifique, (2) les scientifiques sont des personnes et (3) une entité ne peut pas être à la fois une personne et une voiture de course. Cet exemple illustre l'importance des classes pour le raisonnement. Dans ce cas, des définitions en extension sont suffisantes pour exprimer ces trois connaissances et détecter que Alan Turing ne

peut pas être une voiture de course. En revanche, dans certains cas, ces définitions ne sont pas suffisantes, par exemple si notre base contient les connaissances suivantes : (1) Alan Turing est doué de parole, (2) les individus doués de parole sont des personnes et (3) une entité ne peut pas être à la fois une personne et une voiture de course. Dans ce cas, la connaissance (2) est liée à la définition en intension de la classe Personne.

Il arrive fréquemment qu'une classe ne soit définie qu'en extension, en listant ses membres. Or, ajouter sa définition en intension est non seulement un enrichissement en soi, mais cela permet également de faire des inférences qui n'étaient pas possibles avec l'extension seule, et donc d'obtenir de nouvelles connaissances. L'absence de définitions en intension dans les LOD s'explique principalement par la façon dont elles sont construites : de nombreuses bases de connaissances sont créées par l'extraction automatique de données préexistantes. Très souvent, ces données à disposition ne renferment pas la définition en intension. Il peut également être difficile de formuler une définition en intension. Une autre difficulté vient du fait que les données sont parfois collectées de manière collaborative, comme c'est le cas pour Wikipédia par exemple. S'assurer de la complétude et de l'exactitude des données n'est pas toujours possible dans ce cas. Autrement dit, des ressources peuvent manquer dans la définition extensionnelle, tandis que d'autres peuvent être présentes par erreur.

De nombreux travaux existent pour enrichir de manière automatique ou semi-automatique le web des données. Cependant, tandis que beaucoup d'entre eux s'intéressent aux définitions extensionnelles, notamment en cherchant à typer une instance (c.-à-d. lui associer une classe), peu de travaux cherchent à fournir une définition en intension. C'est précisément ce que nous essayons de faire dans cette thèse.

1.4 Contributions

Nous nous intéressons à la découverte de définitions dans le web des données. Concrètement, les définitions en intension correspondent à un couple associant une condition nécessaire (CN) et une condition suffisante (CS). Si x est une instance de la classe Rouge alors x a la couleur rouge (CN), et inversement, si x a la couleur rouge alors x est une instance de la classe Rouge (CS). L'idée est de construire et d'appliquer des règles d'induction de la forme « si $r(x,y)$ et $y : C$ alors $x : r.C$ ». Cela signifie que, étant données y une instance de la classe C et r une relation telle que $r(x,y)$, alors x appartient à une classe, disons D, dont la description comprend $r.C$. Nous utilisons ce type de règles pour construire les définitions des classes. Ces définitions sont de la forme $C_i \equiv e_1 \sqcap e_2 \sqcap \cdots \sqcap e_n$ où e_j est une expression de la forme $r.C_j$.

Le processus de découverte de définitions est résumé figure 1.2. Il se divise en trois tâches principales : (i) préparer les données, (ii) découvrir les définitions, (iii) évaluer la qualité de ces définitions. À partir d'une base de connaissances, un ensemble de triplets est extrait à l'aide d'une requête SPARQL. Cette requête permet à la fois de sélectionner un sous-ensemble de la base et d'opérer un pré-traitement en filtrant les triplets selon des critères. L'ensemble de triplets extraits sert à construire un contexte formel. Ce contexte est ensuite fourni en entrée des différents algorithmes utilisés. Ces derniers permettent d'extraire un ensemble de règles qui sont alors interprétées et évaluées. Finalement, on obtient un ensemble de définitions qui pourront éventuellement être intégrées à la base de connaissances.

Nous nous appuyons sur trois approches pour réaliser la tâche de découverte de définitions : les règles d'associations, les redescriptions et les règles de traduction. Les règles d'association permettent d'exprimer des dépendances entre les attributs, sous la forme « si x, alors y ». Les redescriptions correspondent à des caractérisations multiples d'un même ensemble d'entités, c'est-à-dire deux ensembles d'attributs qui désignent le même ensemble d'objets. La fouille de règles de traduction permet d'extraire aussi bien des associations que des redescriptions, mais est fondée sur le principe de description de longueur minimale. Dans cette thèse, nous décrivons et comparons les trois approches en discutant leurs avantages respectifs.

Enfin, nous menons un ensemble d'expérimentations sur DBpedia pour évaluer et comparer les différentes approches. Pour cela, nous proposons un processus expérimental pour extraire et évaluer les

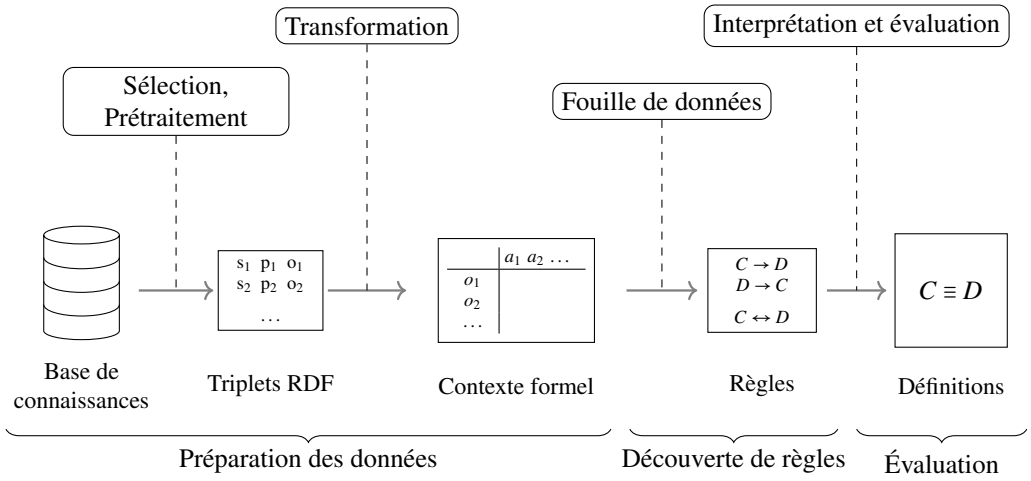


FIGURE 1.2 – Notre processus d'extraction des connaissances pour le web des données.

règles selon les trois approches comparées. Les définitions obtenues sont évaluées quantitativement et qualitativement.

Cette thèse est dans la continuité de travaux de recherche de l'équipe ORPAILLEUR 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.

1.5 Plan de la thèse

Ce mémoire de thèse est divisé en sept chapitres. Le présent chapitre a pour objectif d'introduire le contexte et la problématique, ainsi qu'une vue d'ensemble des contributions. Cette section détaille l'organisation des différents chapitres.

1.5.1 État de l'art

Chapitre 2 : Web des données. Le deuxième chapitre introduit les notions fondamentales du web des données. En particulier, il détaille les différents standards qui régissent son fonctionnement. Il présente ensuite les enjeux de qualité liés aux bases de connaissances ainsi qu'une classification des erreurs de conception. Enfin, la construction et le fonctionnement de DBpedia sont présentés, ainsi que les difficultés rencontrées pour s'assurer de sa qualité.

Chapitre 3 : Analyse formelle de concepts. Le troisième chapitre, présente l'analyse formelle de concepts et les treillis de concepts. Nous présentons un état de l'art qui peut se décomposer en deux parties : nous avons d'un côté les travaux cherchant à étendre l'analyse formelle de concepts (FCA), et de l'autre côté les applications de la FCA dans le domaine de la représentation des connaissances.

Chapitre 4 : Motifs et règles d'association. Le quatrième chapitre présente succinctement la fouille de motifs avant de présenter plus en détails la fouille de règles d'association. Devant la quantité de règles extraites, plusieurs approches ont été proposées pour réduire leur nombre sans perdre d'information. Nous présentons ces méthodes qui peuvent se distinguer en deux catégories : celles qui s'appuient sur une métrique et celles qui utilisent les propriétés des treillis pour extraire un sous ensemble de règle.

Nous détaillons ensuite quelques algorithmes de fouille de motifs et de règles d'association, avant de présenter un état de l'art des travaux de fouille de règles d'association appliqués au web des données.

Chapitre 5 : Fouille de Redescriptions. Le cinquième chapitre est dédié à la fouille de redescriptions. Après avoir introduit les définitions usuelles, nous présentons trois algorithmes représentatifs de la fouille de redescriptions. Nous présentons ensuite une seconde approche, à la croisée des règles d'association et des redescriptions, qui s'appuie sur des notions de théorie de l'information, notamment le principe de description de longueur minimale pour extraire les règles. Enfin, nous présentons plusieurs travaux liés à la fouille de redescription.

1.5.2 Contributions

Chapitre 6 : Découverte de définitions dans le web des données. Ce chapitre et le suivant présentent les différentes contributions de cette thèse. Dans le sixième chapitre, nous abordons les aspects théoriques, en détaillant deux contributions principales. La première est la définition d'une *pattern structure* permettent de classifier des ressources RDF en considérant les triplets dans lesquels elles sont impliquées. La seconde est l'utilisation des règles d'association et des redescriptions pour la découverte de définitions dans le web des données. Nous discutons notamment des points communs et des spécificités de chacune de ces applications en mettant en avant leur complémentarité.

Chapitre 7 : Expérimentations. Nous détaillons dans ce chapitre les expérimentations menées et les résultats obtenus. Dans une première expérience, les différentes approches sont comparées selon différents critères tels que le nombre de règles extraites, leur précision et leur rappel. Une deuxième expérience étend la comparaison entre les règles d'association et les redescriptions, tandis que les deux dernières expériences s'intéressent uniquement aux redescriptions. Des exemples de règles sont présentés et les résultats obtenus sont interprétés et discutés pour chaque expérience.

Chapitre 8 : Synthèse et conclusion. Enfin, le dernier chapitre propose une synthèse des différentes approches étudiées et des résultats obtenus. Il conclut cette thèse en proposant quelques perspectives de recherche offertes par les résultats obtenus.

Chapitre 2

Web des données

Sommaire

Dans ce chapitre, nous présentons les principales notions liées au web des données. Nous commençons par distinguer données, informations et connaissances. Puis nous discutons de la représentation des connaissances pour le web des données. Nous détaillons notamment les standards sur lesquels ces représentations reposent. Nous abordons ensuite le cas particulier de DBpedia, qui est une base de connaissances centrale dans le web des données. Finalement, nous présentons les problématiques liées à la qualité des données sur DBpedia et plus généralement sur le web des données.

2.1 Données, informations et connaissances

Avant d'introduire les bases de connaissances, nous introduisons ici la distinction entre données, informations et connaissances. Les définitions et délimitations de ces notions sont multiples et couvrent de nombreux champs tels que les sciences cognitives, l'économie ou la philosophie. Wille (2002), à la suite de Schreiber et Akkermans (1999), formalise la distinction de la manière suivante :

Données. Les données sont des symboles respectant une syntaxe.

Informations. Une information est une donnée interprétée, c'est-à-dire une donnée accompagnée d'une sémantique.

Connaissances. Une connaissance est une information assimilée, permettant de raisonner et de construire de nouvelles connaissances.

Ces trois notions sont souvent représentées sous la forme d'un triangle dont les données constituent la base (figure 2.1). Cette représentation est en accord avec plusieurs constats. Tout d'abord, l'idée de hiérarchie

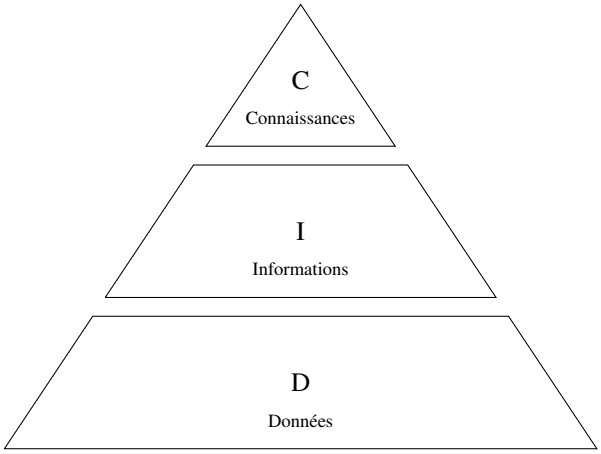


FIGURE 2.1 – Données, informations et connaissances.

entre données et connaissances est pertinente : usuellement, les connaissances sont bien plus précieuses que les données. De plus, cette hiérarchie est consistante avec la représentation du processus d'extraction de connaissances. Enfin, on parle souvent de « masses de données » desquelles on souhaite extraire des connaissances en moins grand nombre, justifiant les différences d'aires entre les trois composants.

Kayser (1997) discute de la relation entre informations et connaissances, remarquant qu'il y a « transition par degrés entre information et connaissances ». Il place la distinction au niveau du rôle des informations et des connaissances dans le raisonnement : tandis qu'une information est simplement manipulée au cours de différents processus, les connaissances peuvent influencer ces processus.

La notion de *continuum* entre données et connaissances a notamment été discutée par Floridi (2010). En particulier, l'auteur discute de la diversité de ce qui peut être regroupé derrière le terme information. Il en propose une classification, reproduite en figure 2.2. Dans un premier temps, il fait la différence entre des informations environnementales et sémantiques : les premières sont des données qui peuvent être interprétées indépendamment de l'émetteur, contrairement aux secondes. Pour reprendre l'exemple de l'auteur, les cercles concentriques d'un tronc d'arbres constituent une information environnementale sur l'âge de l'arbre. Ces informations, qu'elles soient environnementales ou non, peuvent être instructionnelles (un feu de circulation, un manuel de montage de meuble, etc.) ou factuelles (un fait énoncé). Parmi les faits énoncés, certains peuvent être faux. L'auteur parle alors de désinformation si cela est intentionnel de la part de l'émetteur, et de mésinformation sinon. Les informations factuelles et vraies permettent de construire des connaissances. Cette classification soulève la question de la place de la vérité dans la notion de connaissance, que nous n'aborderons pas dans cette thèse.

Du côté du web sémantique, il n'y a pas de distinction claire entre données et connaissances. On parle de bases de *connaissances*, mais aussi de web des *données*. La distinction se fait plutôt du côté opérationnel : on parlera de connaissances lorsque des processus d'inférence sont en jeux, et d'information pour désigner les faits encodés dans la base. La figure 2.3 représente l'organisation d'un système de représentation de connaissances. La base de connaissances est le cœur de ce système, elle se compose d'une ABox (un ensemble d'assertions) et d'une TBox (une structure sur les connaissances). Elle est exprimée dans un langage de description et sert de support au raisonnement. Les connaissances se décomposent en deux ensembles distincts : d'une part les assertions qui expriment des faits, regroupées dans la ABox, et d'autre part les connaissances structurantes, regroupées dans la TBox.

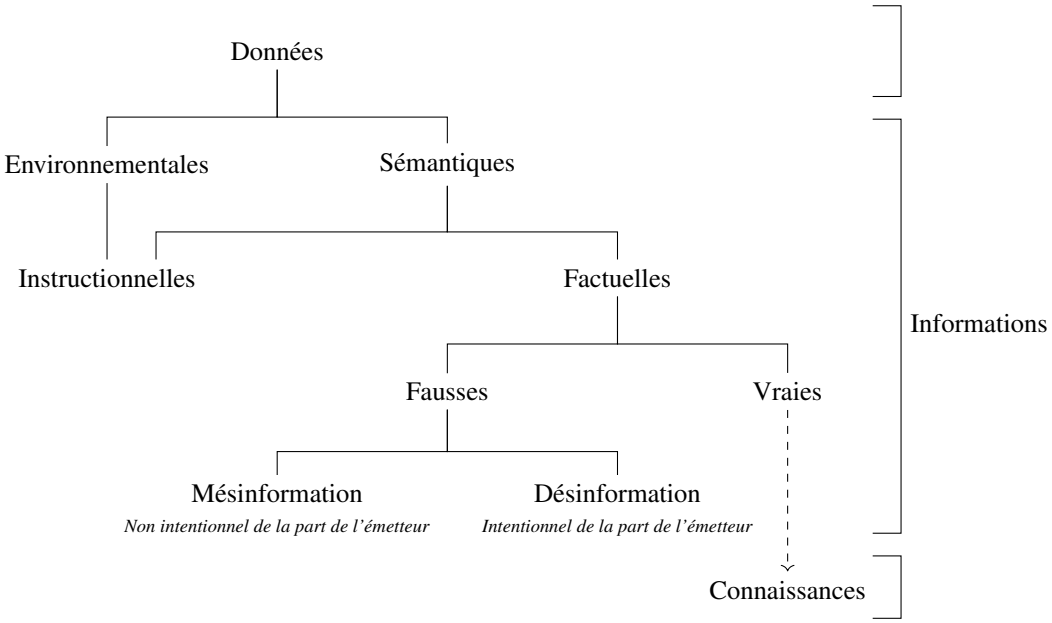


FIGURE 2.2 – Différents types d'information, d'après Floridi (2010)

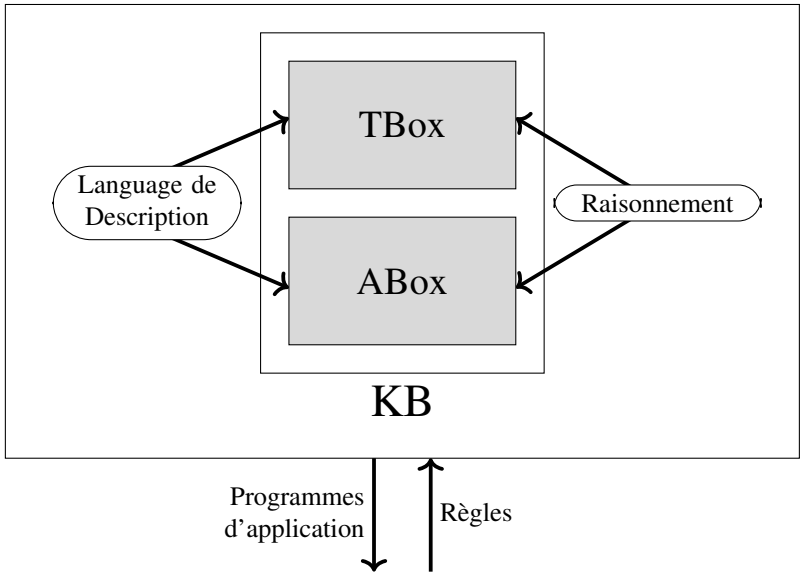


FIGURE 2.3 – Système de représentation des connaissances, d'après Baader, Horrocks et Sattler (2009)

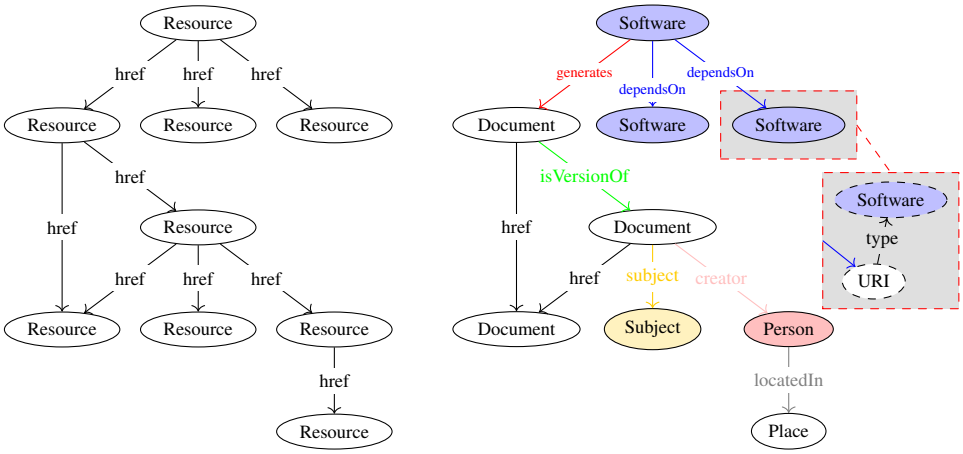


FIGURE 2.4 – Du web « traditionnel » au web des données. D'après Koivunen et E. Miller (2002)

2.2 Représentation des connaissances sur le web

La figure 2.4 représente le web tel que défini initialement et tel que défini par le web des données. À l'origine, le web est simplement constitué de ressources qui sont des documents consultables en ligne, comme des pages web par exemple, liés entre eux par des liens hypertextes. Avec le web des données, une ressource peut être décrite plus finement et les relations entre une ressource et une autre peuvent être annotées sémantiquement. Une URL ne correspond plus nécessairement à un simple document, mais peut désigner un concept, un objet, une personne, etc. Pour rendre compte de ce « nouveau » web, de nouveaux standards sont nécessaires.

2.2.1 Les standards du web des données

La représentation des connaissances dans le web des données repose sur un ensemble de standards définis par le *World Wide Web Consortium*[2] (W3C). Ces standards sont structurés en couches. Les premières couches définissent une syntaxe tandis que les couches plus élevées leurs associent une sémantique. Le *semantic cake* (figure 2.5) représente les différents standards qui régissent le web des données.

URI/IRI

Les URI (*Uniform Resource Identifier*, identifiant uniforme de ressource) sont des chaînes de caractères qui permettent d'identifier une ressource sur le réseau. Les IRI (*Internationalized Resource Identifier*, identifiant internationalisé de ressources) sont une généralisation des URI, permettant notamment d'utiliser des caractères Unicode au lieu de se restreindre à l'ASCII dans l'adresse. Les ressources sont souvent rendues accessibles par le réseau, dans ce cas les URI/IRI correspondent à des URL (*Uniform Resource Locator*, localisateur uniforme de ressources).

Les IRI peuvent se décomposer en deux parties. La première identifie le *namespace*, ou espace de noms d'une ressource, tandis que la seconde partie identifie la ressource au sein de cet espace de noms. Par exemple, `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` identifie la ressource `type` au sein de l'espace de noms `http://www.w3.org/1999/02/22-rdf-syntax-ns`. L'espace de noms peut

2. https://www.w3.org/

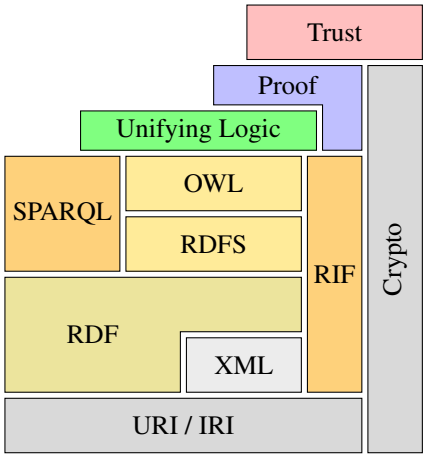


FIGURE 2.5 – Les différentes couches du web des données.

être abrégé par un préfixe. Dans l'exemple précédent, l'espace de noms est usuellement abrégé par le préfixe `rdf` ; la ressource `type` est alors identifiée par `rdf:type`.

RDF et RDFS

Le standard *Resource Description Framework* (Carroll et Klyne, 2004), abrégé RDF, spécifie la façon dont les connaissances sont représentées dans le web des données. Pour cela, il définit la notion de triplet RDF, composé d'un sujet, d'un prédicat et d'un objet et dénoté $\langle \texttt{sujet}, \texttt{predicat}, \texttt{objet} \rangle$. Un triplet $\langle s, p, o \rangle$ correspond à une déclaration logique, ou *assertion*, prenant la forme d'une relation binaire p entre s et o. Le sujet d'un triplet peut être une ressource ou un nœud anonyme [3] (*blank node*), le prédicat est une ressource et l'objet peut être une ressource, un *blank node* ou un littéral (entier, chaîne de caractères, date, etc.).

Définition 1 (Triplet RDF)**.** *Un triplet RDF est un triplet* $\langle s, p, o \rangle \subseteq (U \cup B) \times U \times (U \cup B \cup L)$ *où* U *est l'ensemble des ressources identifiées par une IRI,* B *est l'ensemble des nœuds anonymes et* L *est l'ensemble des littéraux.*

Un ensemble de triplets peut être représenté sous la forme d'un graphe RDF $G = (V, E)$ où V est l'ensemble des sujets et des objets et E est l'ensemble des prédicats. Ainsi un triplet correspond à deux nœuds reliés par une arête. La figure 2.6 présente un exemple de connaissances encodées sous forme de triplets et le graphe correspondant. L'espace de noms `ex:` correspond à notre exemple.

RDF définit également un vocabulaire qui comporte entre autres le prédicat `rdf:type`, une relation d'instanciation entre le sujet et l'objet d'un triplet. Par exemple, le triplet ⟨`ex:Nancy, rdf:type, ex:Ville`⟩ déclare que la ressource `Nancy` est une instance de la ressource `Ville`.

À partir de RDF, les ressources peuvent être distinguées en termes de propriétés, classes et instances. Les propriétés, ou prédicats, expriment des relations binaires. Dans l'exemple de la figure 2.6, `ex:CodePostal` et `rdf:type` sont des propriétés. Les classes correspondent à des ensembles qui regroupent des ressources, comme `ex:Ville` par exemple. Enfin, les ressources qui ne sont ni des propriétés ni des classes sont des instances [4]. Dans l'exemple, `ex:Nancy` et `ex:Russie` sont des instances.

3. Un nœud anonyme est assimilable à une variable existentielle.

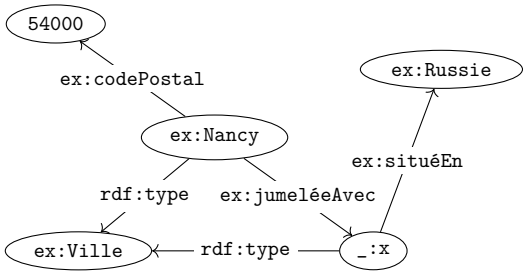
4. Les classes et les propriétés peuvent également être des instances. Par exemple le prédicat `rdf:type` est une instance de `rdf:Property`. Cette nuance n'a pas d'incidence sur le reste de cette thèse.

« Nancy est une ville dont le code postal est 54000. Elle est jumelée avec une ville russe. »

(a) Connaissances encodées

```
ex:Nancy rdf:type ex:City .
ex:Nancy ex:codePostal 54000 .
ex:Nancy rdf:jumeléeAvec _:x .
_:x rdf:type ex:City .
_:x situéEn ex:Russie .
```

(b) Triplets RDF



(c) Graphe

FIGURE 2.6 – Les deux modélisations de ressources en RDF : triplets et graphe.

Le standard RDFS — *RDF Schema* (Brickley et Guha, 2014) — étend le vocabulaire RDF pour permettre de structurer les ressources. Il définit notamment les relations `rdfs:subClassOf` et `rdfs:subPropertyOf` qui permettent de définir un ordre partiel sur les classes et les prédicats respectivement. Par exemple, le triplet ⟨`ex:Ville, rdfs:subClassOf, ex:Lieu`⟩ déclare que la classe `Ville` est une sous-classe de `Lieu`. Ces relations sont transitives. Elles permettent également de faire des inférences. Par exemple, à partir des triplets ⟨`ex:Nancy, rdf:type, ex:Ville`⟩ et ⟨`ex:Ville, rdfs:subClassOf, ex:Lieu`⟩, il est possible de déduire le triplet ⟨`ex:Nancy, rdf:type, ex:Lieu`⟩. Ces relations définissent donc deux ordres partiels, l'un sur les classes, et l'autre sur les prédicats.

OWL

OWL (Dean et Schreiber, 2004), pour *Ontology Web Language*, enrichit le web des données sur deux aspects principaux : d'une part, il offre un nouveau vocabulaire, permettant de modéliser des connaissances plus complexes et d'autre part, il fait le lien entre les standards du web des données et les logiques de description.

Le vocabulaire introduit par OWL intègre de nouveaux prédicats qui expriment des relations entre les classes, telles que la disjonction ou l'équivalence. Il introduit également des propriétés sur les relations, comme la symétrie, la transitivité et la réflexivité, et une distinction des prédicats en trois types. Les prédicats regroupés dans la classe `owl:datatypeProperty` sont les prédicats dont le co-domaine est un littéral, alors que les `owl:ObjectProperty` sont les prédicats dont le co-domaine est une resource ou un *blank node*. Enfin, `owl:AnnotationProperty` regroupe les prédicats servant à annoter ou ajouter des métadonnées et qui ne sont pas pris en compte par les moteurs d'inférence.

OWL prend appui sur les logiques de description, et spécifie notamment le lien entre leur syntaxe et la syntaxe de RDF. Ces relations ont notamment été discutées par Horrocks (2005). Certaines correspondances sont reportées dans la table 2.1. Enfin, OWL définit plusieurs profils, à l'expressivité et à la complexité variables (voir la figure 2.2). Le profil le plus expressif, OWL DL, correspond à la logique de description $\mathcal{SROIQ}$.

SPARQL

Une base de connaissances encodée au format du web des données peut être interrogée par un langage de requêtes, tel que *SPARQL Query Language*, abrégé SPARQL (Harris et Seaborne, 2013). Les requêtes sont construites à partir de motifs de graphes encodés sous forme de triplets, appelés *Basic Graph Patterns* (BGP). Les réponses d'une requête correspondent à l'ensemble des sous-graphes isomorphes au BGP. Des

TABLE 2.1 – Correspondance entre la syntaxe RDF et la syntaxe des logiques de description. $\Delta^{\mathcal{I}}$ est le domaine d'interprétation, construite à partir de (Horrocks, 2005) et (Dean et Schreiber, 2004).

Syntaxe RDF	Syntaxe LDs	Sémantique
`ex:C` (concept atomique)	C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
`ex:r` (rôle atomique)	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
`owl:Thing`	$\top$	$\Delta^{\mathcal{I}}$
`owl:Nothing`	$\bot$	$\varnothing$
`ex:C rdfs:subClassOf ex:D`	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
`ex:p rdfs:subPropertyOf ex:q`	$p \sqsubseteq q$	$p^{\mathcal{I}} \subseteq q^{\mathcal{I}}$
`owl:unionOf`	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
`owl:intersectionOf`	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
`owl:complementOf`	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y, (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y, (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
`rdfs:type`	$C(x)$	$x \in C^{\mathcal{I}}$
`rdfs:domain`	$\exists r.\top \sqsubseteq C$	$r^{\mathcal{I}} \subseteq C^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
`rdfs:range`	$\top \sqsubseteq \forall r.C$	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C^{\mathcal{I}}$
`owl:disjointWith`	$C \sqcap D \sqsubseteq \bot$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \varnothing$

contraintes sur les nœuds du BGP peuvent être ajoutées avec le mot-clé `FILTER`. Un exemple de requête est présenté en figure 2.7.

2.2.2 Les *Linked Open Data*

Berners-Lee (2009) définit cinq niveaux de qualité des données, qualifiant les données de une à cinq « étoiles » selon le nombre de critères remplis parmi les suivants :

1. Rendre le contenu disponible sur le web ;
2. Utiliser du contenu structuré ;
3. Utiliser des formats non propriétaires ;
4. Utiliser des URL pour identifier les ressources ;

TABLE 2.2 – Les différents profils de OWL, d'après Horrocks et al. (2012)

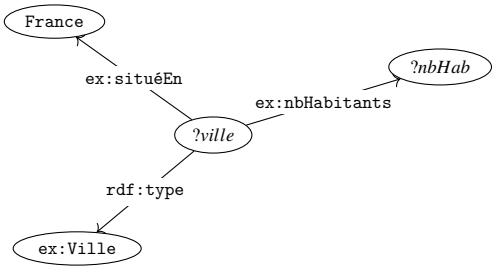
Nom	LD	Description
OWL DL	$\mathcal{SROIQ}$	Il s'agit du profil le plus expressif de OWL. Les autres profils sont un sous-ensemble de OWL DL.
OWL EL	$\mathcal{EL}^{++}$	Le profil OWL EL est pertinent quand il y a un grand nombre de prédicats et/ou de classes.
OWL QL	*DL – Lite*	Le profil OWL QL est adapté lorsqu'il y a de nombreuses instances.
OWL RL	*DLP*	Le profil OWL RL offre un compromis entre une bonne expressivité et un bon passage à l'échelle pour faire des inférences.

« Quelles sont les villes situées en France de plus de 100 000 habitants ? »

(a) Question posée

```
SELECT ?ville WHERE {
  ?ville rdf:type ex:Ville .
  ?ville ex:situéEn ex:France .
  ?ville ex:nbHabitants ?nbHab .
  FILTER (?nbHab > 100000)
}
```

(b) Requête SPARQL



(c) BGP associé

FIGURE 2.7 – Exemple de requête SPARQL.

5. Lier les données aux données d'autres bases de connaissances.

Une base de connaissances sur le web valide le premier niveau. Si elle est au format RDF, elle valide les quatre premiers niveaux. Elle valide le cinquième niveau si elle est en relation avec d'autres bases de connaissances. Le web des données désigne l'ensemble des données « cinq étoiles » : les ressources sont identifiées et accessibles en ligne dans des formats non propriétaires, elles sont structurées, et sont liées les unes aux autres (c.-à-d. toutes les bases de connaissance comportent des instances reliées à des instances d'au moins une autre base). Le *linked open data cloud*[5] (*LOD cloud*) recense les différentes bases de connaissances composant les données ouvertes liées ainsi que leurs relations. En mars 2019 le *LOD cloud* était composé de 1239 jeux de données et de 16147 relations entre ces jeux de données.

2.3 DBpedia

DBpedia (Auer et al., 2007) est une base de connaissances construite par l'extraction automatique à partir de Wikipédia. Elle résulte d'un projet universitaire initié il y a plus de 10 ans. Aujourd'hui, DBpedia est un élément central des LOD. D'après les données du *LOD cloud*, la version anglaise de DBpedia est reliée à 375 des 1239 bases de connaissances du web des données, soit plus d'un quart d'entre elles. Si l'on considère toutes les langues, 431 bases de connaissances sont reliées, soit plus du tiers d'entre elles.

2.3.1 De Wikipédia à DBpedia

Pour bien comprendre la construction de DBpedia, il est nécessaire de s'intéresser à Wikipédia dans un premier temps. Wikipédia est une encyclopédie collaborative créée en 2001, permettant à toute personne de consulter et éditer le contenu. Typiquement, un article sur Wikipédia est composé des éléments suivants :

Le titre de la page Le titre de la page indique le sujet abordé. Sur DBpedia, le titre correspond au label de la ressource.

Un court résumé Ce résumé présente succinctement la ressource. Il est importé dans DBpedia avec le prédicat `dbo:abstract`.

Une image d'aperçu De nombreuses pages ont une image d'aperçu, en plus des images illustrant le contenu. Le lien vers l'image est présent dans DBpedia avec le prédicat `dbo:thumbnail`.

5. https://lod-cloud.net/

TABLE 2.3 – Espaces de noms utilisés par DBpedia.

Préfixe	URL
dbr	http://dbpedia.org/page/
dbp	http://dbpedia.org/property/
dbc	http://dbpedia.org/page/Category:
dbo	http://dbpedia.org/ontology/

Une infobox L'infobox représente les informations clés liées à la ressource. Pour une ville, il peut s'agir par exemple du code postal, du maire, du nombre d'habitants, etc. Ces données sont automatiquement importées dans DBpedia. Les prédicats utilisés dépendent des informations de l'infobox. Par exemple, le code postal est indiqué avec le prédicat `dbo:PostalCode`.

Du contenu textuel Le cœur de l'article est constitué du contenu textuel. Ce contenu n'est pas importé sur DBpedia.

Du contenu structuré Certaines pages ont du contenu structuré intégré au contenu textuel, comme des tableaux par exemple, qui est facilement importé sur DBpedia. Par exemple, la page de Nancy comporte un tableau de données météorologiques par mois, qui se retrouvent sur DBpedia. La relation `dbp:aprSnowDays`, par exemple, indique le nombre de jours où il a neigé en avril pour une ville donnée.

Des références Les références et les liens vers d'autres sites sont listées en bas de page. Les liens sont importés sur DBpedia avec le prédicat `dbo:wikiPageExternalLink`.

Des références vers des pages dans d'autres langues La page comprend des renvois vers les autres langues. Ces renvois sont inclus dans DBpedia : la base de connaissances inclut un espace de noms par langue et une même ressource dans différentes langues est liée par la relation `owl:sameAs`.

Des catégories Les catégories permettent de grouper des pages de manière thématique. Elles sont présentées plus en détails dans la section suivante.

En plus des espaces de noms spécifiques aux différentes langues de l'encyclopédie, DBpedia possède plusieurs espaces de noms pour représenter le contenu de Wikipédia, listés en table 2.3. Ceux-ci distinguent les instances, les classes, les prédicats et les catégories.

La figure 2.8 donne en exemple quelques triplets qui peuvent être extraits à partir d'une page Wikipédia.

2.3.2 La notion de catégorie

Les catégories dans Wikipédia

Comme mentionné plus haut, Wikipédia intègre un système de catégories. Une entrée encyclopédique peut être associée à une ou plusieurs catégories. Par exemple, sur Wikipédia, la page concernant la ville de Nancy [6] est associée à six catégories :

— Communes of Meurthe-et-Moselle

— Nancy, France

— Prefectures in France

— World Heritage Sites in France

6. https://en.Wikipédia.org/wiki/Nancy,_France

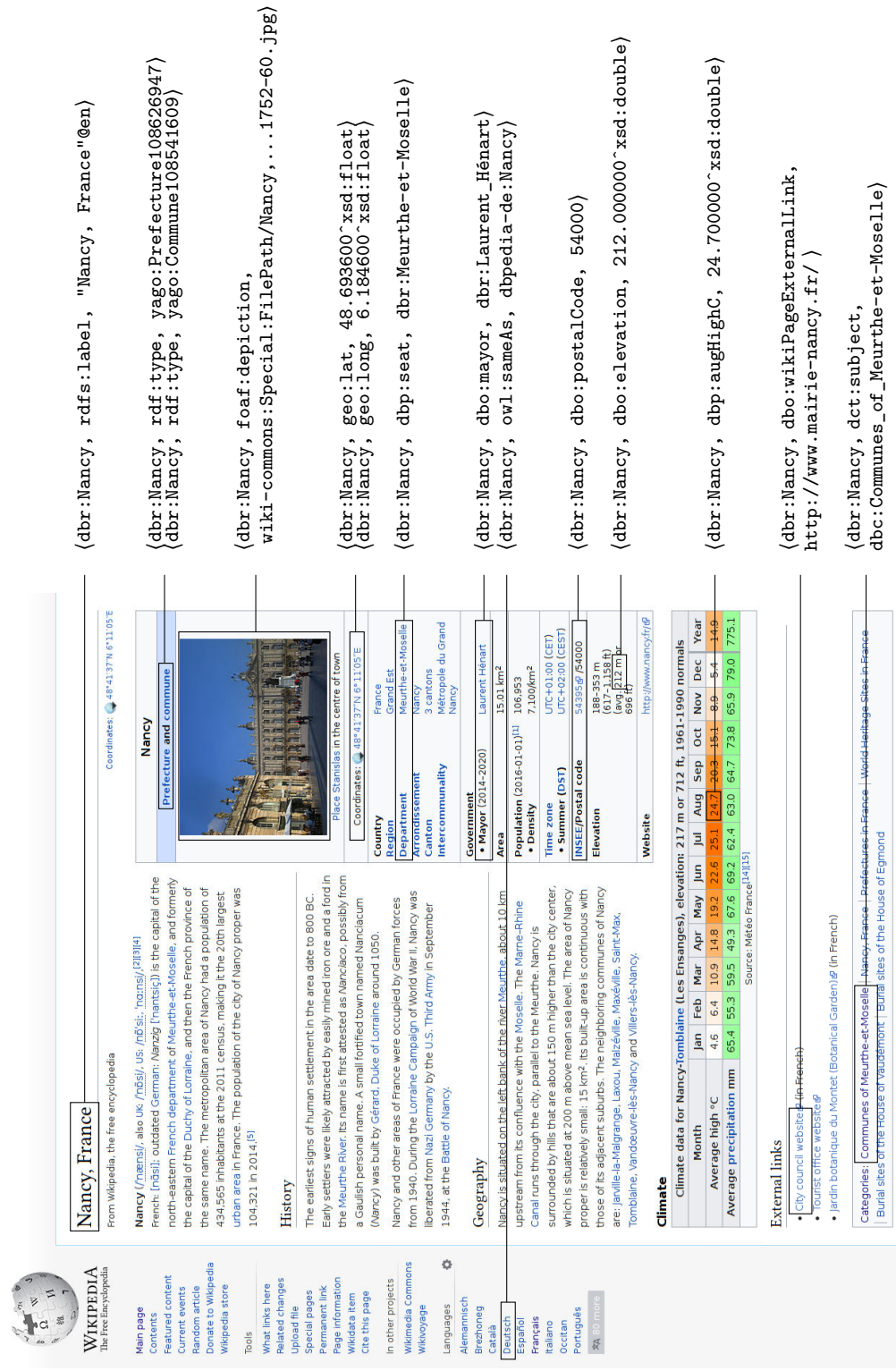


FIGURE 2.8 – Exemples de triplets extraits à partir d'une page Wikipédia.

— Burial sites of the House of Vaudémont

— Burial sites of the House of Egmond

Ces catégories sont créées par les utilisateurs, et correspondent à des pages spécifiques qui listent des entrées de l'encyclopédie.

Deux types de catégories sont distinguées : les *topic categories* et les *set categories*[7]. Les *topic categories* regroupent des pages sur le même sujet. Par exemple, la catégorie *Meurthe-et-Moselle* regroupe des pages qui peuvent être liées à ce département. On y trouve aussi bien la page *Côtes de Toul* que la page *Baccarat (company)* ou encore *Battle of Nancy (1944)*. Les *set categories* listent les pages qui répondent à certains critères. Par exemple, la catégorie *Communes of Meurthe-et-Moselle* regroupe toutes les pages qui traitent d'une commune du département de Meurthe-et-Moselle.

Les catégories sont partiellement ordonnées par une relation de spécificité/généralité. Par exemple, la catégorie *Prefectures in France* est une sous-catégorie de *Communes in Grand Est* et la catégorie *Meurthe-et-Moselle* a pour sous-catégorie *Geography of Meurthe-et-Moselle*. L'ordre partiel établi ne tient pas compte de la distinction entre les deux types de catégories.

Les catégories dans DBpedia

Les catégories de Wikipédia sont automatiquement importées dans DBpedia. Les ressources correspondantes ont un espace de noms dédié, préfixé `dbc`[8]. Pour associer une ressource à une catégorie, le prédicat `dct:subject` est utilisé. Le préfixe `dct`[9] correspond au vocabulaire Dublin Core, dédié à la description de documents. Le prédicat `subject` est utilisé à des fins de classification et permet d'indiquer des mot-clés ou des thèmes associés à une ressource. Sur la page DBpedia associée à la ville de Nancy[10], on retrouve donc six triplets indiquant l'appartenance de la ressource `Nancy` à chacune des catégories Wikipédia mentionnées plus haut :

```
dbr:Nancy dct:subject dbc:Communes_of_Meurthe-et-Moselle .
dbr:Nancy dct:subject dbc:Burial_sites_of_the_House_of_Egmond .
dbr:Nancy dct:subject dbc:Prefectures_in_France .
dbr:Nancy dct:subject dbc:World_Heritage_Sites_in_France .
dbr:Nancy dct:subject dbc:Burial_sites_of_the_House_of_Vaudémont .
dbr:Nancy dct:subject dbc:Nancy,_France .
```

2.3.3 La classification dans DBpedia

DBpedia intègre plusieurs classifications de ses ressources. Nous présentons ici les plus utilisées d'entre elles.

La taxonomie de classes

DBpedia intègre dans `dbo` un ensemble de classes construites et ordonnées manuellement. La taxonomie ainsi formée contient 760 classes partiellement ordonnées au sein d'un arbre dont la racine est `owl:Thing`.[11]

Cette taxonomie permet d'organiser les ressources présentes dans DBpedia. Cependant, elle n'est pas exempte de défauts. Le premier inconvénient de cette taxonomie de classes est que son « pouvoir

7. https://en.Wikipédia.org/wiki/Wikipédia:Categorization#Category_tree_organization
8. http://dbpedia.org/page/Category:
9. http://purl.org/dc/terms/
10. http://dbpedia.org/page/Nancy,_France
11. Les statistiques fournies dans cette section s'appuient sur la version 2016-10 de DBpedia (https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10).

classificatoire » est très limité. En effet, elle ne contient que 760 classes alors que près de 5.5 millions d'instances sont classifiées dans cette taxonomie. De ces chiffres, on peut en déduire que les classes sont très généralistes et ne permettent pas d'opérer une distinction fine des ressources. Le deuxième inconvénient est la répartition des instances par classes. En effet, les classes sont très déséquilibrées, même lorsqu'il s'agit de deux classes au même niveau. Par exemple, la classe `Mammal`, qui est une sous-classe d'Animal, contient 15 109 ressources. Elle possède seulement trois sous-classes : `Horse`, `Cat` et `Dog`. Les classes `Cat` et `Dog` ne contiennent aucune instance, tandis que la classe `Horse` contient 4 379 instances. La classe `Horse` contient une unique sous-classe, `RaceHorse`, qui rassemble la quasi-totalité (4 050) des instances de `Horse`. Enfin, malgré l'attention portée à la structure de cette taxonomie, certaines erreurs structurelles, comme des cycles de subsomption, persistent.

Les catégories

Les catégories offrent une classification supplémentaire aux ressources de DBpedia. Elles sont plus nombreuses que les classes et la classification qui en résulte a une granularité bien plus fine. Contrairement aux classes, une catégorie peut avoir plusieurs parents. L'ensemble des catégories forme donc un graphe orienté acyclique. Les catégories sont organisées selon la relation `skos:broader` et son inverse `skos:narrower`.

Bien que ces relations soient opérationnellement transitives, ce n'est pas toujours évident du point de vue sémantique. Par exemple, la catégorie *Turing Award laureates* est une sous-catégorie de la catégorie *Alan Turing*, elle-même une sous-catégorie de *English computer scientists*. Or, un lauréat du prix Turing, peut être considéré comme lié à Alan Turing, mais il n'a pas forcément de relation avec la catégorie *English computer scientists*.

Classifications « externes »

DBpedia intègre également les classifications de Yago (F. Suchanek, Kasneci et Weikum, 2007) et de WordNET (G. A. Miller, 1998). WordNET est une base de données linguistique développée depuis plus de 20 ans. Elle intègre de nombreuses informations lexicales de la langue anglaise, liant des concepts par des relations d'hyperonymie/hyponymie (est plus spécifique que), de méronymie/holonymie (fait partie de) et de synonymie/antonymie (a le même sens que). WordNET a par la suite été convertie et intégrée aux LOD (Assem, Gangemi et Schreiber, 2006). Yago (*Yet Another Great Ontology*) est construite à partir des données de DBpedia, de WordNET et de GeoNames (une base de données géographiques). En combinant ces trois bases, Yago propose une classification plus fine, combinant à la fois les catégories et les *synsets* de WordNET. Les assertions présentes dans Yago peuvent être spatialement et temporellement délimitées. De plus, elles sont associées à une valeur de confiance, qui indique la fiabilité de l'information.

Intérêt et limites

Les différentes classifications sur DBpedia permettent d'organiser les ressources. Les classes et les catégories sont les deux classifications propres à DBpedia. Nous avons d'une part la taxonomie de classes, construite manuellement par des utilisateurs avertis, avec beaucoup de données manquantes : la plupart des classes sont très peu peuplées, et le nombre total de classes est faible au regard de la taille de DBpedia. D'autre part, nous avons les catégories, créées par les utilisateurs de Wikipédia, qui sont beaucoup plus nombreuses et offrent une classification beaucoup plus fine. Cependant, dans DBpedia, très peu d'informations sont fournies sur les catégories. Aussi, dans cette thèse, nous nous appliquerons des approches de fouille de définitions aux catégories de DBpedia.

2.4 Qualité des données dans le web des données

Avec l'essor des standards du web des données, un nombre croissant de bases de connaissances sont publiées dans les LOD. Plusieurs méthodes de conception sont employées. Certaines s'appuient sur des procédures (semi–)automatiques, d'autres sur du *crowdsourcing* et reposent sur la contribution des utilisateurs. En conséquence de ces méthodes de construction, la base peut se retrouver incomplète voire erronée. Ce constat motive les travaux sur la qualité de conception des bases de connaissances.

Gangemi et al. (2006) distinguent trois composantes d'une base de connaissances : l'aspect structurel, l'aspect fonctionnel et l'aspect ergonomique. L'aspect structurel de la base de connaissances est lié à la syntaxe et aux bonnes pratiques de conception d'ontologies. L'aspect fonctionnel s'intéresse au fait que la base de connaissances est bien en accord avec la réalité. Enfin, l'aspect ergonomique vérifie que la base est encodée de manière à être utilisable dans le but prévu. Une base peut donc être fonctionnelle mais pas ergonomique. Ces trois composantes correspondent aux trois dimensions de l'évaluation d'une base de connaissances introduites par Gómez-Pérez, Fernández-López et Corcho (2004) : la consistance syntaxique, la consistance sémantique et l'ergonomie.

Zaveri, Rula et al. (2016) proposent une classification des différentes méthodes d'évaluation des LOD. Ils définissent 18 dimensions réparties en 4 groupes. Zaveri, Kontokostas et al. (2013) identifient quatre de ces dimensions comme particulièrement fréquentes dans DBpedia et les détaillent en 7 catégories et 17 sous-catégories (voir en figure 2.9). On retrouve des problèmes liés à la précision lorsque des données sont erronées, à la pertinence lorsque des données non pertinentes ou redondantes sont importées dans DBpedia, à la consistance lorsque des valeurs numériques sont mal représentées et à l'interconnexion lorsque des liens vers d'autres bases de connaissances ou des sites web sont erronés. Les erreurs de précision se retrouvent notamment lorsque les données sont typées de manière incorrecte, ou lorsque les valeurs sont erronées. Les erreurs de pertinence se manifestent quand du contenu relatif aux illustrations de Wikipédia est importé (c'est par exemple le cas du triplet lié à l'URL de l'image dans la figure 2.8), ou quand plusieurs prédicats portent la même information. Il existe par exemple plusieurs prédicats synonymes sur les espaces de noms `dbp` et `dbo`, ce qui est source de confusion. Les erreurs de consistance sur DBpedia sont principalement liées aux valeurs numériques. La page DBpedia de Nancy, par exemple, contient des informations sur le nombre de jours enneigés pour chaque mois de l'année. Les valeurs sont typées de manière inconsistante soit comme des *integer* soit comme des *double*. Enfin, les problèmes d'interconnexion viennent principalement d'alignements incorrects. Nous ne développerons pas cet aspect ici.

Gómez-Pérez, Fernández-López et Corcho (2004) proposent une évaluation des taxonomies de classes en classifiant les erreurs qui peuvent être faites à la conception. Comme pour l'évaluation de la base de connaissances, les erreurs sont partitionnées selon trois types d'erreurs : les erreurs d'inconsistance, d'incomplétude et de redondance. La figure 2.10 reproduit leur classification.

Synthèse du chapitre

Dans ce chapitre, nous avons présenté les standards du *World Wide Web Consortium*, qui permettent d'encoder des connaissances sous forme de triplets et de constituer des bases de connaissances. Ces bases de connaissances, liées les unes aux autres, forment le *Linked Open Data Cloud*, qui constitue l'ensemble des données libres ouvertes et liées accessibles sur le web. DBpedia est une base de connaissances construite à partir des données de Wikipédia qui a une position centrale dans les LOD. Wikipédia étant construite de manière collaborative, et l'extraction des données se faisant de façon automatique, il en résulte que DBpedia n'est pas complète (des triplets sont manquants) et peut contenir des erreurs (des triplets faux sont présents). Dans ce mémoire de thèse, nous nous intéressons à la découverte de définitions, qui permettront de compléter DBpedia.

Précision

Extraction incorrecte du triplet

1. La valeur de l'objet n'est pas correctement extraite
2. La valeur de l'objet n'est pas complètement extraite
3. Le modèle de l'infobox n'est pas reconnu

Mauvais typage des données

4. Les données ne sont pas correctement typées

Relation implicite entre les prédicats

5. Un fait est encodé dans plusieurs prédicats
6. Plusieurs faits sont encodés dans un seul prédicat
7. La valeur d'un prédicat est calculée à partir de la valeur d'un autre prédicat

Pertinence

Information non pertinente extraite

8. Du contenu de mise en page Wikipédia est importé
9. Plusieurs prédicats portent la même information
10. Du contenu relatif aux illustrations sur Wikipédia est importé
11. D'autres informations non pertinentes sont importées

Consistance

Mauvaise représentation des valeurs numériques

12. Les valeurs numériques sont représentées de manière erronée ou inconsistante

Interconnexion

Liens externes

13. Liens vers des sites externes erronés, inaccessibles ou obsolètes

Alignement avec d'autres jeux de données incorrect

14. Liens avec Wikimedia
15. Liens avec Freebase
16. Liens avec Geospecies
17. Liens avec générés par Flickr

FIGURE 2.9 – Erreurs de conception dans DBpedia. D'après Zaveri, Kontokostas et al. (2013).

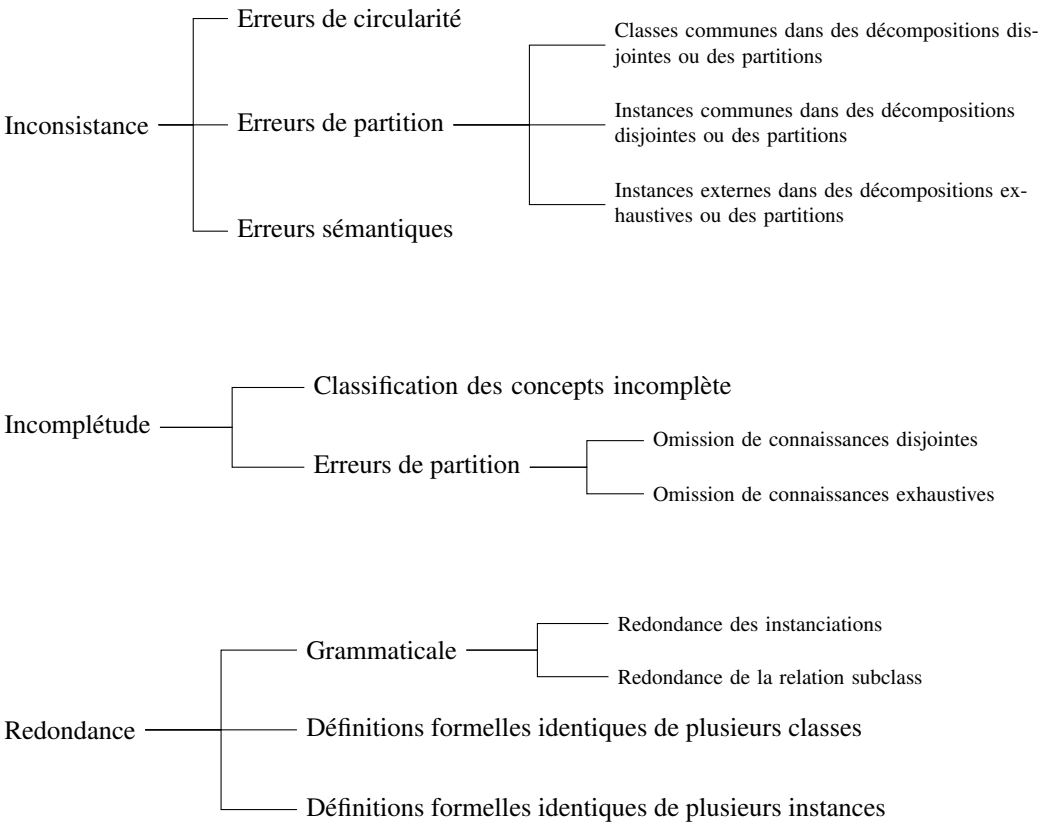


FIGURE 2.10 – Erreurs de conception de taxonomies. D'après Gómez-Pérez, Fernández-López et Corcho (2004).

Chapitre 3

Analyse formelle de concepts

Sommaire

L'analyse formelle de concepts (FCA), introduite par Ganter et Wille (1999), est un cadre mathématique qui, comme son nom l'indique, s'intéresse à la notion de concept. Elle repose notamment sur les travaux de Barbut et Monjardet (1970) sur les relations d'ordre ainsi que de Birkhoff (1940) sur les treillis. L'une des motivations des auteurs est de fournir un cadre formel à la notion de concept tel que conçu dans certaines théories de philosophie du langage, à savoir qu'un concept peut être défini de manière duale, soit en énumérant les « choses » qu'il désigne (définition en *extension*), soit en définissant le « sens » de ce concept (définition en *intension*).

3.1 Contextes et concepts

Les notions de contexte et de concept sont centrales en FCA. Le contexte représente la relation entre des objets et des attributs qualifiant ces objets.

Définition 2 (Contexte formel)**.** *Un contexte formel $K = (G, M, I)$ est constitué de deux ensembles G et M et d'une relation $I \subseteq G \times M$. Les éléments de G sont appelés les objets et les éléments de M sont appelés les attributs.*

Pour dire qu'un objet g est en relation I avec un attribut m, on écrit gIm ou $(g,m) \in I$. Cette relation se lit « l'objet g a l'attribut m ». Alternativement, on pourra dire que « l'attribut m qualifie l'objet g ».

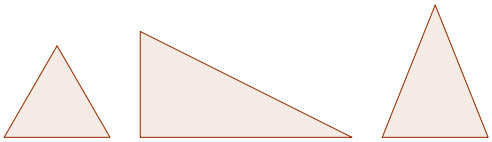


figure 1 figure 2 figure 3

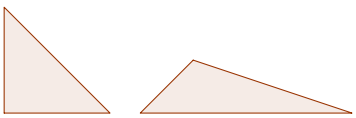


figure 4 figure 5

	a	b	c	d	e	f
	Triangle rectangle	Triangle isocèle	Triangle équilatéral	Triangle quelconque	Triangle	Carré
fig. 1		×	×		×	
fig. 2	×				×	
fig. 3		×			×	
fig. 4	×	×			×	
fig. 5				×	×	

FIGURE 3.1 – Exemple de contexte formel. Dans cet exemple, les objets correspondent à des figures géométriques et les attributs sont des caractéristiques de ces figures.

Usuellement, un contexte formel est représenté sous la forme d'un tableau binaire dont les lignes correspondent aux objets et les colonnes correspondent aux attributs. Une croix à l'intersection d'une ligne et d'une colonne signifie que l'objet de cette ligne possède l'attribut de cette colonne. La figure 3.1 présente un contexte formel. Celui-ci comporte 5 objets et 6 attributs. L'objet 2 a les attributs « Triangle rectangle » et « Triangle ».

À partir d'un contexte, il est intéressant de se demander quels attributs ont en commun certains objets, ou réciproquement, quels objets sont qualifiés par un ensemble d'attributs donnés. Cela est permis par les opérateurs de dérivation.

Définition 3 (Opérateurs de dérivation)**.** *Étant donné un contexte (G,M,I), les opérateurs de dérivation associés sont*

$$A^{\uparrow} = \{m \in M \mid gIm \quad \text{pour tout } g \in A\} \qquad (A \subseteq G)$$
$$B^{\downarrow} = \{g \in G \mid gIm \quad \text{pour tout } m \in B\} \qquad (B \subseteq M)$$

Usuellement, les deux opérateurs de dérivation sont dénotés $.'$ sans distinction. Ce sera également le cas dans cette thèse. Ces deux opérateurs forment une correspondance de Galois.

Définition 4 (Correspondance de Galois)**.** *Soient $\phi : E \to F$ et $\psi : F \to E$ des fonctions sur deux ensembles ordonnés $(E,\leqslant)$ et $(F,\leqslant)$. Ce couple de fonctions est appelé une* correspondance de Galois *entre les deux ensembles ordonnés si :*

1. $e_1 \leqslant e_2 \Rightarrow \phi(e_1) \geqslant \phi(e_2)$
2. $f_1 \leqslant f_2 \Rightarrow \psi(f_1) \geqslant \psi(f_2)$
3. $e \leqslant \psi(\phi(e))$ *et* $f \leqslant \phi(\psi(f))$

Les compositions $(.'')$ des opérateurs sont des opérateurs de fermeture vérifiant les propriétés d'extensivité ($X \subseteq X''$), de monotonie ($X \subseteq Y \Rightarrow X'' \subseteq Y''$) et d'idempotence ($X'' = (X'')''$). Les opérateurs de fermeture permettent de construire les concepts formels, qui sont les éléments clés de l'analyse formelle de concepts.

Définition 5 (Concept formel)**.** *Un concept formel du contexte (G,M,I) est une paire (A,B) avec $A \subseteq G$ et $B \subseteq M$ telle que $A' = B$ et $B' = A$. A est appelé l'extent et B est appelé l'intent du concept (A,B). L'ensemble des concepts formels est noté $\mathfrak{B}(G,M,I)$.*

Par exemple, les objets 3 et 4 du contexte de la figure 3.1 ont tous les deux les attributs « Triangle isocèle » et « Triangle », c.-à-d. $\{3,4\}' = \{b,e\}$. Inversement, on a $\{b,e\}' = \{1,3,4\}$. Ainsi, $(\{1,3,4\},\{be\})$ est un concept formel. L'intent et l'extent d'un concept correspondent respectivement à l'intension et à l'extension d'une définition.

3.2 Classes d'équivalences et treillis

3.2.1 Treillis

Dans cette section, nous présentons quelques éléments essentiels de la théorie des treillis, sur lesquels nous nous appuyons par la suite. Nous nous appuyons notamment sur les travaux de (Birkhoff, 1940) et de Davey et Priestley (2002).

Définition 6 (Infimum et supremum)**.** *Soit E un ensemble ordonné, et $F \subseteq E$. Un élément $x \in E$ est une* borne supérieure *de F si $f \leqslant x$ pour tout $f \in F$. De manière duale, un élément $x \in E$ est une* borne inférieure *de F si $f \geqslant x$ pour tout $f \in F$.*

Si la borne supérieure (resp. la borne inférieure) est unique, on parle de supremum *(resp.* infimum*), que l'on note $\bigvee F$ (resp. $\bigwedge F$).*

Définition 7 (Treillis)**.** *Un ensemble ordonné E, est un treillis si et seulement si pour tout $x,y \in E$, x et y ont un* infimum *et un* supremum*, c.-à-d. $x \bigwedge y$ et $x \bigvee y$ existent.*

L'ensemble des parties d'un ensemble, muni de la relation d'inclusion, forme un treillis. Dans ce cas, on a $X \bigwedge Y = X \cup Y$ et $X \bigvee Y = X \cap Y$. Sa représentation graphique est un diagramme de Hasse.

Par exemple, le diagramme de Hasse associé aux attributs de la figure 3.1 est un treillis qui est reproduit en figure 3.3. Pour des questions de place, l'attribut f n'a pas été représenté.

3.2.2 Treillis de concepts

Les concepts peuvent être ordonnés à partir de la relation d'inclusion sur leur intent ou sur leur extent.

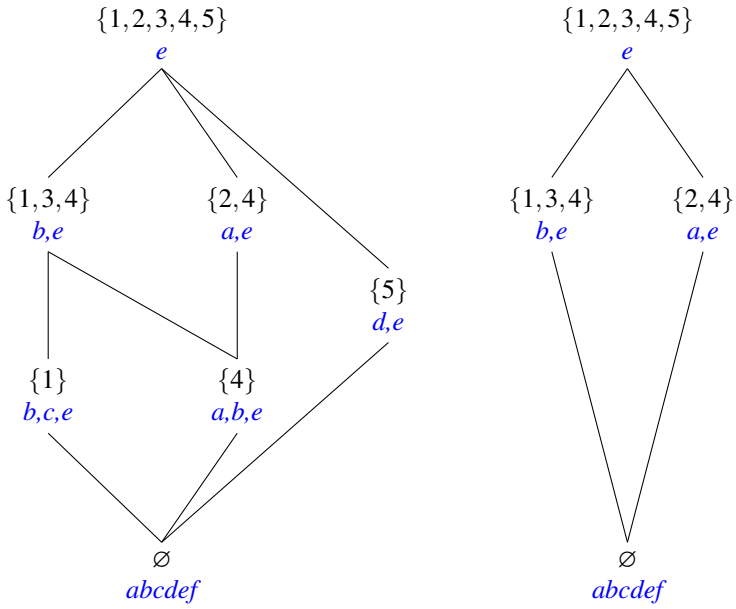
Définition 8 (Ordre partiel sur les concepts)**.** *Soient (A_1,B_1) et (A_2,B_2) deux concepts d'un contexte (G,M,I). (A_1,B_1) est un* sous-concept *de (A_2,B_2) si $A_1 \subseteq A_2$ (ou, de manière équivalente, $B_2 \subseteq B_1$). (A_2,B_2) est alors un* super-concept *de (A_1,B_1). La relation définit un ordre partiel sur les concepts.*

Cet ordre peut être utilisé pour créer un *treillis de concepts*.

Définition 9 (Treillis de concepts)**.** *L'ensemble des concepts $\mathfrak{B}(G,M,I)$ ordonnés selon l'ordre d'inclusion des intents définit un treillis de concepts, ou treillis de Galois, dénoté $\mathfrak{L}(G,M,I)$.*

Le treillis de concepts associé au contexte de la figure 3.1 est représenté en figure 3.2a. Nous remarquons par exemple que $(\{4\},abe) \subseteq (\{1,3,4\},be)$.

La visualisation d'un treillis de concepts peut vite être limitée par le nombre exponentiel de concepts à afficher, ce qui le rend illisible pour un analyste. En effet, pour un contexte de m objets et n attributs, il peut y avoir jusque $2^{\min(m,n)}$ concepts. Pour pallier cet effet, certains outils de visualisation permettent d'explorer le treillis localement, ou par niveaux. On peut cependant souhaiter une vue d'ensemble des données. Dans ce cas, une solution est de ne représenter que les concepts les plus fréquents, ce qui correspond à ne garder que la partie haute du treillis de concepts. Le nouveau treillis construit, nommé treillis iceberg par analogie avec la partie immergée des icebergs, ne contient alors que les concepts dont le nombre d'objets dans l'extent est supérieur à un seuil fixé. Par exemple, la figure 3.2b représente le treillis iceberg du contexte de la figure 3.1 en fixant le nombre minimal d'objets de l'extent à 2.



(a) Treillis de concepts (b) Treillis iceberg

FIGURE 3.2 – Le treillis de concepts associé au contexte de la figure 3.1.

3.2.3 Classes d'équivalence

Les opérateurs de fermeture définissent une relation d'équivalence sur l'ensemble des parties de G et de M d'un contexte (G,M,I). Deux ensembles $A_1, A_2 \subseteq G$ sont équivalents ssi $A_1'' = A_2''$. Ainsi, l'ensemble des sous-ensemble de G peut être partitionné en classes d'équivalence.

Définition 10 (Relation d'équivalence et classe d'équivalence)**.** *Une relation d'équivalence dans un ensemble E est une relation binaire $R \subseteq E \times E$ qui est réflexive, transitive et symétrique. Pour un ensemble E et une relation d'équivalence R, une classe d'équivalence de E est un sous-ensemble $F \subseteq E$ qui regroupe tous les éléments de E qui sont équivalents selon R.*

La figure 3.3 représente les classes d'équivalence des attributs du contexte de la figure 3.1. La classe d'équivalence en orange, par exemple, est composée des sous-ensembles d'objets $\{c\}$, $\{b,c\}$, $\{c,e\}$ et $\{b,c,e\}$, qui qualifient les objets 1, 3 et 4. Les bornes inférieures d'une classe d'équivalence sont appelées les *générateurs*. La borne supérieure est appelée le *fermé*. La classe d'équivalence en orange dans la figure 3.3 dispose d'un générateur ($\{c\}$) et d'un fermé ($\{b,c,e\}$). Le nom « fermé » vient du fait qu'il correspond à la fermeture de n'importe quel élément de sa classe d'équivalence : $\{3\}'' = \{1,3\}'' = \{1,4\}'' = \{3,4\}'' = \{1,3,4\}'' = \{1,3,4\}$. Il en résulte que la borne supérieure d'une classe d'équivalence est unique, ce qui se démontre par l'absurde : supposons que A_1 et A_2 soient deux ensembles incomparables tels que $A_1'' = A_2''$ et A_1 et A_2 sont des bornes supérieures. Alors $(A_1 \cup A_2)'' = A_1'' \cap A_2'' = A_1'' = A_2''$. Comme A_1 est une borne supérieure, on a $A_1 \cup A_2 \subseteq A_1$ et donc $A_2 \subseteq A_1$. De la même manière, $A_2 \cup A_1 \subseteq A_2$, d'où $A_1 \subseteq A_2$. On en déduit que $A_1 = A_2$, ce qui va a l'encontre de l'hypothèse de départ.

Au sein d'un contexte formel (G,M,I), si l'on considère l'ensemble des parties de G et la relation d'équivalence $.''$, l'ensemble des fermés correspond à l'ensemble des extents des concepts formels de (G,M,I). De façon similaire, les fermés de l'ensemble des parties de M correspondent à l'ensemble des

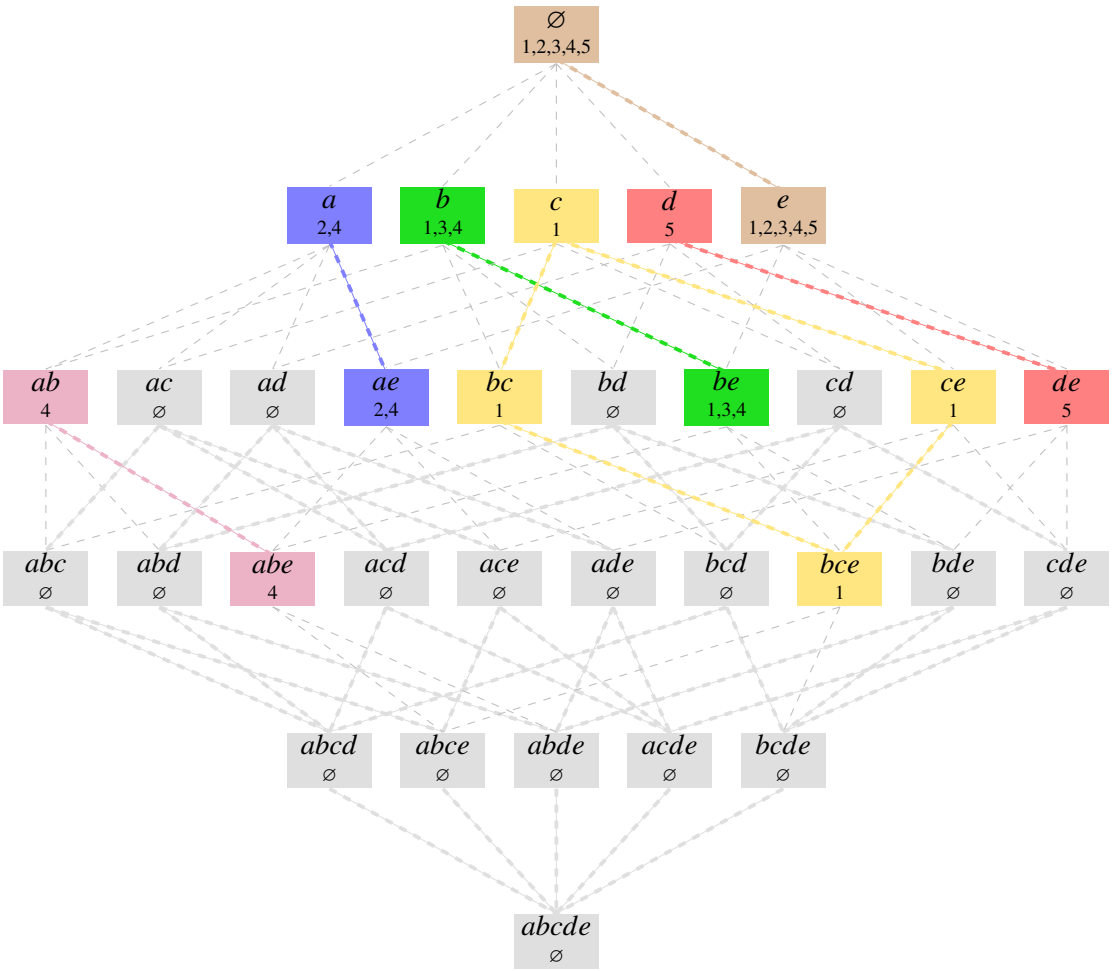


FIGURE 3.3 – Diagramme de Hasse (treillis de l'ensemble des parties) de l'ensemble $M = \{a, b, c, d, e\}$. Pour chaque sous-ensemble $B \subseteq M$, B' (c.-à-d. l'ensemble des objets qui ont tous les attributs de B) est noté en dessous. Les sous-ensembles de la même couleur constituent une même classe d'équivalence. Les traits pleins représentent une relation d'inclusion entre deux ensembles de la même classe d'équivalence, tandis que les traits en pointillés représentent une relation d'inclusion entre les ensembles de classes d'équivalence différentes.

intents.

3.3 Implications entre les attributs

Une implication entre deux ensembles d'attributs X et Y signifie que si un objet a tous les attributs de X, alors il a tous les attributs de Y.

Définition 11 (Implication)**.** *Étant donnés deux ensembles d'attributs $X, Y \subseteq M$, on dit que X implique Y ssi l'ensemble des objets qui ont tous les attributs de X ont également tous les attributs de Y. Autrement dit, $X' \subseteq Y'$. On note alors $X \Rightarrow Y$.*

Dans l'exemple des triangles, on a par exemple les implications $c \Rightarrow b$ et $ab \Rightarrow e$, c'est-à-dire

Triangle équilatéral ⇒ Triangle isocèle et Triangle rectangle, Triangle isocèle ⇒ Triangle.

Comme l'attribut Triangle qualifie tous les objets, on a également $\varnothing \Rightarrow e$. Il n'est pas nécessaire d'énumérer toutes les implications pour avoir une vue d'ensemble des données. Par exemple, si on connaît l'implication $ac \Rightarrow bde$, l'implication $abc \Rightarrow de$ n'apporte pas d'information supplémentaire. En fait, les classes d'équivalence permettent de ne construire que les implications de la forme générateur ⇒ fermé (nous reviendrons sur ce point dans le prochain chapitre). Cependant, cet ensemble d'implications n'est pas minimal. La base de Guigues-Duquenne, ou base canonique (Guigues et Duquenne, 1986), définit un ensemble minimal d'implications à partir desquelles toutes les autres implications peuvent être inférées. Afin de construire cette base, les auteurs introduisent la notion de pseudo-fermé.

Définition 12 (Pseudo-fermé)**.** *Un ensemble X est pseudo fermé ssi il n'est pas fermé et pour tout pseudo fermé Y contenu dans X, la fermeture de Y est contenue dans X. L'ensemble des pseudos-fermés est donc défini comme suit :*

$$PF = \{X \mid X \neq X'' \text{ et } \forall Y \in PF, Y \subset X \Rightarrow Y'' \subset X\}.$$

Dans l'exemple des triangles, les pseudo-fermés sont $\varnothing, \{c,e\}, \{a,d,e\}, \{b,d,e\}$ et $\{a,b,c,e\}$. Finalement, à partir de ces pseudo-fermés et de leur fermeture, nous pouvons construire la base de Guigues-Duquenne.

Définition 13 (Base de Guigues-Duquenne)**.** *La base canonique des implications est l'ensemble*

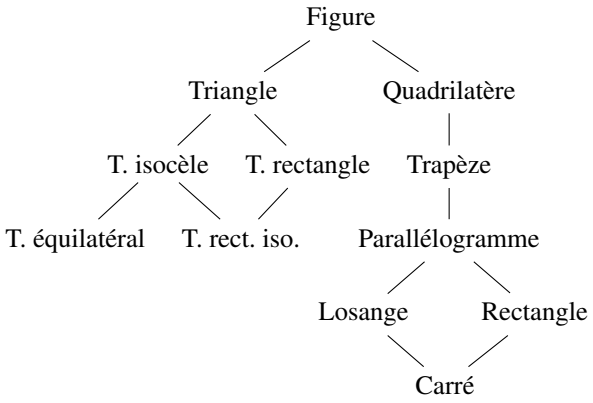
$$GD = \{X \Rightarrow (X''\backslash X) \mid X \text{ est un pseudo-fermé}\}.$$

Cette base est correcte, complète et non-redondante. Correcte, parce qu'elle ne contient que des implications valides ($\forall i \in GD, (G,M,I) \vDash i$) ; complète, car toutes les implications peuvent être dérivées ($(G,M,I) \vDash i \Rightarrow GD \vDash i$) et non-redondante, parce qu'aucune implication ne peut être retirée sans que la base ne reste correcte ou complète ($i \in GD \Rightarrow GD\backslash\{i\} \nvDash i$).

À partir des pseudo-fermés de notre exemple, les implications $\varnothing \Rightarrow e$ et $ce \Rightarrow b$ sont extraites, c'est-à-dire

$\varnothing$ ⇒ Triangle et Triangle, Triangle équilatéral ⇒ Triangle isocèle.

Aux pseudo-fermés de la classe d'équivalence de $abcde$ correspondent trois implications supplémentaires, $ade \Rightarrow bc$, $bde \Rightarrow ac$ et $abce \Rightarrow d$. Cependant la classe d'équivalence de $abcde$ correspond aux motifs qui ne sont présents dans aucun objet, ces implications ne sont donc pas générées.



(a) Connaissances du domaine

	T. équi.	T. Rect	Losange	Rectangle	Carré
Fig. 1	×				
Fig. 2		×			
Fig. 3			×		
Fig. 4				×	
Fig. 5					×

(b) Contexte

FIGURE 3.4 – Exemple de contexte avec des connaissances du domaine.

3.4 FCA en intégrant des connaissances du domaine

La FCA ne considère que la relation entre les objets et les attributs. Cependant, il arrive que l'analyste ait des connaissances supplémentaires qui pourraient être utiles à l'analyse mais qui ne peuvent pas être encodées dans le contexte. Il est fréquent que les attributs considérés ne soient pas totalement indépendants les uns des autres mais qu'ils soient structurés. Avec quelques connaissances de géométrie, par exemple, on peut avoir en tête une classification des figures telle que celle présentée en figure 3.4a.

Considérons le contexte de la figure 3.4b. Construire le treillis de concepts associé n'apporte rien d'intéressant, puisque la bijection entre objets et attributs génère un treillis plat (tous les concepts sont incomparables). Cependant, en intégrant nos connaissances sur les figures, un treillis plus intéressant peut être construit. Dans cette section nous présentons deux travaux qui étendent la FCA pour permettre de tenir compte d'une structure connue *a priori* sur les attributs.

3.4.1 Ajout de connaissances contextuelles

Dans le cadre de la recherche documentaire, Carpineto et Romano (2004) s'intéresse à la classification de documents. Dans leur approche, ils ont donc un ensemble d'objets qui sont des documents, et un ensemble d'attributs qui sont des termes faisant partie d'un thésaurus. Ces termes, qui sont assimilables à des classes de logiques de description, sont partiellement ordonnés par une relation de subsomption (c.-à-d. la relation « plus général que »). Les auteurs ont donc besoin de prendre en compte cette relation entre les attributs. À cette fin, ils proposent une méthode pour considérer des attributs qui font partie d'une taxonomie (c.-à-d. structurés en graphe orienté acyclique). Ils introduisent M^*, l'ensemble des attributs de la taxonomie et définissent un ordre sur M^* qui s'interprète comme une implication : $x \leqslant y$ signifie *avoir l'attribut x implique nécessairement d'avoir l'attribut y*. Pour paraphraser les auteurs, *chaque attribut implique chacun des attributs plus généraux que lui*.

Les auteurs définissent dans un deuxième temps l'intersection $\cap^*$ de deux ensembles d'attributs B_1 et B_2 comme l'ensemble des attributs les plus spécifiques de M^* qui sont plus généraux que B_1 et B_2. Ce nouvel opérateur leur permet de construire un treillis en tenant compte de la structure des attributs.

Dans la figure 3.4 par exemple, on a l'ensemble des attributs M ={T. équi., T. Rect, Losange, Rectangle, Carré} tandis que M^* est l'ensemble des nœuds du graphe des connaissances contextuelles.

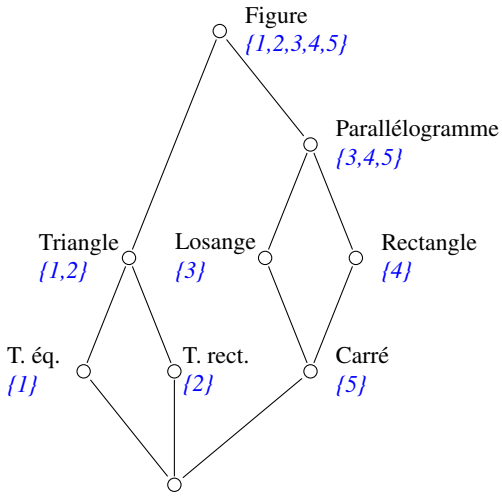


FIGURE 3.5 – Treillis de concepts construit en tenant compte des connaissances du domaine.

Concernant l'intersection $\cap^*$, on a par exemple :

$$\{\text{T. équilatéral}\} \cap^* \{\text{T. Rect}\} = \{\text{Triangle}\}$$
$$\{\text{Trapèze}\} \cap^* \{\text{Losange}\} = \{\text{Trapèze}\}$$
$$\{\text{Losange, Carré}\} \cap^* \{\text{Rectangle}\} = \{\text{Parallélogramme}\}$$

Le treillis résultant est représenté en figure 3.5.

3.4.2 *Pattern Structures*

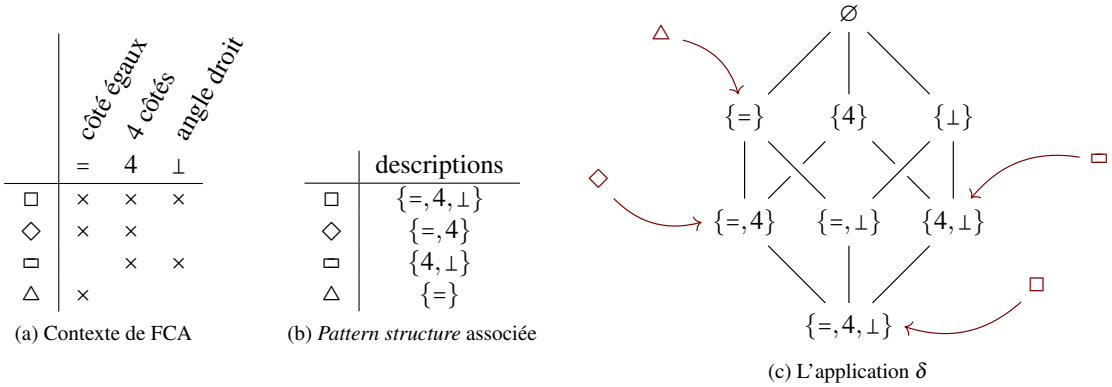
Les *pattern structures* (Ganter et Kuznetsov, 2001) sont une généralisation de la FCA permettant de traiter des données complexes. Contrairement à l'approche de Carpineto et Romano (2004), les *pattern structures* considèrent que l'ensemble des attributs est partiellement ordonné dans un demi-treillis inférieur. Cet ordre partiel est défini par une relation de subsomption, notée $\sqsubseteq$ et associé à une relation de similarité, notée $\sqcap$.

Définition 14 (*Pattern Structure*)**.** *Soient un ensemble d'objets G, un demi-treillis $(D, \sqcap)$, et $\delta : G \to D$ une application de G dans D. Alors $(G, (D, \sqcap), \delta)$ est une* pattern structure*. Les opérateurs de dérivations associés, notées $(.)^\square$, sont les suivantes :*

$$A^\square = \bigsqcap_{g \in A} \delta(g) \text{ pour tout } A \subseteq G;$$
$$d^\square = \{g \in G \mid d \sqsubseteq \delta(g)\} \text{ pour tout } d \in D.$$

$A^\square$ est appelé la description de A et $\sqcap$ est l'opérateur de similarité.

La FCA se transcrit aisément dans le formalisme des *pattern structures*. À un contexte (G, M, I) correspond la *pattern structure* $((G, (\mathcal{P}(M), \cap), \delta)$ où $\mathcal{P}(M)$ est l'ensemble des parties de M et δ est l'application qui à G associe l'ensemble G'. Autrement dit, la description d'un objet correspond à l'ensemble de ses attributs. La figure 3.6 illustre la représentation d'un contexte de FCA dans le formalisme des *pattern structures*. Le treillis représenté est $(\mathcal{P}(M), \cap)$ et les flèches correspondent à l'application δ.



(a) Contexte de FCA (b) *Pattern structure* associée (c) L'application δ

FIGURE 3.6 – Exemple de contexte de FCA dans le formalisme des *pattern structures*.

Comme en FCA, les compositions de ces applications sont des opérateurs de fermeture : étant donné un ensemble d'objets $A \subseteq G$, on a $A \subseteq A^{\square\square}$ et A est fermé quand $A = A^{\square\square}$. De façon similaire, $d \subseteq (D, \sqcap)$, $d \subseteq d^{\square\square}$ et d est fermé quand $d = d^{\square\square}$).

Un pattern concept [12] (A, d) vérifie $A^{\square} = d$ et $d^{\square} = A$ où A et d sont fermés. Étant donné un ensemble d'objets $A \in G$, $(A^{\square\square}, A^{\square})$ est un concept. De façon similaire, si $d \in (D, \sqcap)$ est un attribut, $(d^{\square}, d^{\square\square})$ est un concept. Un ordre partiel sur les concepts est défini de manière analogue à la FCA : $(A_1, d_1) \leqslant (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow d_2 \sqsubseteq d_1$. Cet ordre partiel permet de construire le treillis de concepts associé.

3.5 Extensions et applications de la FCA

3.5.1 Extensions de la FCA

Depuis son émergence dans les années quatre-vingt, plusieurs extensions ont été proposées pour généraliser la FCA. Certaines extensions permettent de considérer un plus grand nombre de dimensions, comme l'approche triadique (Lehmann et Wille, 1995), qui permet de considérer une dimension supplémentaire. Un contexte est alors un quadruplet (G,M,B,Y) où G est l'ensemble des objets, M est l'ensemble des attributs, B est l'ensemble des *valeurs* et la relation $Y \subseteq G \times M \times B$ s'interprète comme « l'objet g prend la valeur b dans la condition m. L'algorithme TRIAS (Jäschke et al., 2006) permet d'extraire les concepts triadiques. L'approche triadique a notamment été utilisée sur des *folksonomies* par Hotho et al. (2006). Une *folksonomie* est un système de ressources que les utilisateurs peuvent étiqueter avec des mots-clés. Les trois dimensions du contexte correspondent aux ressources R, aux utilisateurs U et aux mots-clés M. Si (r, u, m) appartient à la relation d'incidence (c.-à-d. il y a une croix à l'intersection de r, u et m), cela signifie que l'utilisateur u a étiqueté la ressource r avec le mot-clé m. Les concepts émergents permettent de mettre en avant, par exemple, des groupes d'utilisateurs qui étiquettent les même ressources, ou qui utilisent les mêmes mots-clés. L'approche polyadique (Voutsadakis, 2002) généralise la FCA en englobant l'approche triadique, puisqu'elle permet de considérer autant de dimensions que souhaité. L'algorithme DATA PEELER (Cerf et al., 2008) permet d'énumérer les concepts d'un contexte polyadique.

D'autres extensions permettent de prendre en compte des données relationnelles, notamment l'analyse relationnelle de concepts (RCA) (Rouane Hacene et al., 2013). La RCA prend en entrée un ensemble de contextes formels tels que définis en FCA, ainsi que des contextes relationnels qui permettent de définir des relations entre les objets. Le tout forme une « famille relationnelle de contextes » et permet de construire des familles de treillis.

12. Par la suite, nous utiliserons indifféremment le terme concept pour la FCA et les *pattern structures*.

Codocedo et Napoli (2014) introduisent la notion de *pattern structure* hétérogène. L'idée est de permettre l'apposition de différents contextes, offrant de considérer conjointement des valeurs booléennes et numériques, de la RCA et des *pattern structures*.

3.5.2 FCA et représentation des connaissances

Il existe de nombreux formalismes pour représenter les connaissances. Nous nous intéressons ici à trois d'entre eux : les graphes conceptuels, les logiques de description et les graphes RDF.

Dans une étude, Sertkaya (2010) s'intéresse aux travaux qui relient la FCA et les logiques de description (LD) et il distingue deux catégories de travaux. La première catégorie regroupe les travaux qui enrichissent la FCA en utilisant des constructeurs des LDs. On y retrouve notamment les travaux de Ferré et Ridoux (2000), qui introduisent l'analyse logique de concepts (*Logical Concept Analysis*, LCA). Dans cette extension, l'ensemble des attributs est remplacé par des formules logiques partiellement ordonnées dans un treillis et la relation d'incidence est remplacée par une application qui associe chaque objet à une formule logique qui le décrit.

Sertkaya (2010) remarque également la dualité entre les bases de connaissances et la FCA. Dans les bases de connaissances, les connaissances sont construites avec une approche *top-down* (c'est le cas, par exemple, lorsque des experts qui décrivent un domaine). Au contraire, avec la FCA, les connaissances sont découvertes avec une approche *bottom-up* : en partant des faits et en essayant de les généraliser. Une façon de tirer parti de la FCA est alors de construire de nouvelles connaissances de manière *bottom-up* afin de construire ou d'enrichir une base de connaissances. C'est notamment ce que proposent Völker et Niepert (2011), en présentant une méthode pour construire la TBox d'une base de connaissances à partir de sa ABox. Concrètement, les auteurs détaillent la construction de contextes formels à partir des connaissances d'une ABox. Les implications obtenues définissent les taxonomies de classes et de prédicats, ainsi que des propriétés sur les prédicats. Enfin, dans sa thèse, Distel (2011) détaille les liens entre FCA et LDs et propose d'utiliser la FCA pour compléter des bases de connaissances.

Plusieurs travaux se sont intéressés à l'utilisation de la FCA sur des graphes conceptuels (Sowa, 2008). Les graphes conceptuels sont une représentation des connaissances sous forme de graphe orienté biparti. Les nœuds peuvent être des classes ou des relations. Contrairement aux graphes RDF, les graphes conceptuels permettent de représenter des relations n-aires. Andrews et Polovina (2011) proposent un algorithme pour convertir un graphe conceptuel en contexte formel. Ferré (2015) introduit l'extension Graph-FCA, permettant de construire des treillis de concepts à partir de graphes conceptuels. Pour cela, il définit des *projected graph patterns*, c'est-à-dire des concepts formels dont l'intension est un ensemble de variables, et l'extension est un ensemble de relation n-aires. Kötters (2015) propose une approche similaire pour les graphes RDF : les intents des concepts correspondent aux BGP d'une requête SPARQL, et les extents correspondent aux sous-graphes correspondants.

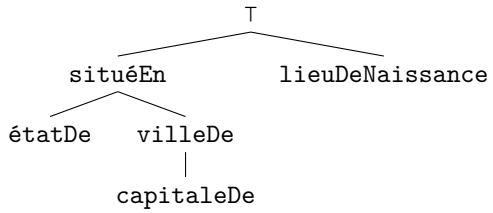
3.5.3 Classification de ressources RDF

Dans cette section, nous présentons plus en détails le travail de Alam et al. (2015), sur lequel nous nous appuyons. Dans ce travail, les auteurs proposent de classifier des ressources RDF au sein d'un treillis de Galois en s'appuyant sur les *pattern structures* et le travail de Carpineto et Romano (2004). Dans un premier temps, les auteurs définissent un ordre partiel sur les paires (prédicat, objet) :

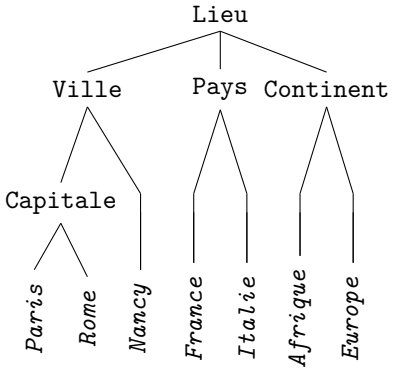
$$(p_i, o_i) \sqsubseteq (p_j, o_j) \Leftrightarrow p_i = p_j, C(o_i)\ \texttt{rdfs:subClassOf}\ C(o_j)$$

```
Paris villeDe Europe
Paris capitaleDe France
Nancy situéEn Europe
Nancy villeDe France
Rome capitaleDe Italie
Rome situéEn Europe
```

(a) Triplets RDF (ABox)



(b) Arbre des prédicats



(c) Arbre des classes et instances

FIGURE 3.7 – Exemple de base de connaissance. La figure 3.7a représente la ABox, c'est-à-dire un ensemble de faits. La figure 3.7b représente l'ordre partiel des prédicats induit par la relation `rdfs:subPropertyOf` tandis que la figure 3.7c représente l'ordre partiel sur les classes induit par la relation `rdfs:subClassOf`, ainsi que les instances. Ces deux ordres constituent la TBox.

où $C(o)$ est la classe de la ressource o. La similarité entre deux paires (prédicat, objet) est alors définie comme suit :

$$(p_i,o_i)\sqcap(p_j,o_j) = \begin{cases}(p_i,\mathrm{lcs}_C(C(o_i),C(o_j))) & \text{si } p_i = p_j,\\ \varnothing & \text{sinon.}\end{cases}$$

Étant donné un ensemble de triplets K la *pattern structure* est alors définie comme suit :

$$\begin{aligned}G &= \{s \mid \langle s,p,o\rangle \in K\}\\ D &= \{(p,o) \mid \langle s,p,o\rangle \in K\} \cup \{(p,c) \mid \langle s,p,o\rangle \in K, (p,o) \sqsubset (p,c)\}\\ \delta(g) &= \{(p,o) \mid \langle g,p,o\rangle \in K\}\end{aligned}$$

Les opérateurs de dérivation sont les suivants :

$$A^{\square} = \bigsqcap_{g\in A} \delta(g) \text{ pour tout } A \subseteq G \quad \text{et} \quad d^{\square} = \{g \in G \mid d \sqsubseteq \delta(g)\} \text{ pour tout } d \in D.$$

L'ensemble des descriptions est composé des paires (prédicat, objet) présentes dans les triplets auxquelles s'ajoutent toutes les paires plus générales au regard de l'ordre introduit.

Considérons par exemple la base de connaissances en figure 3.7. Les descriptions des ressources `Paris`, `Rome` et `Nancy` sont

$$\begin{aligned}\delta(Paris) &= \{(villeDe, Europe), (capitaleDe, France)\}\\ \delta(Rome) &= \{(situéEn, Europe), (capitaleDe, Italie)\}\\ \delta(Nancy) &= \{(situéEn, Europe), (villeDe, France)\}\end{aligned}$$

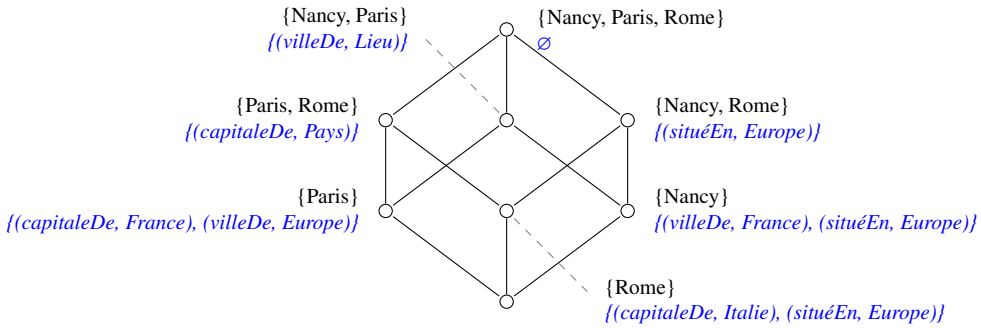


FIGURE 3.8 – Treillis de concepts obtenu à partir de l'exemple de la figure 3.7.

Les similarités de `Paris` avec `Rome` et `Nancy` sont alors

$$\delta(Paris) \sqcap \delta(Rome) = \{(capitaleDe, Pays)\} \text{ et}$$
$$\delta(Paris) \sqcap \delta(Nancy) = \{(villeDe, Lieu)\}.$$

Le treillis de concepts résultant est présenté à la figure 3.8.

Synthèse du chapitre

Dans ce chapitre, nous avons présenté l'analyse formelle de concepts (FCA). Cette approche mathématique permet de classifier des objets en fonctions de caractéristiques (les *attributs*) qu'ils partagent. Nous avons montré dans ce chapitre comment appliquer la FCA à un ensemble de triplets RDF. Les *pattern structures*, qui sont une extension de la FCA permettent de tenir compte de la hiérarchie de classes entre les attributs.

Chapitre 4

Motifs et règles d'association

Sommaire

La fouille de motifs et de règles d'association sont deux processus principaux de la fouille de données. Pour commencer, ce chapitre aborde la fouille de motifs et les règles d'association. Le nombre de règles d'association pouvant augmenter de façon exponentielle par rapport aux données initiales, nous présentons dans ce chapitre deux types de méthodes pour réduire le nombre de règles extraites. Nous détaillons ensuite quelques algorithmes fréquemment utilisés, notamment ECLAT, qui est à la base de l'algorithme que nous employons. Enfin, plusieurs applications liées au web des données sont présentées.

4.1 Motifs

Malgré les liens forts entre la FCA et la fouille de motifs, la fouille de motifs a émergé de manière relativement indépendante de la FCA. La FCA considère des objets et des attributs et s'appuie sur un contexte, qui est une représentation tabulaire des données (fig. 4.1a). La fouille de motifs, en revanche, parle usuellement de *transactions* et d'*itemsets* dans une base de données transactionnelle, qui utilise une représentation horizontale des données (fig. 4.1b). Un *itemset*, ou *motif* est donc un ensemble d'attributs qui se retrouvent simultanément dans plusieurs transactions. Dans ce chapitre, on utilisera sans distinction les termes *itemset* et motif comme des ensembles d'attributs. Le support d'un motif est l'ensemble des transactions dans lesquelles se retrouvent les attributs du motif. En général, la fouille de motifs s'intéresse aux *motifs fréquents*.

	a	b	c	d	e
1	×	×			
2		×	×	×	
3				×	×
4			×		
5	×		×		
6	×	×	×	×	×

(a) Représentation tabulaire

1	a, b
2	b, c, d
3	d, e
4	c
5	a, c
6	a, b, c, d, e

(b) Représentation horizontale

a	1, 5, 6
b	1, 2, 6
c	2, 4, 5, 6
d	2, 3, 6
e	3, 6

(c) Représentation verticale

FIGURE 4.1 – Plusieurs façon de représenter les données. La FCA s'appuie sur une représentation tabulaire (4.1a) tandis que la fouille de motif s'appuie usuellement d'une représentation horizontale (4.1b). Les algorithmes ECLAT et CHARM s'appuient sur une représentation verticale des données (4.1c).

Définition 15 (Motif fréquent)**.** *Un motif est* fréquent *ssi le cardinal de son support est supérieur à un seuil fixé. L'ensemble des motifs fréquents, que l'on notera $\mathcal{F}$, est*

$$\mathcal{F} = \{X \mid |\text{support}(X)| \geqslant \sigma\}$$

Dans les données de la figure 4.1 par exemple, pour un seuil fixé à $\sigma = 2$, l'*itemset* $\{b,c,d\}$ est fréquent car il est présent dans les transactions 2 et 6. En revanche, l'*itemset* $\{c,d,e\}$ n'est présent que dans la transaction 6 et n'est donc pas fréquent.

Les premiers travaux concernant la fouille de motifs s'employaient à énumérer tous les motifs fréquents. Cependant, devant le nombre exponentiel de motifs (pour n attributs, on peut avoir jusqu'à 2^n motifs), des stratégies ont été proposées pour ne garder que certains *itemsets*. Il a donc fallu mettre au point des critères pour déterminer si un motif devait ou non être conservé. Une partie des travaux sur le sujet s'appuie sur la FCA, en tirant notamment parti des classes d'équivalence. En particulier, les générateurs et les fermés, qui correspondent respectivement aux éléments minimaux et maximaux des classes d'équivalence.

Définition 16 (Générateur)**.** *Un motif est un* générateur *s'il n'existe pas d'itemset plus petit ayant le même support. Pour un contexte (G,M,I), l'ensemble des générateurs, que l'on notera $\mathcal{G}$, est*

$$\mathcal{G} = \{X \subseteq M \mid \nexists\, Y \subset X, \text{support}(Y) = \text{support}(X)\}.$$

Définition 17 (Motif fermé)**.** *Du point de vue de la fouille de motifs, un* itemset *est un motif fermé s'il n'existe pas d'itemset plus grand ayant le même support. Pour un contexte (G,M,I), l'ensemble $\mathcal{C}$ des fermés est défini par*

$$\mathcal{C} = \{X \subseteq M \mid \nexists\, Y \supset X, \text{support}(Y) = \text{support}(X)\}.$$

Par exemple, l'*itemset* $\{d,e\}$ est un motif fermé dans les données de la figure 3. Un motif fermé correspond à l'intent d'un concept en FCA. Avec $\sigma = 0$, l'ensemble des motifs fermés correspond à l'ensemble des intents du treillis de concepts associé. En ne conservant que les motifs fermés et fréquents, on obtient alors un treillis iceberg tel que défini dans la section 3.2.2.

4.2 Règles d'association

Les règles d'association ont été introduites simultanément par Agrawal, Imieliński et Swami (1993) et Klemettinen et al. (1994). Cependant, des travaux antérieurs existent, notamment ceux de Luxenburger

(1991), qui parle alors d'implications partielles. L'objectif de la fouille de règles d'association est de trouver des dépendances entre les attributs, c'est-à-dire des observations du type « Lorsqu'un objet possède tels attributs, il possède également tels autres attributs ».

4.2.1 Définition

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 ». Dans cette règle, X est appelé la *prémisse*, et Y la *conséquence*. Le support de cette règle correspond aux objets qui supportent à la fois la prémisse X et la conséquence Y :

$$\text{support}(X \to Y) = X' \cap Y'$$

où $.'$ correspond à la correspondance de Galois. Le cardinal du support sera régulièrement utilisé par la suite. Il sera noté $|\text{support}(X \to Y)|$ ou $|X' \cap Y'|$ et, lorsque cela ne génère pas d'ambiguïté, il sera également nommé support. Une règle $X \to Y$ sera dite fréquente si, pour un seuil σ fixé, $|X' \cap Y'| \geqslant \sigma$.

La *confiance* est une mesure de qualité des règles d'association. Elle correspond à la proportion d'objets qui supportent Y parmi ceux qui supportent X. Elle peut être vue comme la probabilité conditionnelle $P(Y \mid X)$.

$$\text{conf}(X \to Y) = \frac{|X' \cap Y'|}{|X'|} \tag{4.1}$$

Une règle d'association est *valide* si sa confiance est supérieure à un seuil θ défini par l'analyste. Si la confiance vaut 1, il s'agit alors d'une *implication* notée $X \Rightarrow Y$. La confiance n'est pas une mesure symétrique : la règle $X \to Y$ peut être valide sans que la règle $Y \to X$ ne le soit.

Comme pour la fouille de motifs, le grand nombre de règles d'association retournées est une des limitations de cette approche. Aussi, de nombreux travaux se sont penchés sur des méthodes pour réduire le nombre de règles obtenues. Ces travaux se divisent en deux catégories, détaillées dans les sous-sections suivantes. Une première partie des travaux s'intéresse à l'utilisation de mesures pour sélectionner les règles d'association. À partir d'une mesure donnée, seules les règles au-dessus d'un certain seuil sont conservées. La seconde partie des travaux construit une base de règles, c'est-à-dire un sous-ensemble de règles qui soit suffisamment représentatif de l'ensemble de départ.

4.2.2 Mesures sur les règles d'association

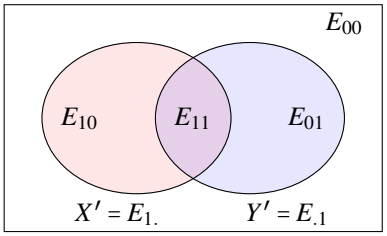
En préambule de cette section, nous introduisons une partition sur les objets qui sera utilisée par la suite. Pour une règle d'association $X \to Y$, l'ensemble des transactions peut être partitionné en fonction des supports de X et Y. Il y a quatre parties à cette partition, notées E_{ij} avec $i, j \in \{0, 1\}$, selon que les objets supportent X ou Y. Par exemple, $E_{11}(X \to Y)$ représente l'ensemble des objets qui supportent X et Y, tandis que $E_{10}(X \to Y)$ représente l'ensemble des objets qui supportent X mais pas Y. Ainsi, $X' = E_{11} \cup E_{10}$, que l'on pourra abréger $E_{1.}$ par la suite. Une représentation schématique de la partition obtenue ainsi qu'un exemple sont présentés à côté de la table 4.1.

Outre le support et la confiance, de nombreuses mesures ont été proposées pour évaluer les règles d'association. Seules quelques unes sont présentées ici, pour une étude plus développée, voir par exemple le livre de Guillet et Hamilton (2007) ou l'article de P.-N. Tan, V. Kumar et Srivastava (2002). La table 4.1 synthétise les mesures présentées dans cette sous-section.

L'une des limitations de la confiance est qu'elle ne tient pas compte du support de la conséquence. Cela implique que si le support de la conséquence est élevé, la confiance envers la règle sera élevée, même si X et Y ne sont pas liés. Pour pallier cela, la conviction (Brin, Motwani, Ullman et al., 1997) propose de tenir compte du support de $\neg Y$.

TABLE 4.1 – Mesures sur les règles d'association. Nous avons également ajouté le coefficient de Jaccard, qui sera traité dans le prochain chapitre. Le schéma à droite représente la partition des objets pour une règle d'association $X \to Y$.

textrmMesure	Formule	Domaine	Symétrique
Support	$\lvert E_{11}\rvert$	$[0;1]$	Oui
Confiance	$\frac{\lvert E_{11}\rvert}{\lvert E_{1.}\rvert}$	$[0;1]$	Non
Conviction	$\frac{\lvert E_{1.}\rvert \times \lvert E_{.0}\rvert}{\lvert E_{10}\rvert}$	$[0;+\infty[$	Non
Intérêt/Lift	$\frac{\lvert E_{11}\rvert}{\lvert E_{1.}\rvert \times \lvert E_{.1}\rvert}$	$[0;+\infty[$	Oui
Nouveauté	$\lvert E_{11}\rvert - \lvert E_{1.}\rvert \times \lvert E_{.1}\rvert$	$]-1;1[$	Oui
Jaccard	$\frac{\lvert E_{11}\rvert}{\lvert E_{11}\rvert + \lvert E_{10}\rvert + \lvert E_{01}\rvert}$	$[0;1]$	Oui



Exemple pour la règle $a \to b$ (fig. 4.1)

$E_{1.} = a' = \{1,5,6\}$
$E_{.1} = b' = \{1,2,6\}$
$E_{11} = a' \cap b' = \{ab\}' = \{1,6\}$
$E_{10} = a' \backslash b' = \{5\}$
$E_{01} = b' \backslash a' = \{2\}$
$E_{00} = G \backslash (a' \cup b') = \{3,4\}$

L'intérêt ou *lift* (Brin, Motwani et Silverstein, 1997 ; IBM, 2018) a été proposé pour estimer la dépendance entre deux ensembles d'attributs. Contrairement à la confiance, cette mesure est symétrique. Si X et Y sont incompatibles (c.-à-d. n'apparaissent pas ensemble dans les données), alors le *lift* prend la valeur nulle. Un *lift* de 1 signifie que X et Y sont indépendants. Au delà de 1, Plus le *lift* est grand, plus X et Y sont dépendants. La *nouveauté* (Piatetsky-Shapiro, 1991) s'apparente au *lift*, mais prend ses valeurs dans $]-1;1[$. C'est-à-dire qu'au lieu de considérer la proportion d'objets qui ont à la fois X et Y parmi les objets qui ont soit X soit Y comme le fait le *lift*, la nouveauté considère la différence entre les objets qui ont à la fois X et Y et les objets qui ont soit X soit Y.

4.2.3 Familles de règles d'association

Le terme de « famille » est emprunté à Szathmary (2006), dont la thèse détaille en profondeur les différentes bases de règles d'association et leurs relations. Les bases de règles, comme pour la base de Duquenne-Guigues, s'appuient sur les propriétés du treillis des attributs et des classes d'équivalence. La figure 4.2 représente les différentes bases de règles évoquées et leur relation.

Bastide et al. (2000) introduisent la base générique des implications. Cette base retourne plus d'implications que la base de Duquenne-Guigues, mais elle a l'intérêt d'être plus facile à obtenir : il suffit de connaître les générateurs et les fermés, et il n'y a pas besoin de calculer les pseudo-fermés.

Définition 18. *La base générique des implications est l'ensemble des implications construites à partir d'un générateur et de sa fermeture.*

$$\mathcal{GB} = \{X \Rightarrow (X'' \backslash X) \mid X \text{ est un générateur et } X \neq X''\}.$$

Toutes les implications peuvent être retrouvées à partir de cette base.

Bastide et al. (ibid.) introduisent également la base informative, qui limite l'ensemble des règles extraites à celles dont l'antécédent X est un générateur et la conséquence Y est un fermé. La fermeture du

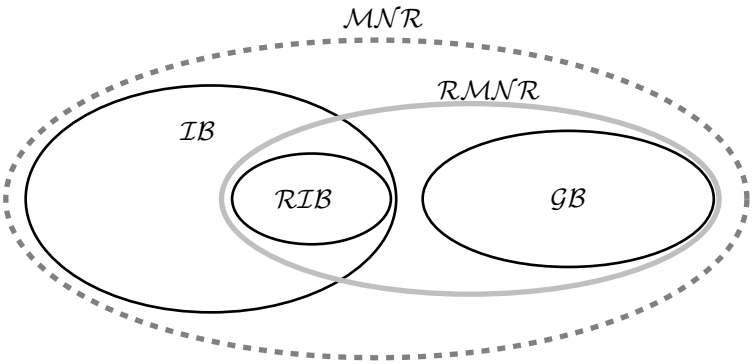


FIGURE 4.2 – Relations entre les différentes bases de règles. La base informative ($\mathcal{IB}$) et la base générique ($\mathcal{GB}$) forment l'ensemble minimal des règles non redondantes, noté $\mathcal{MNR}$, tandis que la base $\mathcal{RIB}$ et la base générique forment la réduction transitive de l'ensemble minimal des règles non redondantes, notée $\mathcal{RMNR}$.

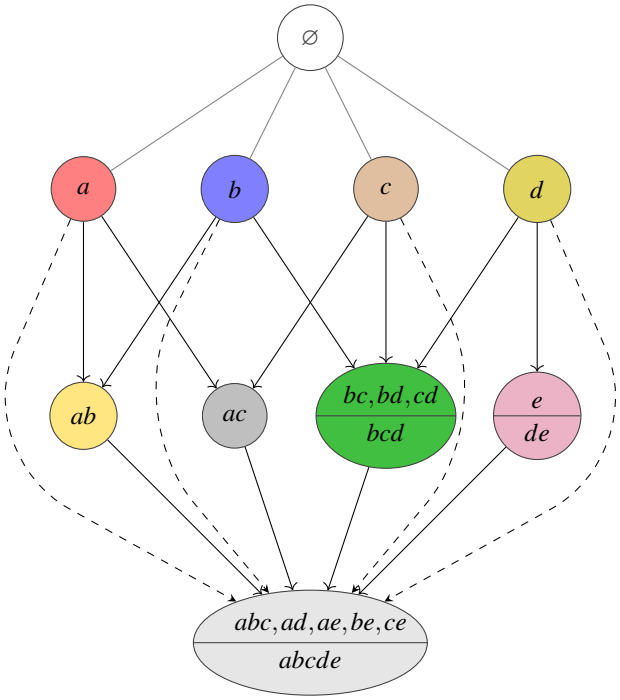


FIGURE 4.3 – Treillis des générateurs et fermés construit à partir du contexte de la figure 4.1. Un nœud représente une classe d'équivalence, avec ses générateurs en haut et ses fermés en bas. Si la classe d'équivalence ne contient qu'un élément (c.-à-d. le générateur et le fermé sont identiques), il est noté au milieu du nœud. Les flèches pleines représentent des règles de $\mathcal{RIB}$ et les flèches en pointillé indiquent les règles qui sont dans $\mathcal{IB}$ mais pas dans $\mathcal{RIB}$.

TABLE 4.2 – Les générateurs et les fermés, groupés par classe d'équivalence, ainsi que les différentes bases de règles extraites à partir du contexte de la figure 4.1.

Classes d'équivalence

$\mathcal{G}$	$\mathcal{C}$
abc ad ae be ce	abcde
bc bd cd	bcd
e	de
ab	
ac	
a	
b	
c	
d	

Implications

$\mathcal{GB}$
abc → de
ad → bce
ae → bcd
be → acd
ce → abd
bc → d
bd → c
cd → b
e → d

$\mathcal{GD}$
ad → bce
bcde → a
bc → d
bd → c
cd → b
e → d

Règles d'association

$\mathcal{IB}$
bc → ade
bd → ace
cd → abe
e → abcd
ab → cde
ac → bde
a → bcde
b → acde
c → abde
d → abce
a → b
a → c
b → a
b → cd
c → a
c → bd
d → e
d → bc

$\mathcal{RIB}$
bc → ade
bd → ace
cd → abe
e → abcd
ab → cde
ac → bde
a → b
a → c
b → a
b → cd
c → a
c → bd
d → e
d → bc

générateur doit être strictement incluse dans le fermé, ce qui signifie que la classe d'équivalence de X est « au dessus » de la classe d'équivalence de Y dans la représentation usuelle du treillis d'attributs (voir la figure 4.3).

Définition 19 (Base informative)**.** *La base informative des règles d'association est l'ensemble des règles :*

$$\mathcal{IB} = \{r : X \to (Y \backslash X) \mid Y \in \mathcal{C} \wedge X \in \mathcal{G} \wedge X'' \subset Y\}.$$

Par exemple, dans la figure 4.3, d est un générateur, et de un fermé. On a $d'' = d \subseteq de$ donc $d \to e$ est une règle d'association de $\mathcal{IB}$, que l'on retrouve dans la table 4.2. La base informative et la base générique forment l'ensemble minimal des règles non redondantes, noté $\mathcal{MNR}$.

La réduction transitive de la base informative, notée $\mathcal{RIB}$, permet de réduire la base informative en retirant les règles d'association qui peuvent être découvertes par transitivité.

Définition 20 (Réduction transitive de la base informative)**.** *La réduction transitive de la base informative des règles d'association est l'ensemble des règles :*

$$\mathcal{RIB} = \{r : X \to (Y \backslash X) \in \mathcal{IB} \mid X'' \textit{ est un sous-ensemble strict maximal de } Y \in \mathcal{C}\}.$$

Par exemple, les règles $a \to b$, $a \to bcde$ et $ab \to cde$ appartiennent à $\mathcal{IB}$. La règle $a \to bcde$ ne sera pas dans $\mathcal{RIB}$. Tout comme la base $\mathcal{IB}$, la base $\mathcal{RIB}$ permet de retrouver l'ensemble des règles d'association. Toutefois, il est beaucoup moins aisé de retrouver la confiance d'une règle par la suite. La base $\mathcal{RIB}$ et la base générique forment la réduction transitive de l'ensemble minimal des règles non redondantes, notée $\mathcal{RMNR}$. Les règles d'association de $\mathcal{IB}$ et $\mathcal{RIB}$ sont illustrées en figure 4.3.

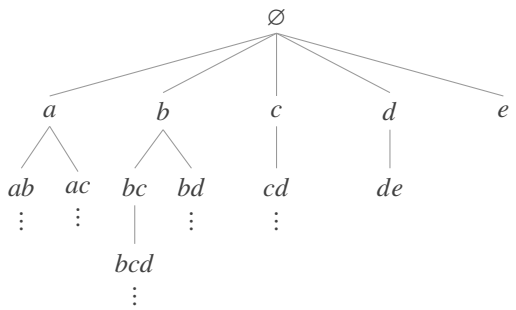


FIGURE 4.4 – Espace de recherche du treillis de la figure 3.3. Lorsqu'aucun des fils d'un nœud ne correspondent à un *itemset* fréquent, ceux-ci ne sont pas représentés et le symbole ⋮ est ajouté sous le nœud.

4.3 Algorithmes de fouille

Dans cette section, nous présentons les algorithmes permettant l'extraction de motifs fréquents et de règles d'associations. Au cours du temps, de nombreux travaux ont généralisé la fouille de motifs, permettant d'extraire des motifs sur des structures plus complexes que des ensembles, comme les séquences (Yan, Han et Afshar, 2003) ou les graphes (Cheng, Yan et Han, 2014). La question de la fouille de données structurées a notamment été traitée par Nijssen (2006). Ici, nous nous intéressons aux motifs uniquement sous la forme d'*itemsets*.

4.3.1 Vue d'ensemble

La fouille de règles d'association se décompose en deux étapes. La première étape est l'énumération de tous les motifs fréquents, et éventuellement fermés selon l'algorithme. Celle-ci est accomplie en explorant le treillis et en élaguant les branches lorsque le support d'un motif est en dessous d'un seuil donné. Selon les stratégies, le treillis peut être exploré en largeur ou en profondeur. Pour un parcours en largeur, tous les attributs fréquents sont extraits. Puis, à partir des attributs extraits, tous les motifs de taille 2 sont générés, et ainsi de suite jusqu'à ce que tous les motifs fréquents soient énumérés. Sur l'espace de recherche représenté figure 4.4, un parcours en largeur permet de découvrir les motifs dans l'ordre suivant : $a, b, c, d, e, ab, ac, bc, bd, cd, de, bcd$. Cette stratégie de recherche est notamment adoptée par l'algorithme A PRIORI (Agrawal et Srikant, 1994). Pour un parcours en profondeur, un attribut fréquent est choisi puis étendu par l'ajout successif d'un nouvel attribut jusqu'à ce que le motif obtenu ne soit plus fréquent. À ce moment, le motif est ajouté à l'ensemble des motifs fréquents et le processus reprend à partir du même motif, duquel est retiré le dernier attribut ajouté. Les motifs de la figure 4.4, sont découverts dans l'ordre suivant : $a, ab, ac, b, bc, bcd, bd, c, cd, d, de, e$. Cette stratégie est notamment utilisée par l'algorithme ECLAT (Zaki, 2000). Pour décider de quel nœud étendre à une profondeur donnée, c'est-à-dire pour décider qui de a ou b étendre en premier, plusieurs stratégies sont possibles. Les plus fréquentes sont l'ordre lexicographique et l'ordre par support décroissant. La seconde étape consiste à générer les règles à partir des motifs extraits à la première étape.

4.3.2 Algorithmes d'énumération de motifs fréquents

L'algorithme ECLAT

L'algorithme ECLAT (ibid.), tire son nom de *Equivalence Class Clustering and bottom-up Lattice Traversal*. Au lieu de considérer une représentation horizontale des données, l'algorithme s'appuie sur une représentation verticale (voir la figure 4.1c). C'est-à-dire qu'il considère les attributs plutôt que les

objets. Cette approche a l'avantage de ne parcourir la base de données qu'une fois : une fois les attributs et leur support en mémoire, tous les motifs sont extraits par intersections ou unions d'ensembles. L'énumération se fait en parcourant en profondeur l'ensemble des ensembles d'attributs. L'algorithme 1 détaille le fonctionnement d'ECLAT.

Dans un premier temps, tous les attributs sont extraits ainsi que leur support (ligne 4). Ils constituent les nouveaux nœuds de l'arbre, juste sous la racine. La méthode ECLAT-EXTEND étend alors un premier nœud X_i. Ses enfants sont générés en considérant les frères X_j du nœud X_i (ligne 9). Le nouveau motif correspond à l'union de X_i et de X_j (ligne 10). Le support est l'intersection des supports de X_i et de X_j (ligne 11). Si le support du nouveau motif est supérieur au seuil fixé, alors il est ajouté à la liste des motifs fréquents (lignes 13-14), et il est à son tour étendu (ligne 15).

Algorithme 1 : ECLAT

Données : $I \subseteq G \times M$, θ : seuil de support
Résultat : F : ensembles des motifs fréquents

```
1  ECLAT (I, θ)
2      F ← ∅ // Initialisation de l'ensemble des motifs fréquents
3      Racine ← (∅, G) // La racine est le motif vide, présent dans tous les objets
4      Noeuds ← {(m, m') | m ∈ M, |m'| ⩾ θ};
5      ECLAT-EXTEND(Noeuds, F);
6  ECLAT-EXTEND (Noeuds, F)
7      pour chaque (X_i, X'_i) ∈ Noeuds faire
8          NouveauNoeud ← ∅;
9          pour chaque (X_j, X'_j) ∈ Noeuds, with j > i faire
               // Seuls les noeuds pas encore étendus sont considérés
10             A ← X_i ∪ X_j // Nouveau motif
11             B ← X'_i ∩ X'_j // Ensemble des objets qui supportent motif
12             si |B| ⩾ θ alors
13                 NouveauNoeud ← (A, B) ;
14                 F ← F ∪ NouveauNoeud // Ajout du motif à l'ensemble des motifs fréquents.
15                 ECLAT-EXTEND(NouveauNoeud, F)
16             fin
17         fin
18     fin
```

L'algorithme CHARM

Par la suite, CHARM (Zaki et Hsiao, 2002) a été proposé afin de ne tenir compte que des ensembles fermés. La stratégie de parcours reste la même, mais les motifs qui ne sont pas fermés sont éliminés au fur et à mesure. Algorithmiquement parlant, cela revient à remplacer les lignes 14-15 d'ECLAT par CHARM-PROPERTY (algorithme 2). Au lieu d'ajouter directement un motif fréquent à l'ensemble des motifs à retourner, CHARM vérifie que seuls les fermés sont ajoutés. Si $X'_i = X'_j$, cela veut dire que X_i et X_j appartiennent à la même classe d'équivalence, et que le fermé de leur classe d'équivalence contient $X_i \cup X_j$. Aussi, tous les motifs trouvés jusque là contenant X_i peuvent être étendus par X_j, et X_j peut être retiré des motifs fréquents puisque $X_i \cup X_j$ a été ajouté. Si $X'_i \subset X'_j$, cela signifie que la fermeture de X_i contient celle de X_j et donc tous les nœuds fréquents contenant X_i peuvent également être étendus par X_j. Inversement, si $X'_j \subset X'_i$, la fermeture de X_j contient celle de X_i et les nœuds fréquents contenant X_j peuvent être étendus par X_i. Si $X'_i \neq X'_j$, alors aucune modification n'est à apporter sur l'ensemble de

motifs déjà extraits. Le motif B peut simplement être ajouté aux motifs fermés fréquents.

Algorithme 2 : CHARM-Property

si $X_i' = X_j'$ **alors**
  $Noeuds \leftarrow Noeuds \backslash X_j$;
  Remplacer tous les X_i par B;
fin
sinon si $X_i' \subset X_j'$ **alors**
  Remplacer tous les X_i par B;
fin
sinon si $X_j' \subset X_i'$ **alors**
  Remplacer tous les X_j par B;
  NouveauNoeud $\leftarrow (A, B)$;
fin
sinon
  NouveauNoeud $\leftarrow (A, B)$;
fin

4.3.3 Des motifs fréquents aux règles d'association

L'extraction des règles d'association peut se faire soit en même temps que l'extraction des motifs, comme l'algorithme A PRIORI par exemple, soit a posteriori. Nous présentons ici l'algorithme utilisé par Szathmary (2006) pour extraire la base $\mathcal{MNR}$ (algorithme 3). L'algorithme prend en entrée à la fois les générateurs et les fermés des classes d'équivalence, ainsi que leur support. Pour extraire $\mathcal{RMNR}$ il suffit de modifier la ligne 4 de l'algorithme pour ne prendre que les fermés qui ne sont pas contenus dans un autre fermé.

4.4 Règles d'association pour les LOD

Völker et Niepert (2011) sont parmi les premiers à appliquer les règles d'association aux LOD. Les auteurs utilisent les règles d'association pour générer automatiquement la TBox d'une base de connaissances à partir de la ABox. Pour cela, ils génèrent différentes tables de transaction à partir des données RDF, puis extraient les règles pour chacune de ces tables. L'extraction se base sur l'algorithme A PRIORI. Les règles obtenues peuvent être des subsomptions de classes ($C \sqsubseteq D$, $C \sqcap D \sqsubseteq E$) ou de prédicats ($r \sqsubseteq s$), des informations sur le domaine ($\exists r.\top \sqsubseteq C$) et le co-domaine ($\exists r^{-}.\top \sqsubseteq C$) d'un prédicat, la transitivité d'une relation ($r \circ r \sqsubseteq r$). Le processus est coûteux, puisque six tables de transactions différentes doivent être créées pour pouvoir découvrir toutes ces relations. Par exemple, pour découvrir les subsomptions de classes, la table $I \times C$ est créée, où I est l'ensemble des instances et C l'ensemble des classes. Pour découvrir les subsomptions de prédicats, en revanche, les transactions sont des paires d'instances ($I \times I$) et les attributs sont les prédicats.

Par la suite, les travaux proposés s'intéressent plus à la question de la complétude. S'agissant d'un enjeu important des LOD, plusieurs travaux proposent des méthodes pour découvrir des règles permettant d'évaluer la complétude des données. Lajus et F. M. Suchanek (2018) utilisent les règles d'association pour découvrir des propriétés que doivent nécessairement avoir les instances de certaines classes. Ces propriétés sont nommées *obligatory attributes*. Par exemple, un téléphone a nécessairement une marque [13].

13. Techniquement, certains téléphones peuvent ne pas avoir de marque, et c'est là un problème récurrent dans la modélisation

Algorithme 3 : Extraction des règles d'association de $\mathcal{MNR}$

Données : $\mathcal{C}$: l'ensemble des fermés et $\mathcal{G}$ leurs générateurs et les supports associés ; θ : seuil de confiance
Résultat : $\mathcal{MNR}$: ensemble des règles d'association extraites

Extraction
  $\mathcal{MNR} \leftarrow \varnothing$;
  pour chaque $g \in \mathcal{G}$ **faire**
    $C_g \rightarrow \{c \in \mathcal{C} \mid g \subset c\}$;
    pour chaque $c \in C_g$ **faire**
      lhs $\leftarrow g$;
      rhs $\leftarrow c \backslash g$;
      $r \leftarrow (lhs \rightarrow rhs)$;
      r.support $\leftarrow g$.support ;
      r.confiance $\leftarrow c$.support / g.support ;
      si $r.confiance \geqslant \theta$ **alors**
        $\mathcal{MNR} \leftarrow \mathcal{MNR} \cup \{r\}$;
      fin
    fin
  fin

Cela se traduit par des règles de la forme

$$C \sqsubseteq \exists r.\top.$$

Au lieu de rechercher des prédicats nécessaires à certaines classes, Aly et al. (2018) recherchent des *cardinalités maximales* de prédicats, c'est-à-dire le nombre maximal de relations que peut avoir une ressource avec ce prédicat. Cela permet d'exprimer, par exemple, que *une ressource ne peut avoir qu'une et une seule marque*. En considérant la cardinalité maximale *contextuelle* les auteurs permettent de découvrir des cardinalités maximales qui ne s'appliquent qu'à certaines classes. Par exemple, *un téléphone ne peut avoir qu'une et une seule marque*. Cela s'avère particulièrement intéressant quand un même prédicat est utilisé dans des contextes variés. Si l'on considère le prédicat `successor` par exemple, il est possible qu'un téléphone ait un nombre illimité de successeurs directs (déclinaison d'une gamme de produits par exemple), tandis qu'un président n'aura qu'un seul successeur direct. Les règles découvertes sont de la forme

$$C \sqsubseteq \leqslant_x r.$$

L'algorithme AMIE+, proposé par Galárraga et al. (2013, 2015), est une référence en matière de fouille de règles sur le web des données. Cet algorithme s'appuie sur le graphe RDF et recherche des implications entre les prédicats, indépendamment de leurs domaine et co-domaine. Par exemple, *le téléphone qui succède à un téléphone de marque M est également de marque M* est une règle qui pourrait être découverte par AMIE+. De manière générique, ces règles sont de la forme

$$B_1 \circ B_2 \circ \cdots \circ B_{n-1} \sqsubseteq B_n$$

où B_i est appelé un atome et correspond à une relation ou à l'inverse d'un relation entre deux variables (c.-à-d. $r(x_i, x_j)$ ou $r^-(x_j, x_i)$) et $\circ$ est la relation de composition. 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 arêtes).

de connaissances. Voir la discussion de Lajus et F. M. Suchanek (2018) sur le sujet et le choix d'accepter ou non une règle qui contient quelques exceptions.

Synthèse du chapitre

Dans ce chapitre, nous avons introduit la notion de motif ainsi que les règles d'association. Ces dernières sont pertinentes pour la recherche de définitions, puisqu'une définition $X \leftrightarrow Y$ peut être envisagée comme une paire de conditions nécessaire et suffisante : $X \rightarrow Y$ et $Y \rightarrow X$. Enfin, nous avons présentés des algorithmes de fouille de règles d'association, et notamment ECLATZ, qui s'appuie sur ECLAT.

Chapitre 5

Fouille de redescriptions

Sommaire

Dans ce chapitre, nous nous intéressons aux redescriptions. Une redescription est un couple d'ensemble d'attributs qui caractérisent le même ensemble d'objets. La fouille de redescriptions a été introduite par Ramakrishnan, D. Kumar et al. (2004) afin de trouver des « sous-ensembles de données qui peuvent offrir des descriptions multiples ». Les auteurs motivent cette nouvelle tâche avec trois scénarios d'utilisation. Premièrement, ils constatent que dans les domaines scientifiques qui s'appuient sur de grosses masses de données (*data-driven sciences*), les données ont de nombreuses caractérisations. La plupart des tâches de fouille reposant sur la recherche de motifs, le fait que des objets similaires puissent être caractérisés de deux manière différentes nuit à leur découverte.

Dans ce cas, la fouille de redescription permet d'identifier deux caractéristiques, ou deux termes, désignant un même ensemble. Deuxièmement, dans les processus de découverte de connaissances, il peut y avoir un glissement de vocabulaire menant à des termes multiples pour désigner une seule chose. Cela introduit inutilement de la redondance dans les données, qui peut être détectée par les redescriptions et corrigée en conséquence. Troisièmement, la fouille de redescriptions peut aussi être utilisée pour trouver une séquence d'approximations permettant de faire du *story-telling* [14] (D. Kumar et al., 2008). Cette

14. Le *story-telling* consiste à passer d'un état à un autre par petites transformations successives. Le jeu inventé par Lewis Caroll consistant à passer d'un mot à l'autre en ne changeant qu'une lettre à la fois est une illustration de *story-telling*. Par exemple, pour passer de FOUR à FIVE : FOUR → FOUL → FOOL → FOOT → FORT → FORE → FIRE → FIVE.

Vues	V_1						V_2			
Attr.	a	b	c	d	e	f	m_1	m_2	m_3	m_4
fig. 1		×	×		×			3	3	Triangle
fig. 2	×				×		×	0	3	Triangle
fig. 3		×			×			2	3	Triangle
fig. 4	×	×			×		×	2	3	Triangle
fig. 5				×	×			0	3	Triangle

m_1 : Possède au moins un angle droit
Valeurs booléennes
m_2 : Nombre maximal de côtés égaux
m_3 : Nombre total de côtés
Valeurs numériques
m_4 : Type de figure
valeurs nominales

FIGURE 5.1 – Exemple de données pour les redescriptions. Les attributs de la première vue correspondent à l'exemple de la figure 3.1 page 24.

approche est notamment utilisée en biologie, pour établir des relations entre diverses observations (Hossain et al., 2012). Ramakrishnan, D. Kumar et al. (2004) proposent le premier algorithme de fouille de redescriptions, nommé CARTWHEELS. Cet algorithme, qui utilise des arbres de décision, est détaillé dans la section 5.2. Les redescriptions sont alors pensées comme des ensembles définis par les propriétés booléennes des objets contenus. Cela revient à rechercher une « quasi-égalité » d'ensembles définis avec les opérateurs ensemblistes usuels. L'intérêt des redescriptions réside notamment dans la définition de cette « quasi-égalité ». Par la suite, Galbrun et Miettinen (2012b) ont généralisé les redescriptions pour permettre de tenir compte de variables numériques et nominales. Une interface graphique, nommée SIREN (Galbrun et Miettinen, 2012a), est proposée pour visualiser et manipuler les redescriptions résultant de la fouille.

5.1 Définition

Dans cette section, nous introduisons de manière générale les redescriptions en nous appuyant sur les définitions de Galbrun et Miettinen (2017). Dans les sous-sections qui suivent, nous reviendrons sur certains points clés abordés ici. La figure 5.1 servira d'exemple. La fouille de redescriptions, comme la fouille de règles d'association, s'appuie sur un ensemble d'objets et d'attributs, parfois appelés *entités* et *descripteurs*. Par souci de consistance entre les différentes approches, nous dénoterons l'ensemble des objets G et l'ensemble des attributs M. L'ensemble des attributs est partitionné en *vues* V_i telles que $M = V_1 \cup \cdots \cup V_n$ et $V_i \neq V_j \Leftrightarrow i \neq j$. La construction de ces vues est discutée dans la section 5.1.1. À chaque paire (objet, attribut) est associée une valeur qui peut être booléenne, numérique ou nominale, selon le domaine de l'attribut. La figure 5.1 présente un exemple comprenant 5 objets et 10 attributs répartis en deux vues. La première vue (attributs a à f) correspond au contexte de la figure 3.1 page 24, qui servait d'introduction à la FCA. La deuxième vue (attributs m_1 à m_4) correspond à de nouveaux attributs.

À partir d'un ensemble d'objets G et d'une partition d'attributs $M = V_1 \cup \cdots \cup V_n$, la fouille de redescription consiste à rechercher des paires de requêtes (q_i,q_j) où q_i et q_j sont des assertions logiques sur les valeurs que peuvent prendre des ensembles d'attributs X_i et X_j. Par exemple, $b \wedge c \leftrightarrow (m_4 = Triangle)$ est une redescription. Les deux ensembles d'attributs doivent provenir de vues disjointes : il ne doit pas y avoir d'attribut de X_i appartenant à la même vue qu'un attribut de X_j et *vice versa*. Par exemple, $(m_3 = 3) \leftrightarrow (m_4 = Triangle)$ n'est pas une redescription parce que m_3 et m_4 sont des attributs de la même vue. Les requêtes sont exprimées en logique propositionnelle et peuvent comprendre des conjonctions, des disjonctions et des négations. Le support d'une requête, noté $\text{support}(q)$ désigne l'ensemble des objets qui satisfont cette requête.

Une requête définit une partition des objets en deux ensembles : ceux qui la satisfont et ceux qui ne la satisfont pas. Un paire de requêtes $q_1 \leftrightarrow q_2$ en définit donc quatre : ceux qui satisfont à la fois q_1 et q_2, uniquement q_1, uniquement q_2, aucune des deux, dénotés respectivement E_{11}, E_{10}, E_{01} et E_{00}. Les

indices indiquent si la requête correspondante est satisfaite. Par exemple, $E_{10}(q_1 \leftrightarrow q_2)$ contient tous les objets qui satisfont q_1 mais pas q_2. Lorsque cela ne génère pas d'ambiguïté, les ensembles $E_{ij}(R)$ seront simplement notés E_{ij}.

Les redescriptions sont extraites en fonction de trois mesures : le support, la similarité et la p-valeur. Le support d'une redescription R est l'ensemble des objets du jeu de données qui satisfont à la fois q_1 et q_2 : $\text{support}(R) = E_{11}$. Idéalement, le support n'est pas trop petit (auquel cas, la redescription est trop spécifique pour être intéressante), ni trop grand (la redescription est alors trop générique). La similarité est discutée dans la section 5.1.2. Elle mesure à quel point les deux requêtes d'une redescription désignent des ensembles similaires (c.-à-d. ayant les mêmes objets). Usuellement, le coefficient de Jaccard est utilisé.

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

Une redescription n'est conservée que si son coefficient de Jaccard est supérieur à un seuil fixé. Par exemple, considérons la redescription $R : b \leftrightarrow (m_2 = 2)$ du jeu de données de la figure 5.1. Ses requêtes sont $q_1 = b$ et $q_2 = (m_2 = 2)$. On a

$$|E_{11}| = |\{f_3, f_4\}| = 2, \qquad |E_{10}| = |\{f_1\}| = 1, \qquad |E_{01}| = |\varnothing| = 0 \qquad \text{et} \qquad |E_{00}| = |\{f_2, f_5\}| = 0.$$

On a donc $|\text{support}(R)| = 2$ et $\text{jacc}(R) = \frac{2}{2+1} = \frac{2}{3}$. Si on requiert un coefficient de Jaccard supérieur à 0.8, alors la redescription n'est pas conservée. En revanche, la redescription $b \wedge \neg c \leftrightarrow (m_2 = 2)$ a un coefficient de Jaccard de 1 : elle est donc conservée.

Des p-valeurs (discutées en section 5.1.3) sont calculées pour évaluer la significativité statistique du support observé de la redescription. Intuitivement, la p-valeur répond à la question « Connaissant le support des deux requêtes, à quel point pouvait-on s'attendre à ce que la redescription ait le support observé ? »

Pour résumer, le but de la fouille de redescriptions est d'extraire des paires de requêtes avec des supports similaires (c.-à-d. un coefficient de Jaccard proche de 1) dont la taille est comprise dans un intervalle déterminé et qui sont statistiquement significatives (c.-à-d. p-valeur proche de 0).

5.1.1 Vues

La partition des objets en vues est un élément caractéristique de la fouille de redescriptions : cela permet de contraindre les attributs qui apparaissent d'un côté ou de l'autre d'une redescription. Cependant, le choix du découpage des attributs reste à la charge de l'analyste. Dans la plupart des cas, la partition est triviale. Parfois, les données à disposition sont clairement distinctes. Par exemple, deux types de capteurs différents fournissent des données sur un même objet. D'autres fois, le problème induit les associations recherchées, et donc la partition. Par exemple, Galbrun et Miettinen (2012b) recherchent des niches bioclimatiques et ont à disposition à la fois des données climatiques et des données sur la faune locale.

Lors d'une démarche exploratoire en revanche, il se peut qu'un analyste désire utiliser les redescriptions sans *a priori* sur ses données. Dans ce cas, il n'y a pas de partition définie. Il est alors possible de considérer chaque attribut comme une vue. Cela revient sensiblement à considérer deux vues avec les mêmes données en vis-à-vis, cependant, le même attribut ne peut pas se retrouver des deux côtés d'une redescription.

Bien qu'il soit en théorie possible d'avoir autant de vues que souhaité, en pratique, la quasi-totalité des algorithmes existants ne prennent en compte que deux vues (ou un seul attribut par vue). Cela s'explique entre autres par la difficulté de définir une mesure de similarité adéquate lorsqu'il y a plus de deux vues. De plus, cela complexifie à la fois l'espace de recherche et l'exploration par l'algorithme de fouille. Enfin, cela produit des règles plus complexes et difficiles à interpréter pour l'analyste. On peut tout de même noter le travail de Mihelčić, Džeroski et Šmuc (2018), qui proposent un algorithme permettant de traiter des données avec plus de deux vues.

5.1.2 Similarité des supports

La similarité mesure à quel point deux ensembles d'attributs sont semblables. Elle est réflexive et symétrique. C'est là une différence majeure avec la confiance (formule 4.1 page 37), utilisée pour la fouille de règles d'association. En effet, ici, les deux requêtes sont traitées de manière identique. Contrairement à l'équivalence, la similarité n'est pas transitive. L'absence de cette propriété est essentielle pour les tâches de type *story telling*.

Plusieurs mesures permettent de calculer une similarité, comme le coefficient de Sørensen-Dice ou des mesures de distance [15]. Cependant, la mesure la plus utilisée dans le domaine est le coefficient de Jaccard [16]. 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. Une redescription dont le coefficient de Jaccard vaut 1 est une définition, c'est-à-dire une double implication.

5.1.3 Significativité des supports

De manière informelle, la p-valeur indique, pour un modèle donné, la probabilité que les données soient au moins aussi extrêmes que celles observées. Elle est utilisée pour effectuer un test d'hypothèse reposant sur deux hypothèses contradictoires. L'hypothèse par défaut ou hypothèse nulle, notée H_0, qui pourrait se résumer en « il n'y a rien de spécial à observer », correspond au cas où les données observées respectent la distribution de probabilités à laquelle l'analyste s'attend. L'hypothèse alternative H_1 correspond au cas où les observations ne suivent pas la distribution prévue. Le test consiste à tester H_0 et déterminer si H_0 permet d'expliquer les données observées. Si ce n'est pas le cas, H_0 est rejetée et l'hypothèse H_1 est retenue. La p-valeur correspond à la probabilité de rejeter H_0 par erreur. Plus elle est faible, plus il y a de chances que H_0 doive être rejetée au profit de H_1.

Par exemple, sur la figure 5.2, la courbe représente une distribution de probabilités de la valeur de x pour une hypothèse nulle H_0. Comme il s'agit d'une distribution de probabilités, l'aire totale sous la courbe vaut 1. La valeur observée de x est notée x_{obs}. La zone grisée correspond donc à la probabilité de tirer une valeur supérieure ou égale à celle observée sous l'hypothèse H_0, c'est-à-dire la p-valeur associée à l'observation $x \geqslant x_{obs}$. Plus x_{obs} est à l'extrémité de cette courbe, moins cette observation est probable selon l'hypothèse nulle. Si la probabilité est sous le seuil de significativité choisi, on considère que x_{obs} est trop improbable dans l'hypothèse H_0 pour être expliquée par celle-ci ; H_0 est alors rejetée.

p-valeur : définition et exemple

Dans le cadre de la fouille de redescriptions, la p-valeur indique la significativité d'une redescription. Pour une redescription $X \leftrightarrow Y$, l'hypothèse nulle est que les supports des requêtes X et Y sont indépendants : $P(X \cup Y) = P(X) \times P(Y)$. On évalue alors la probabilité que leur intersection ait la taille observée sous cette hypothèse d'indépendance. La valeur observée est donc le cardinal du support de $X \cup Y$, soit $|E_{11}|$.

Définition 21 (p-valeur d'une redescription)**.** *Étant donné une redescription $q_1 \leftrightarrow q_2$, la p-valeur est la probabilité, connaissant le cardinal de $E_{1.}$ (c.-à-d. $|E_{1.}|$) et de $E_{.1}$ (c.-à-d. $|E_{.1}|$), que le support de $E_{1.} \cup E_{.1}$ contienne au moins autant d'objets que le support effectivement observé :*

$$\text{pval}(q_1 \leftrightarrow q_2) = P(|E_{1.} \cap E_{.1}| \geqslant |E_{11}| \mid |E_{1.}|, |E_{.1}|, |G|)$$

Ci-dessous, nous verrons que cette définition permet plusieurs méthodes de calcul de la p-valeur.

15. Une mesure de distance satisfait les propriétés de symétrie ($d(A,B) = d(B,A)$), de séparation ($d(A,B) = 0 \Leftrightarrow A = B$) et l'inégalité triangulaire ($d(A,C) \leqslant d(A,B) + d(B,C)$).

16. La fonction 1 – *Jaccard* est une mesure de distance.

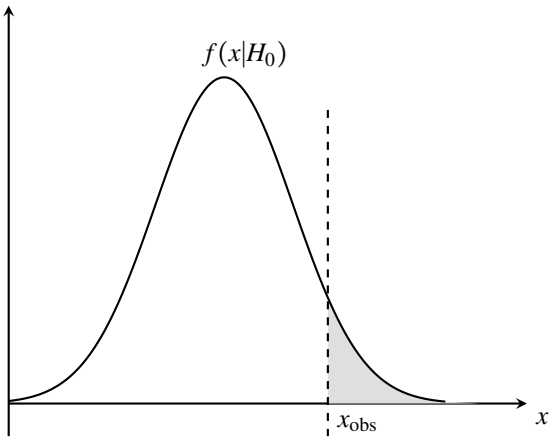
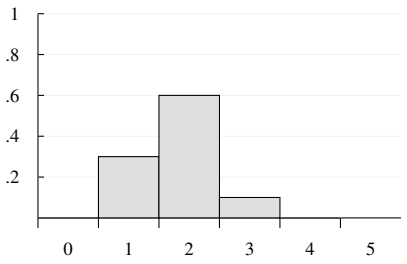


FIGURE 5.2 – Illustration de la p-valeur pour une distribution donnée et une valeur x observée (x_{obs})

.

	A	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
1	×	×	×	×	×	×	×				
2	×	×	×	×				×	×	×	
3	×	×			×	×		×	×		×
4			×		×		×	×		×	×
5				×		×	×		×	×	×
$\lvert E_{11}\rvert$	–	3	2	2	2	2	1	2	2	1	1

(a) Les différents supports possible de B, et le cardinal de E_{11} pour A fixé.



(b) Distribution de probabilités associée à une redescription.

FIGURE 5.3 – Exemple de répartition de probabilités.

Considérons par exemple, dans un contexte de 5 objets, les attributs A et B qui ont tous les deux un support de 3 objets. Sachant que le cardinal du support de A est de 3, il y a $\binom{5}{3} = 10$ supports possibles pour A. Il en va de même pour B. Fixons le support de A : $A' = \{1,2,3\}$ et observons le cardinal de E_{11} en fonction du support de B (figure 5.3a). Parmi les 10 supports possibles de B, $|E_{11}|$ vaut 3 pour un seul d'entre eux, $|E_{11}|$ vaut 2 pour six d'entre eux et $|E_{11}|$ vaut 1 pour les trois derniers. Cette répartition reste la même quel que soit le support de A fixé. Comme il y a $\binom{5}{3}$ supports possibles pour A, on a 10 fois $|E_{11}| = 3$, 60 fois $|E_{11}| = 2$ et 30 fois $|E_{11}| = 1$. On obtient donc la distribution de probabilités représentée dans la figure 5.3b. Maintenant, intéressons-nous à la redescription $A \leftrightarrow B$ observée. Si celle-ci a un support de 2 objets, alors $\text{pval}(A \leftrightarrow B) = 0.7$. En revanche, si elle a un support de 3 objets, $\text{pval}(A \leftrightarrow B) = 0.1$ Ainsi, dans l'exemple de la figure 5.1, p-valeur de la redescription $b \leftrightarrow m_2 \geqslant 1$ est de 0.2.

Méthodes de calcul de la p-valeur d'une redescription

Deux distributions de probabilité peuvent être utilisées pour le calcul de la p-valeur : la loi binomiale et la loi hypergéométrique.

La loi hypergéométrique. Cette loi modélise le problème suivant : nous disposons de N boules dans une urne. Parmi ces boules, K sont noires, les autres sont blanches. On tire n boules (sans remise) dans

l'urne et on observe k, le nombre de boules noires obtenues. La loi hypergéométrique $h(N,K,n,k)$ représente la distribution de probabilité de k, et se calcule comme suit :

$$P(X=k) = \frac{\binom{N}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

La distribution de probabilités d'une redescription peut s'énoncer de manière similaire : connaissant les supports de q_1 et q_2 ainsi que le nombre total d'objets, on peut rapporter le problème à tirer un certain nombre d'objets (désignés par q_1) et regarder parmi ces objets lesquels satisfont q_2. Ainsi, le nombre total de boules correspond au nombre total d'objets ($N = |E|$), le nombre de boules tirées correspond à l'une des requêtes de la redescription ($n = |\text{support}(q_1)|$) et les boules noires correspondent à la seconde requête ($K = |\text{support}(q_2)|$). En conséquence, le nombre de boules noires tirées correspond au support de la redescription ($k = |\text{support}(q_1 \leftrightarrow q_2)|$). On retrouve alors l'équation 5.1 de la table 5.1. En regardant celle-ci de plus près, il peut sembler que la significativité d'une règle ne soit pas symétrique (c.-à-d. significativité$(q_1 \leftrightarrow q_2) \neq$ significativité$(q_2 \leftrightarrow q_1)$. Cependant, il est prouvé que $h(N,K,n,k) = h(N,n,K,k)$ (Egghe et Rousseau, 1997), autrement dit :

$$\sum_{|\text{support}(q_1,q_2)|}^{|E|} \frac{\binom{|\text{support}(q_1)|}{k} \times \binom{|E|-|\text{support}(q_1)|}{|\text{support}(q_2)|-k}}{\binom{|E|}{|\text{support}(q_2)|}} = \sum_{|\text{support}(q_1,q_2)|}^{|E|} \frac{\binom{|\text{support}(q_2)|}{k} \times \binom{|E|-|\text{support}(q_2)|}{|\text{support}(q_1)|-k}}{\binom{|E|}{|\text{support}(q_1)|}}.$$

La loi binomiale. Pour reprendre l'illustration avec les tirages, la loi binomiale modélise le problème suivant : on dispose d'une urne avec N boules dont K boules noires. Les boules noires sont en proportion $p = \frac{K}{N}$. On tire n fois une boule (avec remise) dans l'urne et on observe k, le nombre de boules noires obtenues. La distribution de probabilité de k est alors définie par

$$P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

En transposant le problème aux redescriptions, on obtient la formule 5.2 de la table 5.1, où la probabilité p est notée α. La valeur de α, qui correspond à la probabilité qu'un objet valide à la fois q_1 et q_2, peut être calculée de deux façons. Soit on considère simplement le support des requêtes, soit on considère chacun des attributs qui les constituent. On obtient ainsi deux méthodes de calcul, représentées par les formules 5.3 et 5.4 de la table 5.1.

5.1.4 Redescriptions *versus* règles d'association

La fouille de règles d'association précède la fouille de redescriptions et a été une source d'inspiration pour leurs auteurs. Les deux approches partagent donc de nombreux points communs. Cependant, les approches restent très différentes. Dans cette section, nous nous appuyons sur leurs définitions respectives pour synthétiser leurs points communs et leur différences.

Tout d'abord, les règles d'association acceptent n'importe quel attribut dans leur antécédent et dans leur conséquence. Au contraire, les redescriptions reposent sur le système de vues qui restreint chaque côté de la redescriptions à un sous-ensemble d'attributs.

Ensuite, les redescriptions reposent sur une mesure symétrique. En ce sens, elles peuvent être pensées comme des « règles d'association bidirectionnelles ».

De plus, les deux approches s'appuient sur des données binaires et traitent de relations entre des objets et des attributs. Cependant, les redescriptions ont été étendues pour tenir compte de données numériques et nominales. Bien que certains travaux aillent dans ce sens avec les règles d'association, peu d'implémentations d'algorithme permettant de traiter des données hétérogènes existent, la plupart des algorithmes requérant au préalable de convertir les données sous forme binaire.

TABLE 5.1 – Les différentes *p-value* proposées par Galbrun et Miettinen (2017)

Loi	Formule	
Hypergéométrique	$\sum_{\lvert\text{support}(q_1,q_2)\rvert}^{\lvert E\rvert} \frac{\binom{\lvert\text{support}(q_1)\rvert}{k} \times \binom{\lvert E\rvert - \lvert\text{support}(q_1)\rvert}{\lvert\text{support}(q_2)\rvert - k}}{\binom{\lvert E\rvert}{\lvert\text{support}(q_2)\rvert}}$	(5.1)
Binomiale	$\sum_{\lvert\text{support}(q_1,q_2)\rvert}^{\lvert E\rvert} \binom{\lvert E\rvert}{k} \alpha^k (1-\alpha)^{\lvert E\rvert - k}$	(5.2)
Alpha	**Formule**	
On considère le support de q_1 et q_2.	$\alpha = \frac{\lvert\text{support}(q_1)\rvert \times \lvert\text{support}(q_2)\rvert}{\lvert E\rvert^2}$	(5.3)
On considère le support de chacun des attributs qui composent q_1 et q_2.	$\alpha_q = \begin{cases} \alpha_{q_1}\alpha_{q_2} & \text{si } q = q_1 \wedge q_2 \\ 1 - \alpha_{q_1} & \text{si } q = \neg q_1 \\ \alpha_{q_1} + \alpha_{q_2} - \alpha_{q_1}\alpha_{q_2} & \text{si } q = q_1 \vee q_2 \end{cases}$	(5.4)

Enfin, les redescriptions permettent d'exprimer des formules logiques sur les attributs, tandis que les règles d'association considèrent des ensembles d'attributs.

5.2 Algorithmes

Dans cette section, nous détaillons trois algorithmes qui représentent les principales familles d'algorithmes selon la classification de Galbrun et Miettinen (2017), reproduite figure 5.4.

Le premier algorithme, proposé par Ramakrishnan, D. Kumar et al. (2004), est CARTWHEELS. Son approche s'appuie sur les arbres de décision. Le second, CHARM-L (Zaki et Ramakrishnan, 2005), utilise l'algorithme CHARM et repose sur les treillis de concepts. Enfin, l'algorithme REREMI (Galbrun et Miettinen, 2012b) propose une approche gloutonne qui construit les redescriptions de manière itérative.

5.2.1 CARTWHEELS

CARTWHEELS (Ramakrishnan, D. Kumar et al., 2004) utilise les arbres de décision pour construire des redescriptions. Intuitivement, il s'agit de ramener le problème de fouille de redescriptions à un problème de classification binaire, plus facile à résoudre. Le but est de construire un arbre de décision — avec, par exemple, l'algorithme CART (Breiman et al., 1984) — pour chaque vue, et des les faire « coïncider » au mieux. Pour cela des arbres de décision vont être construits de manière itérative en alternant entre les vues. Initialement, les attributs de la seconde vue sont utilisés pour établir les classes des objets, tandis que les attributs de la première vue sont utilisés comme variables pour construire l'arbre. S'agissant d'arbres de décision, il ne peut y avoir qu'une classe par objet à classifier. Aussi, lors de la première

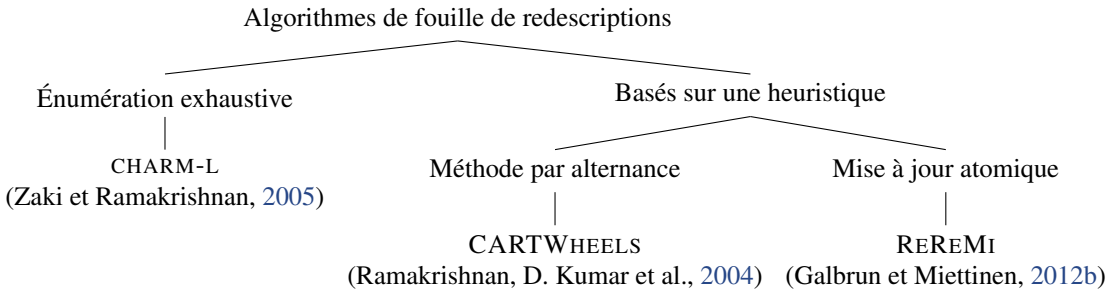


FIGURE 5.4 – Les trois algorithmes de fouille de redescriptions évoqués dans cette thèse, organisés selon la classification de Galbrun et Miettinen (2017).

itération, un attribut du support de chaque objet est sélectionné aléatoirement comme étant sa classe cible. Puis, l'arbre de décision crée une partition des objets, qui définit les classes utilisées à l'étape suivante.

Du fait de la construction des classes des arbres de décision, et notamment parce qu'à la première étape, un seul attribut est utilisé par objet, cet algorithme n'est pas idéal pour les données qui ont plus d'attributs que d'objets : chaque objet risque de générer sa propre classe.

5.2.2 CHARM-L

Zaki et Ramakrishnan (2005) utilisent CHARM-L pour énumérer les redescriptions d'un jeu de données. Les attributs considérés sont nécessairement booléens, et seules des conjonctions sont utilisées. Contrairement aux autres approches, ils n'utilisent pas de vues distinctes entre les attributs. En ce sens, cette approche est extrêmement proche de la fouille de règles d'association, si ce n'est que les règles recherchées sont bidirectionnelles. L'algorithme CHARM-L (Zaki et Hsiao, 2002) génère le treillis de concepts à partir des motifs fermés fréquents extraits par CHARM. En conséquence, les redescriptions trouvées n'utilisent que des conjonctions sur des variables booléennes.

Les auteurs introduisent la notion de redescription conditionnelle, une redescription qui n'est valide que dans un sous-ensemble des données. Par exemple, dans notre exemple des figures géométriques (figure 5.1), *avoir un angle droit* et *être un rectangle* n'est pas équivalent. En revanche, en ne considérant que les figures qui ont quatre côtés, *avoir un angle droit* et *être un rectangle* est équivalent. Une redescription conditionnelle se note $X \leftrightarrow Y \mid Q$ où X, Y et Q sont des ensembles d'attributs, et son coefficient de Jaccard est $\frac{(X' \cap Y') \cap Q'}{(X' \cup Y') \cap Q'}$.

Les auteurs montrent qu'énumérer les redescriptions (conditionnelles ou non) peut se faire en utilisant les générateurs des classes d'équivalence. Deux générateurs $X, Y \subseteq M$ d'une même classe d'équivalence génèrent la règle $X \leftrightarrow Y \mid Q$ où $Q = X \cap Y$. Un exemple est fourni en figure 5.5. L'algorithme 4 détaille la génération des redescriptions à partir des classes d'équivalence.

Cependant, cet algorithme ne permet d'extraire que des redescriptions exactes, c'est-à-dire dont le coefficient de Jaccard vaut 1.

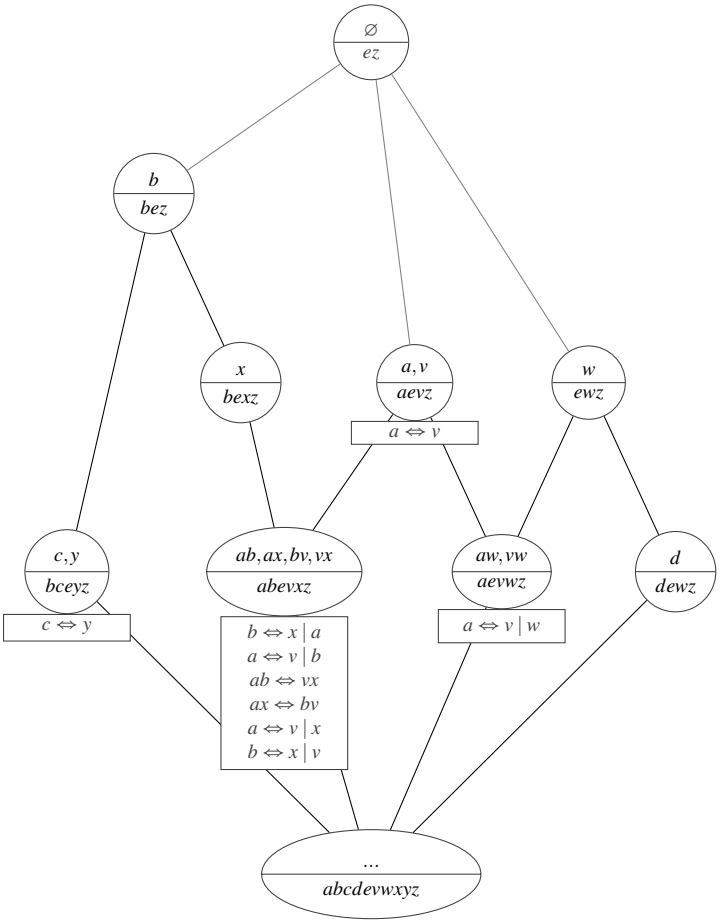
5.2.3 REREMI

Contrairement aux algorithmes présentés précédemment, REREMI ne génère pas directement un ensemble de redescriptions, mais les génère au fur et à mesure, en partant d'un ensemble de paires d'attributs.

Vues	V_1					V_2				
Attr.	a	b	c	d	e	v	w	x	y	z
fig. 1		×	×		×				×	×
fig. 2	×				×	×	×			×
fig. 3		×			×			×		×
fig. 4	×	×			×	×		×		×
fig. 5				×	×		×			×

v : Possède au moins un angle droit
w : Ne possède aucuns côtés égaux
x : Possède exactement 2 côtés égaux
y : Possède exactement 3 côtés égaux
z : Est un triangle

(a) Contexte binaire adapté de l'exemple présenté en figure 5.1. CHARM-L ne considérant pas les vues, celles-ci sont données à titre indicatif.



(b) Treillis de concepts avec les générateurs et les fermés. Les redescriptions extraites avec l'algorithme de Zaki et Ramakrishnan (2005) sont indiquées au niveau de la classe d'équivalence associée.

FIGURE 5.5 – Énumération de redescriptions à partir des générateurs des classes d'équivalence.

Algorithme 4 : CHARM-L pour l'énumération de redescriptions.

Données : $\mathcal{C}$: l'ensemble des fermés, $\mathcal{G}$: l'ensemble des générateurs.
Résultat : $\mathcal{R}$: L'ensemble des redescriptions et redescriptions conditionnelles.

```
CHARM-L
    R ← ∅;
    pour chaque C ∈ C faire
        pour chaque X, Y ∈ G tels que X' = Y' = C faire
            Q ← X ∩ Y;
            R ← X\Q ↔ Y\Q | Q;
            R ← R ∪ R;
        fin
    fin
    retourner R
```

Dans un premier temps, l'ensemble des « redescriptions candidates » est généré. Il s'agit des paires $q_1 \leftrightarrow q_2$, où q_1 contient seulement un attribut $a_1 \in V_1$ et q_2 contient seulement un attribut $a_2 \in V_2$. Les n paires avec le meilleur coefficient de Jaccard sont conservées, n pouvant être configuré par l'analyste. Les paires sont ensuite traitées tour à tour, chacune étant progressivement étendue en ajoutant un attribut à la fois à l'une des requêtes candidates. Une requête candidate q peut être étendue avec un attribut a de quatre façons : $q_1 \wedge a$, $q_1 \vee a$, $q_1 \wedge \neg a$ or $q_1 \vee \neg a$. L'analyste peut restreindre les extensions possibles, par exemple en ne considérant que les conjonctions, ou en excluant les négations. Si l'attribut a est une variable nominale, la valeur de la variable qui augmente le plus le Jaccard est sélectionnée. Si l'attribut est une variable numérique, alors la variable est discrétisée à la volée et l'intervalle qui maximise le coefficient de Jaccard est utilisé. Cela implique théoriquement de tester toutes les paires de valeurs pour les bornes de a, mais il est possible de limiter la complexité de la recherche (voir Galbrun et Miettinen (2012b) page 42). Lorsque la redescription candidate ne peut plus être étendue, que ce soit parce qu'elle a atteint le nombre maximal d'attributs autorisés ou les seuils de supports fixés par exemple, la redescription avec le meilleur coefficient de Jaccard est ajoutée à l'ensemble des résultats.

L'algorithme continue jusqu'à ce qu'il atteigne une profondeur maximale fixée par l'utilisateur ou jusqu'à épuisement des candidats (c'est-à-dire qu'aucune extension ne permet d'augmenter le coefficient de Jaccard). REREMI s'appuyant sur une approche gloutonne, il n'y a pas de garantie d'obtenir les meilleures descriptions : l'algorithme peut s'arrêter à un maximum local : il est possible qu'une redescription ne soit pas étendue alors qu'elle permet d'obtenir une redescription avec des mesures intéressantes pour l'analyste. Les redescriptions ainsi obtenues qui satisfont les critères de sélection sont retournées à l'analyste.

5.3 Traduction des données à l'aide de règles

Un problème récurrent dans la fouille de motifs (fouille d'*itemsets*, de règles d'association, de redescriptions) est le grand nombre de résultats retournés et la redondance de ces résultats. (Leeuwen et Galbrun, 2015) ont proposé une approche pour obtenir un ensemble de résultats plus compact tout en étant représentatif. *Compact*, parce qu'il s'agit de minimiser le nombre total de règles retournées, *représentatif* parce que les règles doivent idéalement couvrir l'intégralité des données. Les motifs retournés sont des règles d'association, unidirectionnelles ou bidirectionnelles (c.-à-d. redescriptions). Ces motifs sont appelés règles de *traduction* du fait de l'analogie avec une tâche de traduction, qui est l'intuition derrière cette approche. Il s'agit en effet de déterminer comment la présence de certains attributs dans un contexte *se traduit* dans un second contexte. Autrement dit, la présence de certains attributs dans une

vue prédit la présence d'autres attributs dans une deuxième vue. Les règles « capturent » ces informations et peuvent alors être utilisées pour reconstruire un contexte à partir de l'autre. Le but de cette approche est donc la sélection d'un ensemble de règles qui soit le plus petit possible tout en permettant une reconstruction des vues la plus fidèle possible. L'algorithme proposé pour répondre à ce problème, nommé TRANSLATOR, s'appuie sur des principes issus de la théorie de l'information, et notamment sur la notion de *description de longueur minimale* (MDL).

5.3.1 Principe de traduction des données

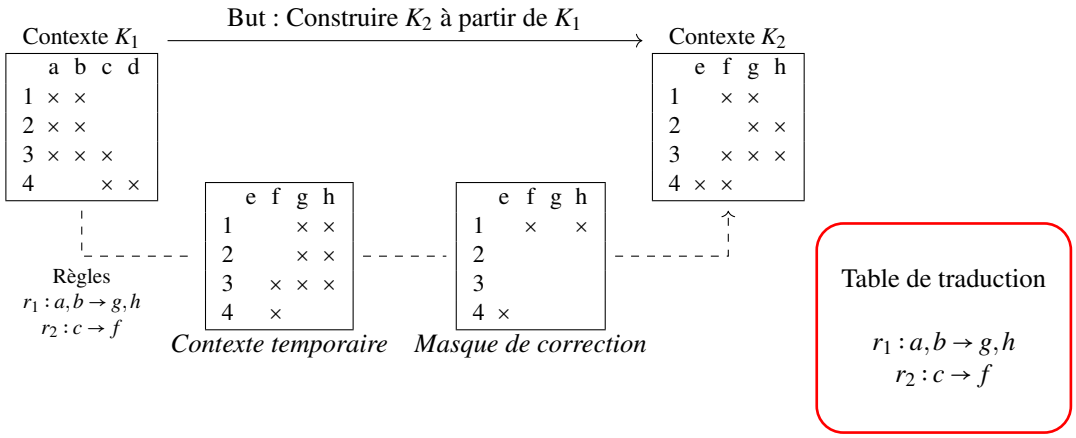


FIGURE 5.6 – Intuition du fonctionnement de TRANSLATOR. Ici, seule la traduction de K_1 vers K_2 est représentée.

La figure 5.6 représente une vue d'ensemble du fonctionnement. Pour simplifier, la figure ne représente que la reconstruction de K2 étant donné K1 ; la direction inverse suit le même principe. Les paragraphes suivants traitent des deux directions.

Le problème peut s'énoncer ainsi : étant donnés deux contextes qui ont les mêmes objets, $K_1 = (G, M_1, I_1)$ et $K_2 = (G, M_2, I_2)$, l'objectif est de découvrir un ensemble de règles qui permettent de reconstruire le contexte K_2 à partir de K_1 et *vice versa*. Une règle de traduction indique que si les attributs de l'antécédent sont présents pour un objet dans le contexte correspondant, alors les attributs de la conséquence sont « probablement » présents dans l'autre contexte. Une règle $X \to Y$ s'interprète donc comme « la présence de l'ensemble d'attributs X dans le contexte K_1 se traduit par la présence de l'ensemble d'attributs Y dans le contexte K_2 ». Les règles peuvent être unidirectionnelles (de K_1 vers K_2 ou de K_2 vers K_1) ou bidirectionnelles (de K_1 vers K_2 et de K_2 vers K_1).

Le but de l'algorithme est alors d'extraire un ensemble de règles $\mathcal{R}$ qui remplisse au mieux les objectifs suivants :

1. Les règles doivent permettre de reconstruire un contexte étant donné l'autre :
   - — Étant donnés K_1 et $\mathcal{R}$, on doit pouvoir déduire K_2 ;
   - — Étant donnés K_2 et $\mathcal{R}$, on doit pouvoir déduire K_1 ;
2. $|\mathcal{R}|$ doit être le plus petit possible.

Deux remarques importantes sont à faire sur ces objectifs. Premièrement, il peut ne pas exister d'ensemble de règles qui permette une reconstruction parfaite. Dans le contexte K_1, par exemple, les objets 1 et 2 ont les même attributs et ne sont donc pas discernables. Cela veut dire que toute règle s'appliquant à l'objet

1 s'appliquera également à l'objet 2. En revanche, dans le contexte K_2, l'objet 1 a l'attribut f que n'a pas l'objet 2, et l'objet 2 a l'attribut h que n'a pas l'objet 1. Il n'y a donc aucune règle qui puisse générer correctement les attributs des objets 1 et 2 dans le contexte K_2 à partir du contexte K_1, d'où la nécessité d'appliquer des masques de correction. Le but est alors de limiter le nombre de corrections nécessaires. Deuxièmement, ces deux objectifs sont divergents : le premier objectif pousse à avoir plus de règles pour représenter au mieux les données, tandis que le second objectif pousse à avoir le moins de règles possibles, indépendamment de la qualité du résultat. Toute la difficulté réside alors dans l'équilibre à trouver entre ces deux objectifs. Une approche utilisant MDL permet de trouver cet équilibre.

5.3.2 Description de Longueur Minimale

La description de longueur minimale, ou MDL (Grünwald, 2004 ; Rissanen, 1978), est une approche qui s'intéresse à la question de la sélection de modèle : étant donné un ensemble d'explications pour une observation donnée, quelle explication doit être choisie ? D'après Rissanen (1978), MDL offre un critère de sélection de modèles, indépendamment de leur complexité, sans l'hypothèse restrictive que les données sont un échantillon d'une véritable distribution [17].

MDL repose sur deux notions. Il y a d'abord la notion d'apprentissage de modèle, et l'idée selon laquelle un apprentissage se base sur la régularité des données ; l'écart entre le modèle et les données relève alors du bruit. Il y a ensuite la notion de compression de données, et l'idée que plus il y a de régularités dans les données, plus la compression de données pourra être importante. De ces deux idées, on peut déduire que sélectionner le meilleur modèle revient à trouver celui qui offre le meilleur compromis entre sa complexité et la compression des données qu'il permet d'opérer. C'est ce qui est fait avec MDL. Théoriquement, la complexité du modèle correspond à la complexité de Kolmogorov (la complexité d'un objet correspond à la longueur du plus court programme capable de produire cet objet). Cependant, la complexité de Kolmogorov n'étant pas calculable, plusieurs approches pratiques permettent de calculer la complexité du modèle dans le cadre de MDL.

Au final, pour un ensemble de modèles $\mathcal{M}$, MDL va sélectionner le modèle qui minimise $L(M_i) + L(D \mid M_i)$ où $L(M_i)$ est la longueur du modèle M_i, et $L(D \mid M_i)$ est la longueur des données encodées avec le modèle M_i.

5.3.3 Algorithme TRANSLATOR

L'algorithme TRANSLATOR (Leeuwen et Galbrun, 2015) a été proposé pour répondre aux deux objectifs de la fouille de règles de traduction en utilisant MDL. Les *données observées* correspondent aux deux contextes K_1 et K_2. Le *modèle* correspond à l'ensemble des règles de traduction, ainsi qu'au *masque* qui permet de rectifier les erreurs engendrées par les règles, c'est-à-dire retirer les items introduits par erreur, et ajouter ceux manquants. L'ensemble des règles permet de construire un contexte $K_{Temp} = (G, M_2, I_{Temp})$ avec

$$I_{Temp} = \{(g,m) \in G \times M_2 \mid \exists X \to Y, X \subseteq g', m \in Y\}$$

Le masque peut être pensé comme un contexte K_{Masque} où la relation d'incidence $I_{Masque} \subseteq G \times M_2$ correspond aux corrections à effectuer sur le contexte temporaire K_{Temp}. Il s'applique comme un OU exclusif sur le contexte temporaire construit à partir des règles pour permettre de générer K_2 : $K_2 = K_{Temp} \oplus K_{Masque}$. Dans l'exemple 5.6 notamment, du fait de la règle $r_1 : ab \to gh$, le masque devra garder en mémoire que *l'objet 1 ne devrait pas avoir l'attribut h*. Dans l'exemple, on a donc $(1,h) \in I_{Temp}$ et $(1,h) \in I_{Mask}$. Le OU exclusif entre K_{Temp} et K_{Mask} permet alors de retirer l'attribut h pour l'objet 1, qui a été introduit par l'application de la règle r_1.

17. *« Minimum Description Length provides a criterion for the selection of models, regardless of their complexity, without the restrictive assumption that the data form a sample from a 'true' distribution. », Rissanen, 2008*

Calcul des longueurs de code

Étant donné un contexte K et un ensemble d'attributs $X \subseteq M$, la *longueur* de X est

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

Cette longueur correspond au nombre minimal de bits requis pour encoder X. La longueur d'une règle $L(X \to Y)$ est définie comme la longueur de l'ensemble d'attributs $X \cup Y$ à laquelle est ajoutée une longueur de 1 bit indiquant si la règle est unidirectionnelle ou bidirectionnelle. Si la règle est unidirectionnelle, 1 bit supplémentaire est requis pour indiquer son sens (de K_1 vers K_2 ou de K_2 vers K_1). Ainsi, une règle unidirectionnelle coûte 1 bit de plus qu'une règle bidirectionnelle, favorisant ainsi l'utilisation de règles bidirectionnelles. La longueur du masque est mesurée en sommant, pour chaque objet, la longueur de l'ensemble d'attributs qu'il supporte : $L(Masque) = \sum_{g \in G} L(g')$.

La « qualité » de la règle se mesure alors en faisant la différence entre sa longueur et la longueur de l'information qu'elle apporte :

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

où $Masque^+$ correspond aux éléments ajoutés au masque (erreurs introduites par la règle) et $Masque^-$ correspond aux éléments retirés du masque (erreurs corrigées par la règle). Il est important de noter que cette mesure dépend du masque, et donc que la qualité d'une règle dépend des règles déjà trouvées.

Déroulé de l'algorithme

L'ensemble de règles est construit de manière itérative. Au départ, il n'y a pas de règle et le masque correspond donc au contexte cible.

La règle $X \to Y$ qui maximise Δ est ajoutée et le masque est actualisé. Une nouvelle règle est ajoutée, et ainsi de suite jusqu'à ce que plus aucune règle ne diminue la longueur de code. La figure 5.7 illustre ce processus.

Le masque étant mis à jour à chaque étape, la qualité Δ 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 sélectionnée apporte une information qui n'est pas contenue dans les autres, ce qui limite la redondance d'information.

5.4 Redescriptions appliquées aux LOD

Les redescriptions ont été majoritairement appliquées dans les domaines des sciences naturelles, notamment avec des travaux autour des niches bioclimatiques (Galbrun et Miettinen, 2012b) et en bioinformatique (Ramakrishnan et Zaki, 2009), et il existe peu d'applications dans le domaine de la représentation des connaissances ou du web des données. On peut tout de même citer Galbrun et Kimmig (2014), qui étendent la fouille de redescriptions aux données relationnelles. Les données sont représentées sous forme de graphe où les nœuds représentent des entités quelconques et les arrêtes des relations : l'approche peut s'appliquer à des graphes RDF. L'idée est de trouver deux motifs de sous-graphes qui impliquent les mêmes ensembles d'entités.

Des approches inductives s'appuyant sur les arbres de décision ont notamment été utilisées (Fanizzi, d'Amato et Esposito, 2010 ; Rizzo, d'Amato et al., 2017b). Dans (Rizzo, Fanizzi et al., 2018), les auteurs s'appuient sur des *evidential terminological decision trees* (EDTD) pour classifier des instances en fonction des assertions dans lesquelles elles sont impliquées. Un EDTD est un arbre de décision dont

les nœuds sont étiquetés avec une proposition logique. Plus celle-ci est proche des feuilles, plus elle est spécifique. Elle est accompagnée d'une valeur entre 0 et 1 pouvant s'interpréter comme la probabilité qu'elle soit vraie. Les auteurs peuvent alors faire correspondre une classe à une proposition logique.

Les mêmes auteurs se sont intéressés à la recherche de classes incompatibles (Rizzo, d'Amato et al., 2017a), qui peut être vue comme la tâche « inverse » à la recherche de redescriptions : il s'agit de trouver deux requêtes qui ne désignent aucun objet en commun. Leur approche consiste à rechercher des ensembles disjoints d'instances *via* la construction d'un arbre de décision pour lequel chaque nœud correspond à la définition d'un concept. Toute paire de feuilles de cet arbre correspond nécessairement à deux concepts disjoints. Völker, Fleischhacker et Stuckenschmidt (2015) proposent un *gold standard* des classes incompatibles pour DBpedia, et comparent une approche supervisée par apprentissage et une approche par induction utilisant les règles d'association.

Synthèse du chapitre

Dans ce chapitre, nous avons présenté les redescriptions, qui cherchent des associations « bidirectionnelles » entre les attributs. Nous avons présenté l'algorithme REREMI, qui permet de découvrir des redescription en utilisant une heuristique qui tient compte du support des attributs. Les redescriptions extraites correspondent à des définitions approximatives (c.-à-d. des définitions qui comportent quelques exceptions).

	a	b	c	d
1	×	×		
2	×	×		
3	×	×	×	
4			×	×

Contexte source

	e	f	g	h
1		×	×	
2			×	×
3		×	×	×
4	×	×		

Contexte cible

Ajout des premières règles

Règles	*Contexte créé*	*Masque*	*Explication*
État initial $\varnothing$	e f g h 1 2 3 4	e f g h 1 × × 2 × × 3 × × × 4 × ×	À la première étape, aucune règle n'est encore trouvée. Le contexte recréé est donc vide, et le masque est identique au contexte cible.
Étape 1 $r_1 : a,b \to g,h$	e f g h 1 × × 2 × × 3 × × 4	e f g h 1 × × 2 3 × 4 × ×	Lorsqu'une première règle est ajoutée, les nouvelles croix sont ajoutées au contexte créé. Le masque est alors mis à jour en conséquence.
Étape 2 $r_1 : a,b \to g,h$ $r_2 : c \to f$	e f g h 1 × × 2 × × 3 × × × 4 ×	e f g h 1 × × 2 3 4 ×	Le processus est le même si on ajoute une deuxième règle : les nouvelles croix sont intégrées au contexte créé, puis le masque est mis à jour.

Evaluation de règles candidates à l'étape 3

Règles	*Contexte créé*	*Masque*	*Explication*
$r_1 : a,b \to g,h$ $r_2 : c \to f$ $r_{cand} : c \to h$	e f g h 1 × × 2 × × 3 × × × 4 × ×	e f g h 1 × × 2 3 4 × ×	La règle apporte plus d'erreurs que de données valides (c.-à-d. le masque gagne plus de croix qu'il n'en perd lorsqu'il s'actualise). Il n'y a donc aucun intérêt à l'ajouter à l'ensemble de règles.
$r_1 : a,b \to g,h$ $r_2 : c \to f$ $r_{cand} : d \to f$	e f g h 1 × × 2 × × 3 × × × 4 ×	e f g h 1 × × 2 3 4 ×	La règle permet d'ajouter des attributs déjà présents. Elle n'a donc pas d'intérêt *par rapport aux règles déjà trouvées*. Elle est redondante et n'est pas conservée.
$r_1 : a,b \to g,h$ $r_2 : c \to f$ $r_{cand} : c \to e$	e f g h 1 × × 2 × × 3 × × × × 4 × ×	e f g h 1 × × 2 3 × 4	La règle apporte autant de données valides que d'erreurs. Elle représente donc un coût nul pour le masque, auquel s'ajoute le coût de son encodage. Elle n'est pas conservée.

FIGURE 5.7 – Fonctionnement de la mise-à-jour du contexte créé et du masque pour plusieurs règles.

Chapitre 6

Découverte de définitions dans le web des données

Sommaire

Ce chapitre et le suivant présentent les différentes contributions de cette thèse. Le présent chapitre aborde les aspects théoriques, et se décompose en deux sections.

Dans la première section, nous recherchons à classifier des ressources à partir de triplets RDF. Pour cela, nous introduisons une *pattern structure* qui tient compte à la fois de la taxonomie des classes et de la taxonomie des prédicats.

Dans la seconde section, nous présentons notre utilisation des règles d'association et des redescriptions pour la découverte de définitions dans le web des données, et nous discutons des spécificités des deux approches. Cette discussion est suivie d'expérimentations qui seront présentées dans le chapitre suivant.

6.1 Une nouvelle *pattern structure* pour les LOD

Dans le chapitre 3, nous avons présenté une *pattern structure* introduite par Alam et al. (2015) pour classifier des ressources RDF au sein d'un treillis. À partir d'un exemple concernant des villes dont les données sont représentées en figure 3.7 page 33, nous avons construit le treillis de concepts associé (figure 3.8 page 34). Bien que la *pattern structure* capture des similarités entre Paris et Rome et entre Paris et Nancy, on remarque que certaines informations présentes dans les données de départ ne sont pas retrouvées. Notamment, on ne découvre pas le fait que Paris et Rome sont toutes les deux des villes

européennes et que Paris et Nancy sont toutes les deux des villes françaises. Cet exemple illustre une limite de l'approche de Alam et al. (2015), qui ne tient pas compte de la relation `rdfs:subPropertyOf` entre les prédicats. Afin de pallier ce manque, nous proposons donc une *pattern structure* qui tient compte des deux ordres partiels définis par les relations `rdfs:subClassOf` et `rdfs:subPropertyOf`.

6.1.1 Prérequis

Les relations `subClassOf` et `subPropertyOf` définissent des ordres partiels entre les classes et entre les prédicats respectivement. On note $(C, \sqsubseteq_C)$ l'ensemble des classes ordonnées selon la relation `subClassOf` et $(P, \sqsubseteq_P)$ l'ensemble des prédicats ordonnés selon la relation `subPropertyOf`. Au sein d'une base de connaissances, il est fréquent de faire en sorte qu'une classe n'ait qu'un parent ; ainsi la relation `subClassOf` devrait former un arbre enraciné. Cependant, il arrive qu'une classe ait plus d'un parent, et la relation forme alors un graphe orienté acyclique. Dans cette section, nous supposons que, dans la base de connaissances étudiée, les classes et les prédicats forment des arbres enracinés. Sur cette structure en arbre, il est possible de définir un plus petit ancêtre commun, pour tout couple de classes ou de prédicats.

Définition 22 (Plus petit ancêtre commun)**.** *Étant donné un arbre $(H, \leqslant)$, le plus petit ancêtre commun de deux nœuds x et y de cet arbre est le nœud z tel que $x \leqslant z, y \leqslant z \nexists z' \leqslant z$ tels que $x \leqslant z'$ et $y \leqslant z'$. On note $\mathrm{lcs}(x, y)$ le plus petit ancêtre commun de x et y.*

Cette définition permet d'obtenir la super-classe la plus spécifique de deux classes. Lorsqu'il s'agira de l'arbre des classes $(C, \sqsubseteq_C)$, on notera le plus petit ancêtre commun lcs_C. De façon similaire, on notera lcs_P le plus petit ancêtre commun de deux prédicats. D'après l'exemple de la figure 3.7, on a donc $\mathrm{lcs}_C(Ville, Pays) = Lieu$ et $\mathrm{lcs}_P(villeDe, étatDe) = situéEn$.

Il est fréquent que les objets des triplets ne soient pas des classes mais des instances. Par exemple, dans le triplet $\langle Nancy, villeDe, France \rangle$, *France* est une instance, et non une classe. Il est donc utile de pouvoir intégrer ces instances à la taxonomie de classes, afin de déterminer la classe commune à deux instances. On veut, par exemple, être capable de dire que Paris et Rome sont toutes deux des capitales. Pour cela, on intègre les instances à la taxonomie de classes en les considérant comme des classes ne contenant qu'elles-même.

Ainsi, si o_1, o_2 sont deux instances de C, et que l'on souhaite connaître leur plus petit ancêtre commun, c'est-à-dire $\mathrm{lcs}_C(o_1, o_2)$, on supposera que cela revient à considérer deux classes $C_{o1} = \{o_1\}$ et $C_{o2} = \{o_2\}$ avec $C_{o1} \sqsubseteq_C C$ et $C_{o2} \sqsubseteq_C C$. On obtiendra $\mathrm{lcs}_C(o_1, o_2) = \mathrm{lcs}_C(C_{o1}, C_{o2}) = C$. Afin d'alléger les notations dans ce chapitre, les instances et les classes seront considérées indifféremment. En reprenant l'exemple de la figure 3.7, on a $\mathrm{lcs}_C(Nancy, Paris) = Ville$.

Nous utilisons également la définition d'un ordre partiel sur les produits cartésiens d'ensembles ordonnés tel que défini par Davey et Priestley (2002).

Définition 23 (Ordre partiel sur un produit cartésien d'ensemble ordonnés)**.** *Soient $P_1, \ldots, P_n$ des ensembles ordonnés. Le produit cartésien $P_i \times \cdots \times P_n$ forme un ensemble de n-upplets qui peuvent être partiellement ordonnés en imposant un ordre membre à membre défini par*

$$(x_1, \ldots, x_n) \leqslant (y_1, \ldots, y_n) \Leftrightarrow \forall i, x_i \leqslant y_i \text{ dans } P_i.$$

Nous définissons finalement le minimum sur un ensemble d'éléments partiellement ordonnés comme l'ensemble de ses minorants.

Définition 24 (Minimum sur les ensembles partiellement ordonnés)**.** *Soit $(P, \leqslant)$ un ordre partiel et $X \subseteq P$ un ensemble d'éléments partiellement ordonnés, avec $X = \{x_1, \ldots, x_n\}$. Le minimum de X est l'ensemble des minorants de X appartenant à X.*

$$min(X) = \{x_i \in X \mid \forall x_j \in X, x_i \leqslant x_j\}$$

À partir de là, il nous est possible de définir notre nouvelle *pattern structure*.

6.1.2 Définition de la *pattern structure*

L'ensemble des objets est constitué des sujets des triplets de la base de connaissances. L'ensemble des descriptions $(D, \sqsubseteq)$ est le produit cartésien des deux ordres partiels $(C, \sqsubseteq_C)$ et $(P, \sqsubseteq_P)$ dont l'ordre $\sqsubseteq$ est défini par la définition 23. Le plus petit ancêtre commun de deux paires (prédicat, objet) est défini comme

$$\text{lcs}((p_i, o_i), (p_j, o_j)) = (\text{lcs}_P(p_i, p_j), \text{lcs}_C(o_i, o_j)).$$

Ainsi, pour un ensemble de triplets K, la *pattern structure* est définie comme suit :

$$\begin{aligned} G &= \{s \mid \langle s, p, o\rangle \in K\} \\ \delta(g) &= \min\{(p, o) \mid \langle g, p, o\rangle \in K\} \\ \delta(g_i) \sqcap \delta(g_i) &= \min\{\text{lcs}((p_i, o_i), (p_j, o_j)) \mid \forall (p_i, o_i) \in \delta(g_i)), \forall (p_j, o_j) \in \delta(g_j)\}. \end{aligned}$$

Les correspondances de Galois sont

$$\begin{aligned} A^{\square} &= \bigsqcap_{g \in A} \delta(g) \text{ pour tout } A \subseteq G \text{ et} \\ d^{\square} &= \{g \in G \mid d \sqsubseteq \delta(g)\} \text{ pour tout } d \in D. \end{aligned}$$

Proposition 1. $\delta(s_1) \sqsubseteq \delta(s_2) \Leftrightarrow \delta(s_1) \sqcap \delta(s_2) = \delta(s_2)$.

Démonstration.

$$\begin{aligned} \delta(x) \sqcap \delta(y) = \delta(y) \quad &\Leftrightarrow \quad \min(\{lcs((p_i, o_i), (p_j, o_j)) \mid \forall (p_i, o_i) \in \delta(x)), \forall (p_j, o_j) \in \delta(y)\}) = \delta(y) \\ &\Leftrightarrow \quad \forall (p_i, o_i) \in \delta(x), \exists (p_j, o_j) \in \delta(y) \text{ t.q. } lcs((p_i, o_i), (p_j, o_j)) = (p_j, o_j) \\ &\Leftrightarrow \quad \forall (p_i, o_i) \in \delta(x), \exists (p_j, o_j) \in \delta(y) \text{ t.q. } (p_i, o_i) \leqslant (p_j, o_j) \\ &\Leftrightarrow \quad \delta(x) \sqsubseteq \delta(y) \end{aligned}$$

□

La proposition est inversée par rapport à l'équation usuelle : $\delta(x) \sqsubseteq \delta(y) \Leftrightarrow \delta(x) \sqcap \delta(y) = \delta(x)$. En effet, les *pattern structures* s'appuient originellement sur un demi-treillis inférieur tandis que les relations `subClassOf` et `subPropertyOf` forment des demi-treillis supérieurs.

À partir de la base de connaissances de la figure 3.7 page 33, le treillis de concepts obtenu est représenté en figure 6.1. La similarité obtenue pour Paris et Nancy est alors

$$\delta(Paris) \sqcap \delta(Nancy) = \{(situéEn, Europe), (villeDe, France)\}.$$

6.1.3 Discussion

L'approche proposée ici pourrait être étendue en tenant compte de plusieurs ordres partiels sur un même ensemble de ressources. En effet, actuellement, la similarité entre une ville v_1 située en France et une ville v_2 située en Europe est que ce sont toutes les deux des villes situées dans un lieu. Cependant, la relation `situéEn` a vraisemblablement toutes les propriétés requises pour définir un ordre partiel entre les ressources qu'elle lie (il est légitime de penser que cette relation est réflexive, anti-symétrique et transitive), et il serait intéressant de pouvoir tenir compte de cet ordre. Ainsi, en sachant que la France est située en Europe, on pourrait dire que v_1 et v_2 sont toutes les deux situées en Europe.

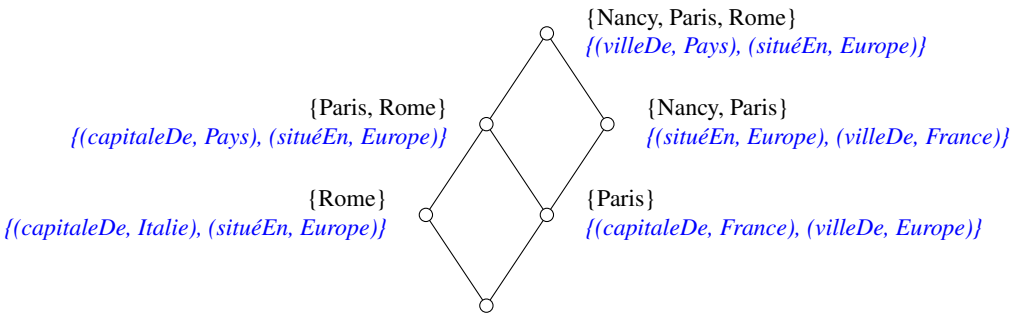


FIGURE 6.1 – Treillis de concepts obtenu avec la *pattern structure* tenant compte de l'ordre sur les prédicats et de celui sur les classes.

L'une des limites de cette approche est liée à la conception des bases de connaissances dans les LOD. En effet, alors que la taxonomie des classes est très utilisée et généralement détaillée, ce n'est pas le cas de la taxonomie des prédicats. Dans DBpedia par exemple, l'arbre correspondant à la taxonomie des classes a une profondeur de 8 classes. En revanche, l'arbre des prédicats a une profondeur de 4 avec seulement 3 prédicats à cette profondeur. Autrement dit, la similarité entre deux prédicats sera très souvent le sommet de la taxonomie. Dans YAGO, seulement une centaine de prédicats, très génériques, sont utilisés. Il n'est donc pas possible de tirer parti de la *pattern structure* introduite ici dans ces bases de connaissances. À ce jour, il existe très peu de bases de connaissances qui profitent pleinement de l'intérêt des taxonomies de prédicats.

6.2 Découverte de définitions

6.2.1 Motivation

Dans le premier chapitre, nous avons vu que le contenu de DBpedia est construit à partir d'informations extraites de l'encyclopédie Wikipédia. Dans Wikipédia, une catégorie C est une page spécifique de Wikipédia qui liste toutes les pages liées à C. Dans DBpedia, les catégories apparaissent dans le codomaine de la relation `dct:subject`. Par exemple, le triplet $\langle x, \texttt{dct:subject}, \texttt{Smartphones}\rangle$ indique que x appartient à la catégorie `Smartphones`.

En terme de représentation des connaissances et de raisonnement, le nom d'une catégorie est une simple expression syntaxique : la sémantique n'existe que par le sens qu'un agent humain apporte à la suite de lettres `s-m-a-r-t-p-h-o-n-e-s`. Ainsi, contrairement à ce que l'on pourrait souhaiter, une catégorie n'a pas de définition formelle dans DBpedia. Il est alors impossible de procéder à une classification des catégories dans la mesure où celles-ci ne sont pas définies en terme de conditions nécessaires et suffisantes. Nous utilisons donc les règles d'association et les redescriptions pour construire ces définitions. Nous recherchons des paires $(C, \{d_1, \ldots, d_n\})$ où C désigne une catégorie, comme `Nokia_Mobile_Phone` par exemple, et d_i correspond à une paire (prédicat, objet), comme (`manufacturer,Nokia`) par exemple. L'ensemble des d_i correspond alors à une description possible de C. En s'appuyant sur le formalisme des logiques de description, cette définition pourrait s'écrire $C \equiv d_1 \sqcap \cdots \sqcap d_n$, comme par exemple

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

Ces définitions sont particulièrement utiles sur DBpedia, puisqu'elles permettent d'améliorer la complétude de la base de connaissance en découvrant des triplets manquants. Supposons par exemple que la définition `Nokia_Mobile_Phone` $\equiv$ `Phone` $\sqcap \exists$`manufacturer.Nokia`, soit présente dans DBpedia. Dans

	(situéEn, France) f	(situéEn, Italie) i	(situéEn, Europe) e	(type, Ville) c	(type, Musée) m	Ville française VF	Capitale européenne CE	Musée parisien MP
Nancy	×		×	×		×		
Rome		×	×	×			×	
Paris	×		×	×		×	×	
Le_Louvre	×		×		×			×

FIGURE 6.2 – Contexte associé à la base de connaissances de la figure 3.7 page 33, à laquelle s'ajoutent trois catégories : Ville française, Capitale européenne et Musée parisien.

ce cas, si un élément x appartient à la catégorie `Nokia_Mobile_Phone`, alors x est une instance de `Phone` et x et `Nokia` sont liés par la relation `manufacturer`. Autrement dit, la présence du triplet $\langle x, \texttt{dct:subject}, \texttt{Nokia\_Mobile\_Phone}\rangle$ induit nécessairement la présence des triplets $\langle x, \texttt{rdf:type}, \texttt{Phone}\rangle$ et $\langle x, \texttt{manufacturer}, \texttt{Nokia}\rangle$. Réciproquement, si x est une instance de `Phone` et est manufacturé par Nokia, alors x devrait appartenir à la catégorie `Nokia_Mobile_Phone`.

6.2.2 Extraction et représentation des données

Dans notre approche, nous utilisons des triplets RDF de la ABox pour découvrir les définitions. Ces triplets peuvent être extraits depuis une base de connaissances $\mathcal{B}$ à l'aide d'une requête SPARQL `Q` : tous les triplets qui satisfont `Q` sont conservés.

$$K = \{(s,p,o) \in \mathcal{B} \mid \texttt{Q} \vDash (s,p,o)\}$$

Nous créons alors un contexte à partir de cet ensemble triplets K. Les objets du contexte sont les sujets de chaque triplet et les attributs du contexte sont les paires (prédicat, objet) de chaque triplet :

$$G = \{s \mid \langle s,p,o\rangle \in K\},$$
$$M = \{(p,o) \mid \langle s,p,o\rangle \in K\}.$$

La relation d'incidence est $I \subseteq G \times M$ telle que

$$sI(p,o) \Leftrightarrow \langle s,p,o\rangle \in K.$$

Le contexte obtenu à partir de la figure 3.7 page 33 est présenté en figure 6.2 et le treillis associé est représenté en figure 6.3. À partir de cet exemple, on a notamment $\{f,c\}' = \{Nancy, Paris\}$ et $\{VF\}' = \{Nancy, Paris\}$. Puisque $\{f,c\}' = \{VF\}'$, on peut alors construire la définition suivante :

$$\text{Ville\_Française} \equiv \exists \text{situéEn}.\{\text{France}\} \sqcap \exists \text{type}.\{\text{Ville}\}. \tag{6.1}$$

La définition d'une catégorie peut être pensée en terme de condition nécessaire et suffisante :

$$C \equiv d_1 \sqcap \cdots \sqcap d_n \text{ équivaut à } C \sqsubseteq d_1 \sqcap \cdots \sqcap d_n \text{ et } d_1 \sqcap \cdots \sqcap d_n \sqsubseteq C.$$

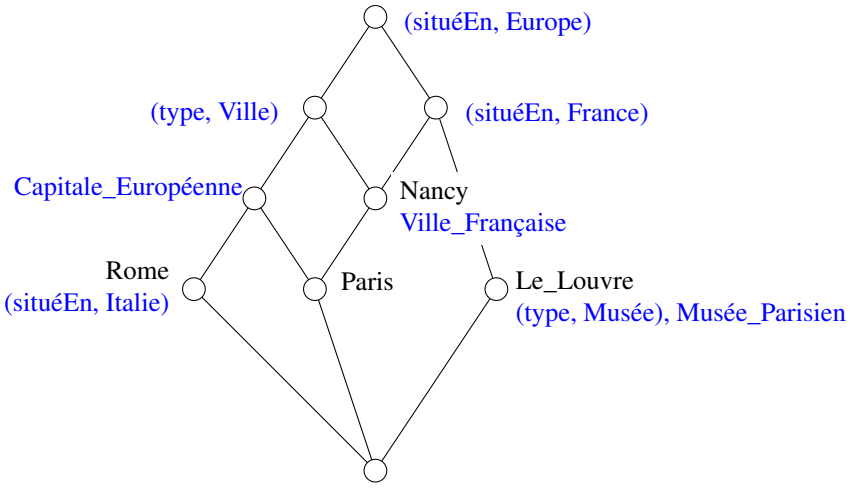


FIGURE 6.3 – Treillis de concepts construit à partir du contexte de la figure 6.2.

La condition nécessaire ($C \sqsubseteq d_1 \sqcap \cdots \sqcap d_n$) est l'ensemble des caractéristiques que doit impérativement avoir une ressource pour appartenir à une catégorie tandis que la condition suffisante ($d_1 \sqcap \cdots \sqcap d_n \sqsubseteq C$) est l'ensemble des caractéristiques qui assurent qu'une ressource appartient à une catégorie. Considérons par exemple la définition 6.1. D'une part, si x appartient à la catégorie `Ville_Française`, alors x est nécessairement une ville et est nécessairement située en France (condition nécessaire). D'autre part, si x est une ville et est située en France, alors x appartient de fait à la catégorie `Ville_Française` (condition suffisante).

Cependant, les données peuvent être incomplètes, voire erronées. Par exemple, il se peut qu'une ville située en France ne soit pas associée à la catégorie `Ville_française`. Dans ce cas, on perd l'égalité entre une classe $\mathtt{C_i}$ et sa description $\exists \mathtt{r}{:}\mathtt{C_j}$. Autrement dit $\{\mathtt{C_i}\}' \neq \{\exists \mathtt{r}{:}\mathtt{C_j}\}'$. Nous devons alors permettre une recherche approximative : $\{\mathtt{C_i}\}' \simeq \{\exists \mathtt{r}{:}\mathtt{C_j}\}'$. Pour cela, nous nous appuyons sur les règles d'association et les redescriptions.

6.2.3 Découverte de définitions avec les règles d'association

Pour rechercher des définitions avec les règles d'association, nous considérons conjointement la règle $X \to Y$ et sa réciproque $Y \to X$, et nous cherchons à estimer à quel point chacune de ces règles s'approche 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 25 (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,*

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

Bien que d'autres mesures puissent être utilisées, la confiance est la mesure de référence pour les règles d'association : c'est cette mesure qui a été utilisée dès l'introduction des règles d'association Luxenburger, 1991, et elle reste la mesure majoritairement utilisée. Nous avons donc choisi, dans ce travail, de nous appuyer sur cette mesure.

Avec les règles d'association, tous les attributs du contexte peuvent faire partie de l'antécédent comme de la conséquence d'une règle. Dans notre formulation, au contraire, on considère deux ensembles d'attributs distincts : une règle doit avoir d'un côté des catégories et de l'autre des descriptions. À cette fin, nous définissons une partition sur les attributs, et nous mettons en place un post-traitement sur les règles extraites.

Afin de permettre cette distinction sur les attributs, l'ensemble des paires (prédicat, objet) est partitionné en deux ensembles : $M = M_{subject} \cup M_{descr}$ avec $M_{subject} \cap M_{descr} = \varnothing$. L'ensemble $M_{subject}$ contient les paires qui permettent d'associer une catégorie à une ressource, c'est-à-dire l'ensemble des paires dont le prédicat est `dct:subject`. L'ensemble M_{descr} contient le reste des paires. On a donc

$$M_{subject} = \{(p,o) \in M \mid p = \texttt{dct:subject}\} \qquad \text{et} \qquad M_{descr} = \{(p,o) \in M \mid p \neq \texttt{dct:subject}\}$$

Pour que les règles obtenues définissent des classes, ce qui est l'objectif de notre approche, il faut donc que les règles d'association soient de la forme « Classes → Descriptions » ou « Descriptions → Classes ». C'est-à-dire que la règle $X \to Y$ est telle que $X \subseteq M_{subject}, Y \subseteq M_{descr}$ ou $X \subseteq M_{descr}, Y \subseteq M_{subject}$. Nous effectuons un post-traitement afin de filtrer les règles et de ne conserver que celles qui respectent cette condition. Nous nous appuyons sur le fait qu'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 a une confiance θ_0, alors la règle r_i a une confiance $\theta_i \geqslant \theta_0$. Donc, pour un seuil θ fixé, si r_0 est valide, l'ensemble des r_i est valide.

Pour ne conserver que les règles de la forme « Classes → Descriptions » ou « Descriptions → Classes », les règles sont donc 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). Autrement dit, seules les règles $X \to Y$ qui satisfont l'une des deux contraintes alternatives suivantes sont conservées :

1. l'antécédent ne contient que des classes ($X \subseteq M_{subject}$) et il y a au moins une description dans la conséquence ($Y \cap M_{descr} \neq \varnothing$) ;
2. l'antécédent ne contient que des descriptions ($X \subseteq M_{descr}$) et il y a au moins une classe dans la conséquence ($Y \cap M_{subject} \neq \varnothing$).

Par exemple, si l'on considère les règles suivantes :

r_0 $\{$(situéEn, France), Capitale_Européenne$\}$ → $\{$(type, Ville)$\}$
r_1 $\{$(situéEn, France)$\}$ → $\{$(type, Ville), Ville_Française$\}$
r_2 $\{$(situéEn, France)$\}$ → $\{$(type, Ville)$\}$
r_3 $\{$(situéEn, France)$\}$ → $\{$Ville_Française$\}$

La règle r_0 n'est pas conservée parce que l'antécédent contient à la fois une catégorie et une description. La règle r_1, en revanche, peut être décomposée en deux règles, r_2 et r_3. Seule r_3 est conservée. Si la confiance de sa réciproque est supérieure à θ, alors on obtient la quasi-définition

$$\text{Ville\_Française} \equiv \exists \text{situéEn}.\{\text{France}\}.$$

L'algorithme 5 présente le post-traitement effectué pour construire les quasi-définitions à partir des règles d'association.

6.2.4 Découverte de définitions avec les redescriptions

Dans cette section, nous détaillons l'utilisation de la fouille de redescriptions pour la découverte de définitions. Une redescription correspond à une définition en terme de condition nécessaire et suffisante :

Algorithme 5 : Construction des quasi-définitions à partir des règles d'association extraites.

Données : $\mathcal{A}$ l'ensemble des règles d'association extraites, C l'ensemble des catégories, D l'ensemble des descripteurs
Résultat : $\mathcal{Q}$ l'ensemble des quasi-définitions

Extraction
  $\mathcal{Q} \leftarrow \varnothing$;
  cat, descr ← false ;
  pour $i \in (1, |\mathcal{A}| - 1)$ **faire**
    // Pour chacune des règles d'association extraites
    $r_1 = (X_1 \rightarrow Y_1) \leftarrow \mathcal{A}[i]$;
    si $X_1 \subseteq C$ && $Y_1 \cap D \neq \varnothing$ **alors**
      // Si l'antécédent ne contient que des catégories et la conséquence contient des descripteurs
      cat ← true ;
    sinon si $X_1 \subseteq D$ && $Y_1 \cap C \neq \varnothing$ **alors**
      // Si l'antécédent ne contient que des descripteurs et la conséquence contient des catégories
      descr ← true ;
    fin
    si *cat* || *descr* **alors**
      // Si l'on se trouve dans l'un des deux cas précédents
      pour $j \in (i+1, |\mathcal{A}|)$ **faire**
        // Parmi les règles d'association restantes
        $r_2 = (X_2 \rightarrow Y_2) \leftarrow \mathcal{A}[j]$;
        si $X_1 \subseteq Y_2$ && $X_2 \subseteq Y_1$ **alors**
          // Si l'antécédent de r_1 est inclus dans la conséquence de r_2 et *vice versa*
          si *cat* && $X_2 \subseteq D$ **alors**
            // Si l'antécédent de r_1 ne contient que des catégories et l'antécédent de r_2 que des descripteurs
            $\mathcal{Q} \leftarrow \mathcal{Q} \cup (X_1 \leftrightarrow X_2)$;
            // La quasi-définition construite avec les antécédents de r_1 et r_2 est ajoutée
          sinon si *descr* && $X_2 \subseteq C$ **alors**
            // Si l'antécédent de r_1 ne contient que des descripteurs et l'antécédent de r_2 que des catégories
            $\mathcal{Q} \leftarrow \mathcal{Q} \cup (X_2 \leftrightarrow X_1)$;
            // La quasi-définition construite avec les antécédents de r_1 et r_2 est ajoutée
          fin
        fin
      fin
    fin
  fin
retourner $\mathcal{Q}$;

une catégorie est associée à des caractéristiques partagées par l'ensemble de ses membres, et ces caractéristiques sont propres aux membres de cette catégorie.

Pour appliquer la fouille de redescriptions à un ensemble de triplets, nous transformons dans un premier temps ces triplets dans un format qui pourra être utilisé par l'algorithme REREMI. Cette opération est similaire à la construction du contexte détaillée dans la section 6.2.2.

La fouille de redescriptions repose sur un système de partition des attributs en vues, ce qui est parfaitement adapté à notre cas, puisque nous souhaitons avoir les catégories d'un côté de notre définition, et les descriptions de l'autre. Les vues sont donc construites à partir de la partition définie en section 6.2.3. On a donc $M = M_{descr} \cup M_{subject}$ et $M_{descr} \cap M_{subject} = \varnothing$ où $M_{subject}$ est l'ensemble des attributs (p, o) tels que $p =$ `dct:subject` et M_{descr} est l'ensemble des attributs (p, o) tels que $p \neq$ `dct:subject`. Rechercher des définitions revient alors à rechercher des redescriptions $q_1 \leftrightarrow q_2$ où q_1 est une requête ne construite avec un ensemble d'attributs $C \in M_{subject}$ et q_2 est une requête construite à partir d'un ensemble d'attributs $D \in M_{descr}$.

Comme nous l'avons détaillé dans le chapitre 5, l'algorithme REREMI nous permet de tenir compte de variables numériques et catégorielles. Cependant, ici, nous utilisons seulement des valeurs binaires. Ce choix est motivé par la volonté de comparer les résultats obtenus avec les règles d'association d'une part, et d'avoir une approche agnostique sur les données traitées d'autre part. En effet, l'utilisation de données numériques nécessite d'avoir des informations supplémentaires sur les données, notamment sur la cardinalité des prédicats.

Recherche de définitions avec les règles de traduction

Nous nous sommes également intéressés à la recherche de définitions avec l'algorithme TRANSLATOR. Les données utilisées et le découpage en vues sont les mêmes que pour les redescriptions. L'algorithme extrait à la fois des redescriptions et des règles d'association. Dans notre approche, nous ne conservons que les redescriptions extraites. Nous ne pouvons pas ordonner les règles extraites selon une mesure de qualité telle que la confiance ou le coefficient de Jaccard, aussi nous nous appuyons sur l'ordre dans lequel les règles sont extraites.

Recherche de catégories incompatibles

Nous nous intéressons également à la recherche de catégories incompatibles, c'est-à-dire des catégories qui ne partagent aucune ressource : $C_i \sqcap C_j \equiv \bot$. L'intégration de ces connaissances est très utile dans le cadre du web des données. Tout d'abord, connaître les catégories incompatibles permet de réduire la complexité pour le moteur d'inférence. Par ailleurs, elle permet notamment de détecter les inconsistances. Supposons par exemple qu'il existe dans DBpedia un triplet qui indique que les catégories `Smartphones` et `Sports_cars` sont incompatibles. Un utilisateur qui essaierait d'associer la catégorie `Sports_cars` à une ressource appartenant déjà à `Smartphones` pourrait être stoppé, par un message d'erreur par exemple, parce que ces catégories ne sont pas compatibles. Il lui faudrait alors choisir entre les deux catégories. Ainsi, les classes incompatibles guident les utilisateurs qui ajoutent des connaissances.

Nous recherchons donc deux catégories $C_i, C_j \in M_{subject}$ telles que $C'_i \cap C'_j \simeq \varnothing$. Cela revient à rechercher des redescriptions avec un coefficient de Jaccard proche de 0. Plus précisément, nous considérons le coefficient suivant :

$$\frac{E_{10} + E_{01}}{E_{11} + E_{10} + E_{01}}$$

Cette nouvelle mesure a été intégrée au sein de SIREN, le reste de l'algorithme fonctionne de manière identique.

TABLE 6.1 – Comparaison des trois algorithmes utilisés pour rechercher des définitions. Les éléments en italique ne sont pas utilisés dans notre approche.

	ECLAT	REREMI	TRANSLATOR
Approche	Règles d'association	Redescriptions	Règles de traduction
Données	Bool.	Bool., *Num., Cat.*	Bool.
Mesure de qualité	Confiance $\frac{\lvert X' \cap Y' \rvert}{\lvert X' \rvert}$	Jaccard $\frac{\lvert X' \cap Y' \rvert}{\lvert X' \cup Y' \rvert}$	Compression basée sur MDL
Symétrique	Non	Oui	Oui/Non [18]
Expressivité	Conj.	Conj., *Disj., Nég.*	Conj.
Énumération exhaustive	Oui	Non	Non

6.2.5 Discussion

Principales différences entre les règles d'association et les redescriptions

Chacune des trois approches utilisées possède ses spécificités, synthétisées dans la table 6.1. L'algorithme ECLAT offre une énumération exhaustive (au seuil de support près) des règles d'association. Cependant, il est nécessaire d'opérer un post-traitement afin de construire les définitions. L'algorithme TRANSLATOR se distingue d'ECLAT et REREMI selon deux aspects : premièrement, il s'appuie sur MDL pour estimer la qualité d'une règle. Secondement, la qualité d'une règle dépend de celles déjà trouvées, ce qui met l'accent sur la qualité de l'*ensemble* de règles extraites plutôt que sur chacune des règles prises indépendamment.

Proposition 2. *Pour un même seuil, toutes les redescriptions sont nécessairement des quasi-définitions (déf. 25 page 68) :*

$$\min(\mathrm{conf}(A \to B), \mathrm{conf}(B \to A)) \geqslant \mathrm{jacc}(A \leftrightarrow B)$$

Démonstration. On a

$$\mathrm{jacc}(A \leftrightarrow B) = \frac{|A' \cap B'|}{|A' \cup B'|}.$$

et

$$\min(\mathrm{conf}(A \to B), \mathrm{conf}(B \to A)) = \min\left(\frac{|A' \cap B'|}{|A'|}, \frac{|A' \cap B'|}{|B'|}\right) = \frac{|A' \cap B'|}{\max(|A'|, |B'|)}.$$

Puisque $A' \subseteq A' \cup B'$, on a $|A'| \leqslant |A' \cup B'|$ et donc

$$\frac{|A' \cap B'|}{|A'|} \geqslant \frac{|A' \cap B'|}{|A' \cup B'|}.$$

18. TRANSLATOR extrait à la fois des règles symétriques et non symétriques. Dans notre approche, seules les règles symétriques sont conservées.

C	X	Y
	×	
×	×	×
×	×	×
		×

Eclat $C \equiv X \sqcap Y$
ReReMi $C \equiv X \sqcap Y$

C	X	Y
×	×	×
×	×	×

Eclat $C \equiv X \sqcap Y$
ReReMi $C \equiv X,\quad C \equiv Y$

C	X	Y
		×
×	×	×
×	×	×

Eclat $C \equiv X \sqcap Y, C \equiv X$
ReReMi $C \equiv X$

FIGURE 6.4 – Différences de règles extraites entre ECLAT et REREMI.

En appliquant le même raisonnement avec B', on obtient

$$\frac{|A' \cap B'|}{|B'|} \geqslant \frac{|A' \cap B'|}{|A' \cup B'|}.$$

On peut donc en déduire $\min(\text{conf}(A \to B), \text{conf}(B \to A)) \geqslant \text{jacc}(A \leftrightarrow B)$. □

On peut donc s'attendre à ce que les définitions découvertes avec REREMI soient un sous-ensemble des définitions découvertes avec ECLAT. Cependant, REREMI repose sur une heuristique qui évite d'impliquer des attributs qui n'ont pas d'incidence sur le support. Autrement dit, REREMI ne va pas sélectionner les mêmes attributs qu'ECLAT. Quelques exemples sont représentés dans la figure 6.4.

Notion de redondance dans les règles

Pour mieux appréhender les règles obtenues, nous introduisons ici la notion de redondance.

À partir de l'ordre partiel défini sur les paires (prédicat, objet), on définit l'attribut d'une quasi-définition comme redondant s'il est plus général qu'un autre attribut présent dans la quasi-définition.

Définition 26 (Attribut redondant)**.** *Soit un ensemble d'attributs* $X = \{(p_1, o_1), \ldots (p_n, o_n)\}$*. Un attribut* (p_i, o_i) *est dit redondant s'il existe* $(p_j, o_j) \in X$ *avec* $i \neq j$ *tel que* p_i `subP` p_j *et* o_i `subC` o_j*.*

Nous pouvons alors nous intéresser à la redondance des quasi-définitions.

Définition 27 (Définition redondante)**.** *Soient deux définitions* D_1 *et* D_2*. On dit que* D_1 *est redondante par rapport à* D_2 *si* D_1 *peut être inférée à partir de* D_2 *et de la base de connaissances.*

On considère les quasi-définitions valides suivantes :

$$Q_1 : C \leftrightarrow (p_0, o_0), (p_1, o_1)$$
$$Q_2 : C \leftrightarrow (p_0, o_0), (p_2, o_2)$$

où C est une catégorie et $(p_i, o_i), i \in \{0, 1, 2\}$ sont des paires (prédicat, objet). Les paires (p_1, o_1) et (p_2, o_2) peuvent être comparables ou non. Soit les paires sont comparables ($(p_1, o_1) \leqslant (p_2, o_2)$), et la quasi-définition Q_2 est dite *redondante par rapport* à Q_1 dans le sens où Q_2 peut être inférée à partir de Q_1. Soit les paires sont incomparables et nous obtenons alors deux définitions alternatives d'une même catégorie. Ces deux définitions peuvent alors être considérées comme une seule définition disjonctive.

Considérons par exemple les deux définitions suivantes :

$$\textit{Carré} \equiv \textit{Rectangle} \sqcap \textit{Losange} \sqcap \textit{Quadrilatère}$$
$$\textit{Carré} \equiv \textit{Rectangle} \sqcap \textit{Losange} \sqcap \textit{Parrallélogramme}.$$

La première définition est *redondante* par rapport à la deuxième, car *Quadrilatère* peut être inféré à partir de *Parallélogramme* [19]. Cela ne signifie pas nécessairement que la première règle n'est pas utile. Une définition est une condition nécessaire et suffisante : être à la fois un rectangle, un losange et un quadrilatère est une condition *suffisante* pour être un carré. Si la finalité est de trouver quelles ressources sont des carrés, alors la définition redondante est préférable, puisque sa condition suffisante est plus « facile à respecter » car les parallélogrammes sont nécessairement des quadrilatères.

Considérons maintenant les deux définitions suivantes :

$$Carré \equiv Rectangle \sqcap \exists possède.Côtés\_consécutifs\_égaux$$
$$Carré \equiv Losange \sqcap \exists possède.Angle\_droit$$

Dans ce cas, nous avons deux définitions qui ne sont pas comparables. Ces deux définitions peuvent être considérées comme une seule définition disjonctive :

$$Carré \equiv (Rectangle \sqcap \exists possède.Côtés\_consécutifs\_égaux) \sqcup (Losange \sqcap \exists possède.Angle\_droit)$$

Synthèse du chapitre

Dans ce chapitre, nous avons présentés deux contributions : la définition d'une *pattern structure* pour le web des données et la comparaison des règles d'association et des redescriptions pour la découverte de définitions. Nous avons défini une *pattern structure* permettant de tenir compte des hiérarchies de classes et de prédicats définies par les relations `rdfs:subClassOf` et `rdfs:subPropertyOf`. La *pattern structure* que nous avons définie permet de représenter les ressources sous la forme d'un treillis de concepts, ce qui offre un support à la classification et la visualisation de ces données. Nous avons également proposé d'appliquer la fouille de règle d'association et la fouille de redescriptions à des données RDF, et nous avons comparé les deux approches, en mettant en avant leur similarité et leur complémentarité.

19. Dans ce cas précis *Quadrilatère* peut être inféré également à partir de *Rectangle* et de *Losange*. Les attributs *Quadrilatère* et *Parallélogramme* sont des attributs redondants au sein de leurs règles respectives.

Chapitre 7

Expérimentations

Sommaire

Dans les chapitres précédents, nous avons identifié et présenté trois approches permettant de découvrir des définitions dans le web des données. Nous détaillons dans ce chapitre les expérimentations menées sur DBpedia pour comparer les approches introduites. Trois algorithmes sont utilisés : l'algorithme ECLAT *via* la plateforme CORON (M. Kaytoue et al., 2010) pour les règles d'association, l'algorithme REREMI *via* l'interface SIREN pour les redescriptions et l'algorithme TRANSLATOR pour les règles de traduction. Nous présentons ici les principaux paramètres de ces algorithmes et les valeurs fixées pour ces paramètres.

Le processus d'expérimentation, présenté en figure 7.1, est identique pour les trois approches. Ce chapitre comporte trois sections principales, correspondant aux différentes étapes du processus. La première section présente les jeux de données extraits et leurs caractéristiques. La deuxième section détaille, pour chaque algorithme, les opérations effectuées afin d'obtenir les définitions à partir des données de départ. Les définitions obtenues sont présentées dans la troisième section. Enfin, une quatrième section conclut ce chapitre.

Quatre expérimentations ont été menées au cours de cette thèse. Elles sont résumées dans la table 7.1. La première expérimentation compare les trois approches sur quatre jeux de données différents. La

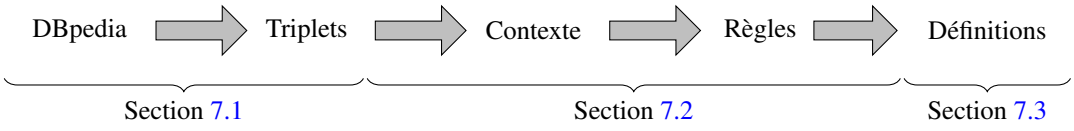


FIGURE 7.1 – Plan du chapitre 7.

deuxième expérimentation propose une comparaison plus poussée des règles d'association et des redescriptions. Les troisième et quatrième expérimentations explorent l'expressivité des redescriptions en s'intéressant à d'autres formes de définitions.

TABLE 7.1 – Vue d'ensemble des expérimentations effectuées

	Algorithme	Jeux de données	Règles recherchées
1	ECLAT, REREMI, TRANSLATOR	`Turing_Award_laureates`, `Smartphones`, `Sports_cars`, `French_films`	$C_1 \sqcap \cdots \sqcap C_m \equiv \exists p_1.\{o_1\} \sqcap \cdots \sqcap \exists p_n.\{o_n\}$
2	ECLAT, REREMI	`Turing_Award_laureates`, `Women_Mathematicians`, `Mathematicians`, `Samsung_Galaxy`, `Smartphones`, `Sports_cars`, `Hospital_films`, `Road_movies`, `French_films`	$C \equiv \exists p_1.\{o_1\} \sqcap \cdots \sqcap \exists p_n.\{o_n\}$
3	REREMI	`Smartphones`	$C \equiv \exists p_1.\{o_1\} \sqcup \cdots \sqcup \exists p_n.\{o_n\}$
4	REREMI	`Turing_Award_laureates`, `Smartphones`, `Sports_cars`, `French_films`	$C_i \not\equiv C_j$

7.1 Jeux de données

Les expérimentations ont été menées sur neuf jeux de données issus de DBpedia. Les triplets sont extraits à partir d'une requête SPARQL. Celle-ci permet d'extraire tous les triplets dont le sujet est membre d'une catégorie donnée. La figure 7.2 présente la requête utilisée. Dans un premier temps, nous recherchons toutes les ressources associées à une catégorie (ligne 2), et nous extrayons tous les triplets qui ont pour sujet l'une de ces ressources (ligne 3). Le reste de la requête est un ensemble de filtres pour ne considérer que des données de DBpedia. Ainsi, le sujet doit être une ressource (ligne 6), tandis que le prédicat doit faire partie de DBpedia (lignes 9-10) ou être `dct:subject` (ligne 11) ou `rdf:type` (ligne 12). Nous ne tenons pas compte des informations issues de wordnet (ligne 15) et nous retirons les prédicats indiquant une URL (lignes 16-17). De plus, nous avons retiré les prédicats liés à Wikipédia (lignes 18-20). Enfin, nous nous assurons que l'objet est une catégorie (ligne 23) ou appartient à DBpedia (lignes 24-25). Cela permet notamment d'éviter d'obtenir des ressources de Yago ou Wikidata.

Pour constituer nos jeux de données, nous avons sélectionné neuf catégories réparties en trois domaines. Le recours à plusieurs domaines permet de vérifier que les observations ne sont pas propres à un domaine en particulier. Nous avons utilisé des tailles de jeux de données variées. Les petits jeux de données forment des contextes de moins de 100 objets et 1500 attributs, ce qui signifie qu'il y a moins de 1500 paires (prédicat, objet) uniques dans les triplets extraits. Les jeux de données de taille moyenne ont environ 600 objets pour 2000 à 10000 attributs. Enfin, les grands jeux de données ont plus de 1500 objets et 10000 attributs. Il n'y a pas de catégories d'objets manufacturés de cette taille, nous avons donc seulement deux grands jeux de données. À défaut, nous avons donc considéré deux jeux de données de taille moyenne pour les objets manufacturés. La table 7.2 fournit une vue d'ensemble des jeux de données extraits.

```
SELECT DISTINCT ?s ?p ?o WHERE {
    ?s dct:subject dbc:Cat .
    ?s ?p ?o .

    FILTER (
          STRSTARTS(STR(?s), "http://dbpedia.org/resource/")
        )
    FILTER (
          STRSTARTS(STR(?p), "http://dbpedia.org/property/") OR
          STRSTARTS(STR(?p), "http://purl.org/dc/terms/") OR
          STRSTARTS(STR(?p), "http://dbpedia.org/ontology/") OR
          STRSTARTS(STR(?p), "http://www.w3.org/1999/02/22-rdf-syntax-ns#")
        )
    FILTER (
          (?p != dbp:wordnet_type) AND
          (?p != dbp:website) AND
          (?p != dbp:url) AND
          (?p != dbo:wikiPageWikiLink) AND
          (?p != dbo:wikiPageExternalLink) AND
          (?p != dbo:thumbnail)
        )
    FILTER (
          STRSTARTS(STR(?o), "http://dbpedia.org/resource/Category:") OR
          STRSTARTS(STR(?o), "http://dbpedia.org/ontology/") OR
          STRSTARTS(STR(?o), "http://dbpedia.org/resource/")
        )
}
```

FIGURE 7.2 – Requête utilisée pour extraire les jeux de données. Chaque triplet extrait a pour sujet une ressource (?*s*) appartenant à (`dct:subject`) une catégorie donnée (`dbc:Cat`). Le préfixe `dbc` est utilisé pour toutes les catégories de DBpedia, tandis que le préfixe `dct` est pour les ressources issues de Dublin Core, un vocabulaire générique permettant de décrire des ressources.

Avec les jeux de données `Women_Mathematicians` et `Mathematicians` d'un côté et `Samsung_Galaxy` et `Smartphones` de l'autre, nous considérons des catégories et des « sous-catégories ». Cela signifie que tous les triplets présents dans le corpus `Women_Mathematicians` (resp. `Samsung_Galaxy`) sont présents dans le corpus `Mathematicians` (resp. `Smartphones`).

Les statistiques des jeux de données sont présentées dans la table 7.3. Les contextes créés à partir des jeux de données ont des attributs avec un faible support, ce qui implique que les contextes sont très peu denses. Cependant, les contextes sur les objets manufacturés sont plus denses que ceux des personnes et des films.

Le nombre de prédicats est faible par rapport au nombre de paires (prédicat, objet), signifiant que beaucoup d'attributs partagent le même prédicat, et diffèrent uniquement sur leur objet.

TABLE 7.2 – Vue d'ensemble des jeux de données.

	Persons	Objects	Films
Small	`Turing_Award`	`Samsung_Galaxy`	`Hospital_films`
Medium	`Women_Mathematicians`	`Smartphones`, `Sports_cars`	`Road_movies`
Large	`Mathematicians`	—	`French_films`

TABLE 7.3 – Statistiques sur les jeux de données extraits.

Dataset	Triples	$\|G\|$	$\|M\|$	$\|M_{subj}\|$	$\|M_{descr}\|$	Predicates	Density
`Samsung_Galaxy`	940	59	277	30	247	33	5.2e−2
`Turing_Award_laureates`	2642	65	1360	503	857	35	2.2e−2
`Hospital_films`	1984	71	1265	490	775	46	1.6e−2
`Women_mathematicians`	9652	552	4243	1776	2467	98	2.9e−3
`Smartphones`	8418	598	2089	359	1730	98	5.8e−3
`Sports_cars`	9047	604	2730	435	2295	61	4.7e−3
`Road_movies`	20056	689	9314	2652	6662	103	2.4e−3
`Mathematicians`	32536	1660	12279	3848	8431	202	1.2e−3
`French_films`	121496	6039	25487	6028	19459	111	6.4e−4

7.2 Méthodologie

7.2.1 Obtention de définitions

Avec les règles d'association

Nous utilisons la plateforme CORON pour extraire les règles d'association. Cette plateforme implémente plusieurs algorithmes de fouille de règles d'association. Nous utilisons ici ECLATZ, qui s'appuie sur ECLAT pour extraire les motifs fréquents. Les règles extraites sont construites à partir des motifs fréquents et fermés. Le support et la confiance minimum sont fixés à 5% et 50% respectivement.

Une fois les règles d'association extraites depuis la plateforme CORON, un post-traitement en deux étapes est effectué. Dans un premier temps, les règles sont filtrées selon les critères détaillés dans la section 6.2.3. Seules les règles dont la prémisse ne contient pas à la fois des catégories et des descriptions sont conservées. Les attributs dans la conséquence de ces règles sont également filtrés pour n'avoir que des descriptions ou des catégories selon l'antécédent. Dans un second temps, nous pouvons extraire les quasi-définitions à partir des règles restantes. Nous cherchons donc des paires de règles d'association (R_1, R_2) telles que l'antécédent de R_1 est inclus dans la conséquence de R_2, et inversement. L'ensemble des paires de règles obtenues forme notre corpus de quasi-définitions.

Avec les redescriptions

Nous utilisons ici l'algorithme REREMI, qui est l'un de ceux proposés par l'interface SIREN. Celle-ci permet de configurer de nombreux paramètres de l'algorithme, nous ne détaillons ici que les principaux. Concernant la forme des règles extraites, il est possible de considérer des conjonctions, des disjonctions et des négations en spécifiant si ces opérateurs peuvent être trouvés dans la partie gauche de la redescription, dans sa partie droite, ou dans les deux. Le nombre maximal d'attributs de chaque côté de la redescription doit également être spécifié. Concernant la méthode de recherche et de *scoring* des redescriptions candidates, le support, le coefficient de Jaccard et la p-valeur sont utilisés.

Avec les règles de traduction

TRANSLATOR possède plusieurs paramètres permettant de personnaliser l'heuristique utilisée pour la fouille de règles de traduction. Dans l'expérience avec TRANSLATOR, nous avons laissés tous les paramètres par défaut.

7.2.2 Méthode d'évaluation

Nous avons recours à une évaluation humaine pour déterminer la validité des règles obtenues. Dans un premier temps, nous avons choisi de faire appel à trois évaluateurs pour évaluer les corpus `Turing_Award_laureates`, `Smartphones`, `Sports_cars` et `French_films`. Étant donnée une définition $C \leftrightarrow d_0, \ldots, d_n$ d'un jeu de données X, où C est une catégorie et d_i une description, chaque évaluateur devait répondre par oui ou par non à la question suivante :

> « Dans le domaine de X, est il équivalent d'appartenir à C et d'avoir les propriétés d_0 et … et d_n ? ».

Le recours à trois évaluateurs plutôt qu'un seul permet de tester si les opinions sur la validité des règles divergent ou non. En cas de divergence, nous retenons l'opinion majoritaire. Cependant, les évaluations montrent que les définitions recherchées ne sont pas sujettes à controverse pour les évaluateurs : plus de 95% des règles sont évaluées de la même façon par les trois évaluateurs, ce qui laisse penser que nous aurions pu n'utiliser qu'un seul évaluateur. Les autres jeux de données, qui ont été utilisés plus tardivement, n'ont donc été évalués que par une seule personne.

Par la suite, nous utiliserons les notations suivantes :

$\mathcal{A}$ est l'ensemble des règles d'association extraites par CORON

$\mathcal{Q}$ est l'ensemble des définitions candidates (quasi-définitions, redescriptions ou règles de traduction) obtenues. On utilisera la notation en indice pour distinguer les règles obtenues par une approche en particulier : $\mathcal{Q}_X$ désigne l'ensemble des règles obtenues par l'algorithme X Par exemple, $\mathcal{Q}_{\text{REREMI}}$ est l'ensemble des redescriptions obtenues par REREMI.

$\mathcal{D}$ est l'ensemble des règles (quasi-définitions, redescriptions ou règles de traduction) évaluées comme valides. On parlera alors de définitions, et on a donc $\mathcal{D} \subseteq \mathcal{Q}$. De façon similaire à précédemment, on notera $\mathcal{D}_X$ désigne l'ensemble des définitions obtenues *via* l'algorithme X

$\mathcal{C}$ est l'ensemble des catégories définies, et $\mathcal{C}_X$ l'ensemble des catégories définies par l'algorithme X.

Précision et rappel

Nous utilisons la précision et le rappel (Rijsbergen, 1979) pour évaluer et comparer les différentes approches. La figure 7.3 résume le calcul de la précision et du rappel dans une tâche de classification usuelle : on considère un ensemble d'objets N qui appartiennent ou non à une classe, et un processus qui permet d'associer automatiquement un objet à une classe.

Pour calculer la précision et le rappel sur un jeu de données, il est nécessaire de connaître l'ensemble des règles valides qui peuvent être extraites de ce jeu de données (A sur le schéma de la figure 7.3). Cependant, nous ne connaissons pas ces règles, et il n'existe pas de *gold standard* pour la tâche que nous souhaitons effectuer. Pour comparer les algorithmes entre eux, nous construisons un *silver standard* : nous considérons que l'ensemble des règles valides retournées par ces algorithmes constitue l'intégralité des définitions qui peuvent être extraites du jeu de données. Le rappel est donc calculé par rapport aux règles retournées par les autres algorithmes :

$$\text{rappel}_{\mathcal{D}}(X) = \frac{\mathcal{D}_X}{\mathcal{D}}$$

	Valide	Non valide	
Retrouvée	$A \cap B$	$\overline{A} \cap B$	B
Non retrouvée	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$	$\overline{B}$
	A	$\overline{A}$	N

$$\text{precision} = \frac{A \cap B}{B}$$
$$\text{rappel} = \frac{A \cap B}{A}$$

FIGURE 7.3 – Calcul de la précision et du rappel. A est l'ensemble des éléments valides et B l'ensemble des éléments retrouvés par les différents algorithmes. Ici, les éléments pourront être les règles, les catégories ou les sujets des triplets.

Nous utilisons également le rappel calculé sur les catégories définies :

$$\text{rappel}_{\mathcal{C}}(X) = \frac{\mathcal{C}_X}{\mathcal{C}}$$

En revanche, les règles extraites étant évaluées manuellement, nous pouvons aisément calculer la précision :

$$\text{precision}(X) = \frac{|\mathcal{D}_X|}{|\mathcal{Q}_X|}$$

7.3 Résultats

Les règles extraites dépendent du domaine du jeu de données et ne peuvent pas être généralisées à l'intégralité de DBpedia. Ainsi, une définition $A \cong B$ extraite à partir d'un jeu de données D doit être interprétée $A \sqcap D \cong B \sqcap D$. Pour découvrir des définitions plus générales, il serait nécessaire de prendre des jeux de données plus larges en entrée, en considérant par exemple un jeu de données à propos de personnes au lieu de `Turing_Award_laureates`. Cela entraîne d'autres difficultés, notamment de passage à l'échelle. Il faudrait alors considérer un échantillonnage des données. C'est une piste intéressante qui reste à explorer dans de futurs travaux. Pour illustrer les résultats des différentes expérimentations, des définitions extraites sont reportées dans les tables 7.4, 7.6, 7.8 et 7.10. Elles sont numérotées de 1 à 70.

7.3.1 Expérimentation 1 : Comparaison des trois approches

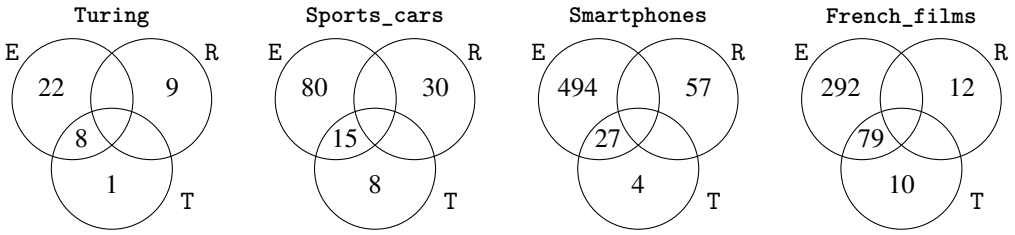
La table 7.4 présente quelques règles découvertes, et la table 7.5 donne des statistiques sur les résultats obtenus.

Les algorithmes sont comparés sur la base des définitions extraites et des catégories définies. La figure 7.4 présente des diagrammes de Venn pour chaque corpus. Elle se divise en trois sous-figures. La première sous-figure (7.4a) présente 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 sont trouvées à la fois par ECLAT et TRANSLATOR. Au total, ECLAT a extrait 30 définitions. La sous-figure 7.4b présente le 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 notamment le cas des règles 2, 8 à 10, 12, 14, et 17 à 20 dans la table 7.4.

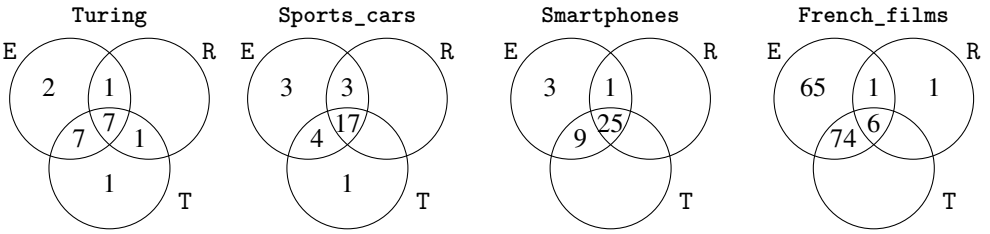
Le rappel, tel que défini précédemment, ne peut pas être utilisé comme une mesure de performance. En effet, certaines règles se recouvrent (ont des attributs en commun des deux côtés). C'est le cas des règles 6 à 10 (table 7.4) qui définissent toutes la catégorie `McLaren_vehicles`. Tandis que TRANSLATOR

TABLE 7.4 – Exemples de 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.

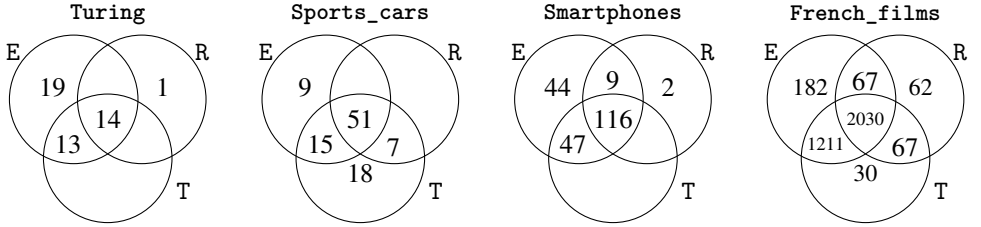
	Turing_Award_laureates	
R	**Harvard_University_alumni** ≅ ∃.almaMater{Harvard_University}	R1
ET	**Harvard_University_alumni** ⊓ **Turing_Award_laureates** ≅ Agent ⊓ Person ⊓ Scientist ⊓ ∃.almaMater{Harvard_University}	R2
E	**Turing_Award_laureates** ≅ Agent ⊓ Person ⊓ ∃.award{Turing_Award}	R3
ET	**Turing_Award_laureates** ≅ Agent ⊓ Person ⊓ 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** ≅ Automobile ⊓ MeanOfTransportation ⊓ ∃.assembly{Woking} ⊓ ∃.assembly{Surrey} ⊓ ∃.assembly{England} ⊓ ∃.bodyStyle{Coupé} ⊓ ∃.manufacturer{McLaren_Automotive}	R8
E	**McLaren_vehicles** ⊓ **Sports_cars** ≅ Automobile ⊓ MeanOfTransportation ⊓ ∃.assembly{England} ⊓ ∃.assembly{Surrey} ⊓ ∃.bodyStyle{Coupé}	R9
E	**McLaren_vehicles** ⊓ **Sports_cars** ≅ Automobile ⊓ MeanOfTransportation ⊓ ∃.assembly{Surrey} ⊓ ∃.bodyStyle{Coupé}	R10
	Smartphones	
ET	**Firefox_OS_devices** ⊓ **Open-source_mobile_phones** ⊓ **Smartphones** ⊓ **Touchscreen_mobile_phones** ≅ Device ⊓ ∃.operatingSystem{Firefox_OS}	R11
R	**Nokia_mobile_phones** ≅ ∃.manufacturer{Nokia}	R12
ET	**Nokia_mobile_phones** ⊓ **Smartphones** ≅ 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** ≅ Film ⊓ Q11424 ⊓ Work ⊓ ∃.director{Georges_Méliès}	R18
ET	**Films_directed_by_Jean_Rollin, French_films** ≅ Film ⊓ Q11424 ⊓ Work ⊓ ∃.director{Jean_Rollin}	R19
ET	**Film_scores_by_Gabriel_Yared, French_films** ≅ Film ⊓ Q11424 ⊓ Work ⊓ ∃.musicComposer{Gabriel_Yared}	R20



(a) Définitions extraites.



(b) Catégories définies.



(c) Triplets inférés.

FIGURE 7.4 – Définitions extraites, catégories définies et triplets inférés par ECLAT (E), REREMI (R) et TRANSLATOR (T) pour chaque jeu de données. Les zones non remplies correspondent à 0.

TABLE 7.5 – Expérience 1 : Évaluation des résultats. Pour chaque jeu de données, le nombre de définitions candidates extraites ($|\mathcal{Q}|$) et évaluées comme valides ($|\mathcal{D}|$) est reporté, ainsi que le nombre moyen de catégories ($|\overline{C_i}|$) et de descriptions ($|\overline{D_i}|$) par règle. La précision correspond au *silver standard* $\frac{\mathcal{D}_X}{\mathcal{D}}$.

(a) `Turing_Award_laureates`

X	ECLATZ	REREMI	TRANSLATOR
$\mathcal{Q}_X$	47	12	11
$\mathcal{D}_X$	30	9	9
Préc.	.64	.75	.85
$\overline{\lvert C_i\rvert}$–$\overline{\lvert D_i\rvert}$	2–4	1–1	3–5

(b) `Sports_cars`

X	ECLATZ	REREMI	TRANSLATOR
$\mathcal{Q}_X$	132	52	31
$\mathcal{D}_X$	95	30	23
Préc.	.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) `Smartphones`

X	ECLATZ	REREMI	TRANSLATOR
$\mathcal{Q}_X$	810	98	41
$\mathcal{D}_X$	521	57	31
Préc.	.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) `French_films`

X	ECLATZ	REREMI	TRANSLATOR
$\mathcal{Q}_X$	546	36	93
$\mathcal{D}_X$	371	12	89
Préc.	.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

n'extrait qu'une seule règle (8), REREMI extrait 2 règles (6 et 7) et ECLAT en extrait 9 (seules 8 à 10 sont reportées ici).

La table 7.2 page 78 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 ont 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 en figure 7.4b.

D'après la figure 7.4b, 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 $A \to B$ a le même support que la règle $A \to B,C$, alors seule la règle $A \to B,C$ est conservée. En revanche si le support de $A \to B$ est strictement supérieur au support de $A \to B,C$, ECLAT génère deux règles. En conséquence, on obtient avec ECLAT des règles qui ne diffèrent que d'un attribut, comme les règles 9 et 10 par exemple, alors que TRANSLATOR ne génère que la règle 8 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 6 et 7 obtenues par REREMI et 8, produite par ECLAT (table 7.4). 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 12 et 13.

Une autre conséquence de l'heuristique employée par REREMI est la taille (c.-à-d. le nombre d'attributs) des définitions : les définitions générées par REREMI contiennent moins d'attributs que les définitions de TRANSLATOR et ECLAT. REREMI a en moyenne entre 1 et 2 attributs de chaque côté de la

TABLE 7.6 – Résultats des expérimentations pour chaque jeu de données et chaque algorithme.. La tableau ci-dessous rapporte dans sa partie gauche le redescriptions ($|R|$) extraites, le nombre de redescriptions valides ($|V_R|$) et le nombre de catégories définies à partir de ces redescriptions valides ($|Cat_R|$). Enfin, la précision moyenne obtenue, pour un coefficient de Jaccard fixé à 0.5 est indiquée dans la dernière colonne (P_R La partie droite comporte le nombre de règles d'association ($|AR|$) extraites, le nombre de quasi-définitions ($|QD|$) obtenues et le nombre de quasi-définitions valides ($|V_{QD}|$). Le nombre de catégories définies ($|Cat|$) est indiqué, ainsi que la précision moyenne sur le jeu de données pour une confiance fixée à .5 ($P_{QD} = \frac{|V_{QD}|}{|QD|}$).

Jeu de données	Redescriptions				Règles d'association				
	$\|R\|$	$\|V_R\|$	$\|Cat_R\|$	P_R	$\|AR\|$	$\|QD\|$	$\|V_{QD}\|$	$\|Cat_{QD}\|$	P_{QD}
`Turing_Award_laureates`	33	16	16	0.48	563803	57	34	18	0.60
`Women_Mathematicians`	2	2	2	1.00	20483	6	5	5	0.83
`Mathematicians`	12	12	7	1.00	96807	29	24	14	0.83
`Samsung_Galaxy`	6	4	2	0.67	47004	20	20	5	1.00
`Smartphones`	24	22	17	0.92	3558380	34	26	12	0.76
`Sports_cars`	25	18	15	0.72	75030	49	35	17	0.71
`Hospital_films`	31	11	9	0.35	4345921	18	5	2	0.28
`Road_movies`	13	9	9	0.69	333491	34	18	13	0.53
`French_films`	6	6	4	1.00	371771	186	165	106	0.89

définition tandis que pour ECLAT et TRANSLATOR, il y a en moyenne 3 catégories et 4 descriptions par définition.

Autre conséquence, les conjonctions dans les définitions extraites par REREMI n'ont pas exactement la même sémantique que celles extraites par ECLAT et TRANSLATOR. Par exemple, si l'on considère la définition 15, 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 14, 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 14 nous parait préférable à la définition 15, parce qu'elle est apparue plus facile à interpréter.

On peut estimer le nombre de triplets qui peuvent être inferrés à partir des définitions obtenues, et qui ne sont pas déjà dans la base de connaissances. Par exemple, étant donné la règle 16, si une ressource r appartient à la catégorie `Pathé_Films` (c.-à-d. le triplet $\langle r, $ `:subject, Pathé_Films` $\rangle \in KB$), alors on s'attend à ce que le triplet $\langle r,$ `distributor, Pathé` $\rangle$ soit présent dans la base. La figure 7.4c indique, pour chaque jeu de données, le nombre de triplets inférés à partir des définitions de $\mathcal{D}$ qui n'étaient pas dans DBpedia.

On remarque que les règles d'association permettent d'obtenir la meilleure complétion en terme de triplets. Elles sont suivies par les règles de traduction. Les redescriptions, en revanche, ne permettent de retrouver qu'un faible nombre de triplets en comparaison des deux autres approches.

7.3.2 Expérimentation 2 : Redescriptions et règles d'association

La deuxième expérimentation laisse de côté les règles de traduction pour pousser la comparaison entre les redescriptions et les règles d'association. Par rapport à l'expérience précédente, davantage de jeux de données ont été utilisés, et une seule catégorie par règle est tolérée (voir la table 7.1). Les résultats obtenus, synthétisés dans la table 7.6, recoupent les observations faites précédemment. La figure 7.5

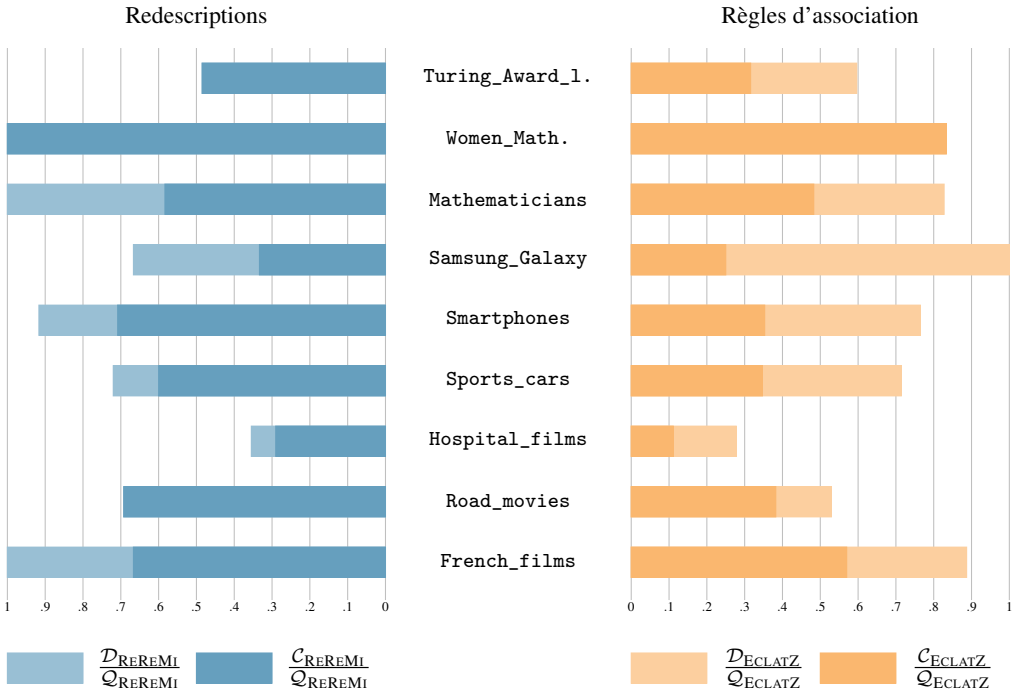


FIGURE 7.5 – Nombre de règles valides et de catégories définies proportionnellement au nombre de règles retournées, par algorithme et par jeu de données.

représente le nombre de définitions obtenues proportionnellement au nombre de règles retournées par ECLAT et REREMI.

Dans le domaine des films, les règles extraites concernent les directeurs, acteurs et distributeurs. Dans le domaine des personnes, les règles traitent majoritairement des universités que ces personnes ont fréquenté et les prix qu'elles ont remporté. Cela est notamment dû au fait que les trois corpus de personnes concernent des scientifiques. Finalement, dans le domaine des objets , les définitions sont principalement à propos des constructeurs et des marques.

Comme vu dans l'expérience précédente, il y a bien plus de quasi-définitions que de redescriptions extraites. Cela est particulièrement flagrant dans le corpus `French_films`, où les redescriptions permettent de découvrir 6 définitions, contre 186 définitions pour les règles d'association.

Erreurs. Parmi les règles invalides, trois principaux types d'erreurs ont été rencontrés. Il y a tout d'abord les implications détectées comme des définitions (règle 26 par exemple). Dans ces cas, soit E_{10} soit E_{01} vaut 0, ce qui peut être utilisé comme aide à la détection de cette erreur. Il y a ensuite les redescriptions qui sont « trop approximatives », c'est-à-dire que la définition est globalement vraie, mais qu'il y a trop d'exceptions pour qu'elle soit considérée comme valide. Par exemple, dans la règle 44 du corpus `French_films`, la catégorie `Films_directed_by_Jean_Girault` est définie par l'attribut `dbo:starring dbr:Louis_de_Funès`. Or, bien que Girault et de Funès aient beaucoup d'œuvres cinématographiques en commun, cette définition n'est pas valide pour autant. Enfin, il y a les erreurs dues à un manque de données dans la base. Par exemple, la catégorie `German_drama_films`, est définie dans la règle 39 comme un film en langue allemande. Le fait que les films soient de genre dramatique est perdu parce que cette information n'est pas fournie dans les données. La règle 50 est un autre exemple, dans

TABLE 7.7 – Exemples de définitions obtenues par ECLAT et REREMI pour quelques corpus.

	Women_Mathematicians	
E	**Women_mathematicians** ≅ Person	R21
E	**Hungarian_mathematicians** ≅ Person ⊓ ∃.birthPlace{Budapest}	R22
E	**École_Normale_Supérieure_alumni** ≅ Agent ⊓ Person ⊓ Scientist ⊓ ∃.almaMater{École_Normale_Supérieure}	R23
R	**Whitehead_Prize_winners** ≅ ∃.award{Whitehead_Prize)}	R24
R	**University_of_Michigan_alumni** ∃.almaMater{University_of_Michigan}	R25
	Mathematicians	
E	**Hungarian_mathematicians** ≅ Person ⊓ ∃.birthPlace{Budapest}	R26
R	**Fields_Medalists** ≅ ∃.award{Fields_Medal)}	R27
R	**Fields_Medalists** ≅ Scientist ⊓ ∃.award{Fields_Medal}	R28
R	**Fields_Medalists** ≅ ∃.field{Mathematics} ⊓ ∃.award{Fields_Medal}	R29
	Hospital_films	
R	**Films_produced_by_Peter_Rogers** ≅ ∃.director{Gerald_Thomas}	R30
R	**Films_produced_by_Peter_Rogers** ≅ ∃.starring{Kenneth_Williams}	R31
R	**Films_directed_by_Gerald_Thomas** ≅ ∃.producer{Peter_Rogers}	R32
R	**Films_directed_by_Gerald_Thomas** ≅ ∃.starring{Kenneth_Williams}	R33
E	**British_comedy_films** ≅ ∃.starring{Donald_Sinden} ⊓ Film ⊓ Work	R34
E	**Films_produced_by_Peter_Rogers** ≅ ∃.director{Gerald_Thomas} ⊓ ∃.producer{Peter_Rogers} ⊓ ∃.starring{Kenneth_Williams} ⊓ Film ⊓ Work	R35
	Road_movies	
R	**Films_directed_by_Wim_Wenders** ≅ ∃.director{Wim_Wenders}	R36
R	**20th_Century_Fox_films** ≅ ∃.distributor{20th_Century_Fox}	R37
E	**Tamil-language_films** ≅ ∃.language{Tamil_language} ⊓ Film ⊓ Work	R38
E	**German_drama_films** ≅ ∃.language{German_language} ⊓ Film ⊓ Work	R39
E	**Walt_Disney_Pictures_films** ≅ ∃.distributor{Walt_Disney_Studios_Motion_Pictures} ⊓ ∃.studio{Walt_Disney_Pictures} ⊓ Film ⊓ Work	R40
	French_films	
R	**Films_directed_by_Georges_Méliès** ≅ ∃.studio{Star_Film_Company}	R41
R	**Films_directed_by_Georges_Méliès** ≅ ∃.director{Georges_Méliès} ⊓ ∃.language{Silent_film}	R42
E	**Films_directed_by_Georges_Méliès** ≅ ∃.studio{Star_Film_Company} ⊓ Film ⊓ Work	R43
E	**Films_directed_by_Jean_Girault** ≅ ∃.director{Jean_Girault} ⊓ ∃.starring{Louis_de_Funès} ⊓ ∃.writer{Jean_Girault} ⊓ Film ⊓ Work	R44
	Samsung_Galaxy	
E	**Phablets** ≅ ∃.manufacturer{Samsung_Electronics} ⊓ ∃.type{Phablet} ⊓ Device	R45
E	**Phablets** ≅ ∃.manufacturer{Samsung_Electronics} ⊓ ∃.operatingSystem{Android_(operating_system)} ⊓ ∃.series{Samsung_Galaxy_Note_series} ⊓ ∃.type{Phablet} ⊓ Device	R46
E	**Phablets** ≅ ∃.manufacturer{Samsung_Electronics} ⊓ ∃.operatingSystem{Android_(operating_system)} ⊓ ∃.form{Mobile_phone_form_factor} ⊓ ∃.series{Samsung_Galaxy_Note_series} ⊓ Device	R47
R	**Phablets** ≅ ∃.type{Phablet}	R48
	Smartphones	
R	**Samsung_mobile_phones** ≅ ∃.manufacturer{Samsung_Electronics}	R49
E	**Sony_mobile_phones** ≅ ∃.input{Capacitive_sensing} ⊓ ∃.input{Proximity_sensor} ⊓ ∃.input{Touchscreen} ⊓ ∃.type{Slate_phone} ⊓ ∃.type{Smartphone} ⊓ Device	R50

ce cas ; seuls les téléphones de la marque Sony ont des informations sur les éléments matériels qui les composent. En conséquence, le fait d'avoir un capteur de proximité par exemple, est associé à la catégorie `Sony_mobile_phones`.

Redondance. La figure 7.5 présente la proportion de règles valides parmi les règles retournées, ainsi que le nombre de catégories définies proportionnellement au nombre de règles extraites. On observe que les proportions sont similaires entre les redescriptions et les règles d'association. Cependant, les barres correspondant aux règles d'association ont une surface claire plus importante que les redescriptions, ce qui signifie que beaucoup de règles d'association définissent une même catégorie. Par exemple, dans le corpus `Samsung_Galaxy`, 7 règles d'association définissent la catégorie `Phablets` contre 2 redescriptions.

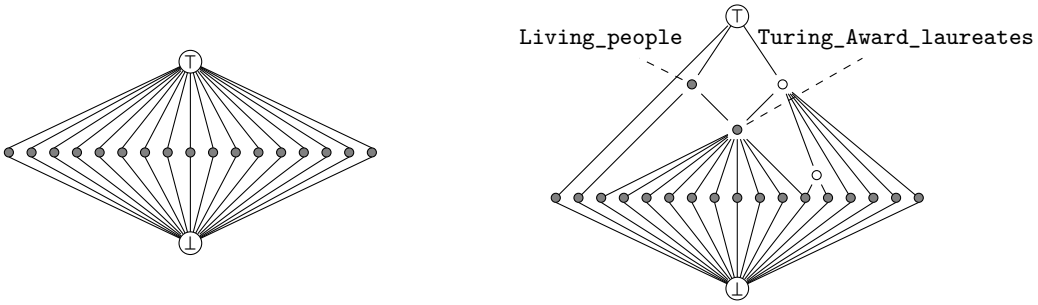


FIGURE 7.6 – Treillis de concepts construits à partir de définitions obtenues avec les redescriptions (à gauche) et avec les règles d'association (à droite) pour le jeu de données `Turing_Award_laureates`. Les noeuds en gris sont des concepts objets, c'est-à-dire qu'ils définissent une catégorie.

Vers une classification des catégories. Tandis que les redescriptions fournissent des définitions courtes et faciles à interpréter, les règles d'association ont l'intérêt de fournir des définitions qui permettent de construire une classification. À partir des règles extraites, un contexte $K = (G, M, I)$ est construit, où G est l'ensemble des catégories définies, M est l'ensemble des paires (prédicat, objet) impliquées dans au moins une définition, et I est l'ensemble des couples (catégorie, (prédicat,objet)) telles que la paire (prédicat,objet) intervient dans la définition de la catégorie. Le treillis de concepts associé est construit et celui du jeu de données `Turing_Award_laureates` est présenté en figure 7.6.

Nombre de règles extraites et diversité des jeux de données. La table 7.6 présente le nombre de définitions ainsi que le nombre de catégories extraites. Comparées aux règles d'association, les redescriptions sont 2 à 10 fois moins nombreuses. Contrairement à une hypothèse que nous avions formulée, il n'y a pas de corrélation entre le nombre de règles extraites et la densité du contexte. En revanche, les jeux de données `Smartphones` et `Sports_cars`, qui sont similaires, ont sensiblement le même nombre de règles. Cela pourrait représenter la « diversité » du jeu de données : plus il y a de catégories définies, plus le jeu de données est diversifié. Il faut cependant garder à l'esprit que les données utilisées ne tiennent pas compte des valeurs numériques présentes dans DBpedia, et que certaines catégories, telles que `5th-century_mathematicians` par exemple, ne peuvent pas être définies avec les données à disposition.

Le rôle de la confiance et du coefficient de Jaccard. La figure 7.7 indique la précision des définitions extraites en fonction de la confiance et du coefficient de Jaccard. Le fait que la précision augmente

en fonction du coefficient de Jaccard montre que cette mesure est adaptée à la fouille de redescriptions dans les LOD. En dehors du jeu de données `Road_movies`, le Jaccard retourne des résultats similaires à la confiance. La précision dépend du jeu de données et semble corrélée au nombre de redescriptions extraites. Cela expliquerait pourquoi les redescriptions ont une meilleure précision que les règles d'association, et cela reste cohérent avec la balance précision/rappel inhérente à ce genre de tâches.

La faible précision du jeu de données `Hospital_films` est difficile à expliquer par ses caractéristiques (figure 7.5). En revanche, si l'on regarde les triplets du jeu de données et les règles extraites, il semble que quelques instances sur-représentent l'ensemble de la catégorie.

7.3.3 Expérimentations 3 et 4 : Autour des redescriptions

Les deux dernières expériences s'intéressent au potentiel des redescriptions. Dans un premier temps, nous regardons l'expressivité des redescriptions, en permettant l'utilisation de disjonctions et de négations dans les règles. Dans un second temps, nous recherchons des catégories incompatibles, c'est-à-dire des catégories qui n'ont aucun objet en commun.

Expérimentation 3 : Vers des règles plus expressives

Dans cette expérimentation, nous nous intéressons à la définition des catégories en utilisant des conjonctions, des disjonctions et des négations. Pour cela, nous utilisons REREMI sur le jeu de données `Smartphones` selon deux modalités.

Première modalité : seules les disjonctions sont autorisées ;

Seconde modalité : les disjonctions et les négations sont autorisées.

S'agissant d'une démarche exploratoire, nous utilisons des seuils peu restrictifs : le coefficient de Jaccard minimum est fixé à 0.4 et la p-valeur à 0.2. Nous autorisons une seule catégorie du côté gauche de la règle, et jusqu'à 8 descripteurs du côté droit. La table 7.8 présente quelques exemples de définitions utilisant des conjonctions, des disjonctions et des négations. Les règles 51 à 55 sont obtenues dans la première modalité, les règles 56 à 60 sont obtenues dans la seconde modalité.

Dans la première modalité, nous obtenons 29 règles définissant 5 catégories. Parmi ces règles, 27 contiennent des disjonctions. Leur utilisation permet de « raffiner » certaines règles. Par exemple, la règle 51 raffine la règle 49 trouvée précédemment. Dans près de la moitié des cas, la disjonction est utilisée pour $\exists$`.manufacturer{Samsung` $\sqcup$ `Samsung_Electronics}`, comme le montrent les règles 51 à 54. La disjonction permet dans ce cas de considérer deux attributs synonymes. Dans les autres cas, la disjonction sert à énumérer de nombreux attributs qui qualifient des objets différents. Par exemple, la règle 56 montre une définition qui énumère les parties d'un ensemble : la catégorie `Android_(operating_system)_devices)` est constituée de tous les téléphones de marques utilisant l'OS Android (il est rare qu'une marque propose deux OS différents). Dans ce cas, la disjonction permet de représenter une partition. La table 7.9 indique la répartition des types de définitions par catégorie définie.

Dans la deuxième modalité, nous obtenons 76 règles, dont la répartition est représentée en figure 7.8. Cependant, certaines contraintes spécifiées dans le fichier de configuration ne sont pas prises en compte par l'interface de l'algorithme. Ainsi, nous observons des règles contenant plusieurs catégories et/ou ayant un Jaccard inférieur au seuil fixé. Une fois ces règles retirées, il reste 48 règles. Neuf d'entre elles ne comprennent pas de négation. Nous les laissons de côté puisqu'elles ont déjà été discutées dans les expériences précédentes. Parmi les 38 règles restantes, 31 ont la catégorie à la forme négative et 14 sont des « double négation » : tous les attributs de la partie gauche et de la partie droite sont mis à la forme négative. En effet, $\text{jacc}(A,B) \neq \text{jacc}(\neg A, \neg B)$ et dans certains cas, le Jaccard est plus élevé lorsque les attributs sont mis à la forme négative. C'est par exemple le cas de la règle 57 : son coefficient Jaccard

20. Dans cette règle, Smartphone vient de la paire (`dbp:type, Smartphone`) et non de (`rdf:type, Smartphone`).

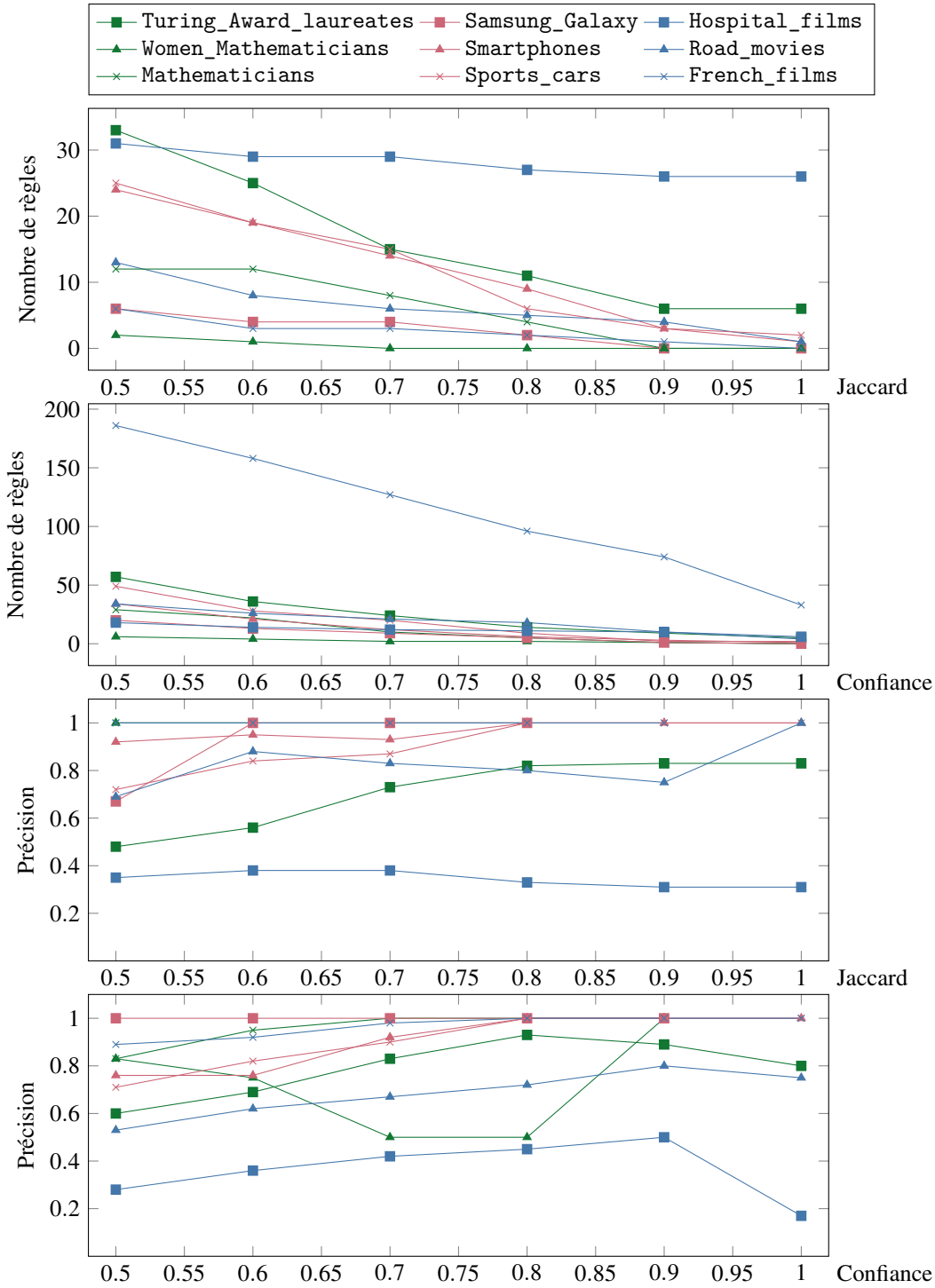


FIGURE 7.7 – Rappel et précision des règles (redescription ou quasi-définition) extraites en fonction du coefficient de Jaccard et de la confiance.

TABLE 7.8 – Exemples de règles découvertes contenant des disjonctions et/ou des négations.

.87	**Samsung_mobile_phones** ≅ ∃.manufacturer{Samsung} ⊔ ∃.manufacturer{Samsung_Electronics}	R51
.73	**Samsung_mobile_phones** ≅ (∃.operatingSystem{Android_(operating_system)} ⊓ ∃.manufacturer{Samsung_Electronics}) ⊔ ∃.manufacturer{Samsung}	R52
.80	**Samsung_Galaxy** ≅ (∃.manufacturer{Samsung} ⊔ ∃.manufacturer{Samsung_Electronics}) ⊓ ∃.operatingSystem{Android_(operating_system)}	R53
.64	**Samsung_Galaxy** ≅ ((Smartphone ⊓ ∃.manufacturer{Samsung_Electronics}) ⊔ ∃.manufacturer{Samsung}) ⊓ ∃.operatingSystem{Android_(operating_system)} [20]	R54
.66	**Android_(operating_system)_devices)** ≅ ∃.operatingSystem{Android_(operating_system)} ⊔ ∃.networks{GSM})	R55
.52	**Android_(operating_system)_devices)** ≅ ∃.manufacturer{Motorola_Mobility} ⊔ ∃.manufacturer{Xiaomi} ⊔ ∃.manufacturer{LG_Electronics} ⊔ ∃.manufacturer{Ninetology} ⊔ ∃.manufacturerSamsung_Electronics ⊔ ∃.networks{UMTS} ⊔ ∃.manufacturer{Sony_Mobile_Communications} ⊔ ∃.manufacturer{HTC}	R56
.96	¬ **Nokia_mobile_phones** ≅ ¬∃.manufacturer{Nokia}	R57
.74	**Android_(operating_system)_devices)** ≅ (∃.operatingSystem{Android_(operating_system)} ⊓ ¬ Software ⊓ Device)⊔ ∃.manufacturer{Meizu}	R58
.82	**Nokia_mobile_phones** ≅ (¬ ∃.operatingSystem{Android_(operating_system)} ⊓ ∃.manufacturer{Nokia} ⊓ ¬ Smartphone)⊔ ∃.series{Nokia_Lumia} [20]	R59
.82	**Nokia_mobile_phones** ≅ ¬ ∃.operatingSystem{Android_(operating_system)} ⊓ ∃.manufacturer{Nokia}	R60
.44	**Touchscreen_mobile_phones** ≅ ¬ Software ⊓ Device	

TABLE 7.9 – Répartition des disjonctions utilisées selon leur type et la catégorie définie.

Catégorie définie	Partition	Synonyme	Total
`GPS_navigation_devices`	1	0	1
`Touchscreen_mobile_phones`	6	0	6
`Android_(operating_system)_devices`	9	3	12
`Samsung_mobile_phones`	0	4	4
`Samsung_Galaxy`	0	4	4
Total	16	11	27

est de .96, alors qu'à l'affirmative (règle 12 table 7.4), il est de .82. Il reste 7 règles utilisant la négation uniquement sur des attributs de la description. Trois d'entre elles sont reportées dans la table 7.8

Expérimentation 4 : Catégories incompatibles

Dans cette expérience, on s'intéresse à la découverte de catégories « incompatibles », c'est-à-dire des paires de catégories (C_i, C_j) telles qu'aucun sujet s ne vérifie à la fois $\langle s, \texttt{dct:subject}, C_i \rangle$ et $\langle s, \texttt{dct:subject}, C_j \rangle$. Par exemple, on a `Nokia_Mobile_Phone` ⊓ `Turing_Award_laureate` ⊑ ⊥ ce qui signifie que ces deux catégories sont disjointes ou incompatibles. Quelques exemples de catégories incompatibles obtenues avec REREMI sont présentées dans la table 7.10.

Concernant le jeu de données `Turing_Award_laureates`, les résultats sont décevants. En effet, la plupart des incompatibilités par REREMI n'en sont pas. L'une des hypothèses expliquant cette observation est liée au domaine du jeu de données, à savoir les personnes. Dans les jeux de données décrivant des personnes, les catégories sont utilisées pour les caractériser selon des aspects variés de leur vie (personnelle, professionnelle, publique, ...). Ainsi, beaucoup de catégories sont détectées comme incompatibles non pas parce qu'elles le sont effectivement, mais parce qu'elles concernent deux aspects différents de la

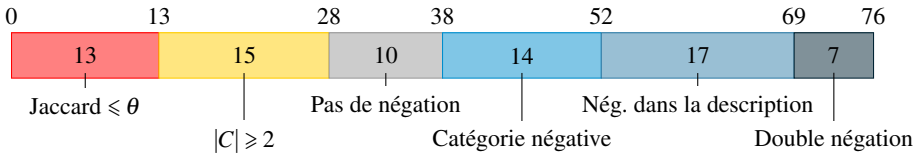


FIGURE 7.8 – Répartition des règles lorsque REREMI utilise la forme négative. La moitié d'entre elles ne sont pas pertinentes, soit parce que l'algorithme n'aurait pas dû les retrouver (en rouge et orange), soit parce qu'elles n'ont pas de négation (en gris).

TABLE 7.10 – Incompatibilités découvertes par REREMI. Toutes les règles avaient un Jaccard à 0.

N.	Catégories incompatibles
	`Turing_Award_laureates`
R61	Harvard_University_alumni ⊓ Scientist_from_California ⊑ ⊥
R62	Harvard_University_alumni ⊓ Jewish_American_scientists ⊑ ⊥
R63	Theoretical_computer_scientists ⊓ IBM_Fellows ⊑ ⊥
	`Sports_cars`
R64	1970s_automobiles ⊓ Cars_introduced_in_1998 ⊑ ⊥
R65	Kit_cars ⊓ Coupes ⊑ ⊥
R66	1960s_automobiles ⊓ Lotus_racing_cars ⊑ ⊥
	`Smartphones`
R67	Blackberry ⊓ Nokia_mobile_phones ⊑ ⊥
R68	Mobile_phones_introduced_in_2013 ⊓ Mobile_phones_introduced_in_2014 ⊑ ⊥
R69	Touchscreen_mobile_phone ⊓ Nokia_platforms ⊑ ⊥
	`French_films`
R70	1980s_drama_films ⊓ 1970s_comedy_films ⊑ ⊥

vie d'une personne et qu'il n'y a pas d'individu dans le jeu de données qui présente simultanément ces deux caractéristiques. C'est par exemple le cas de la règle 62.

Les autres jeux de données présentent de meilleurs résultats. Dans le jeu de données `Sports_cars`, une grande partie des catégories sont incompatibles car elles caractérisent des voitures de deux époques différentes, comme le montre la règle 64 par exemple. Pour les smartphones, la plupart des incompatibilités trouvées correspondent à des marques différentes, comme par exemple dans la règle 67. Beaucoup de catégories incompatibles sont découvertes dans ces deux jeux de données.

Dans le jeu de données `French_films`, en revanche, une seule incompatibilité est trouvée par REREMI. Cela peut s'expliquer par la faible densité de ce jeu de données. En effet, si deux catégories ont un faible support, il y a de fortes chances qu'aucune ressource ne soit caractérisée par ces deux catégories en même temps. Ainsi, la p-valeur de la règle décrivant ces deux catégories comme incompatibles est élevée.

7.4 Conclusion

Les trois expérimentations menées ici nous permettent de mieux cerner les intérêts et les limites des différentes approches envisagées pour la tâche de découverte de définitions. La première expérience, en permettant de définir plusieurs catégories en une seule règle, rend difficile l'interprétation des résultats et la comparaison des algorithmes. La deuxième expérience, en n'acceptant qu'une catégorie par règle, permet d'étendre la comparaison entre les règles d'association et les redescriptions. Enfin, la dernière expérimentation permet de discuter les spécificités des redescriptions.

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.

La table 7.11 présente le nombre de prédicats utilisés, toutes règles confondues, par jeu de données et par algorithme, dans la deuxième expérimentation. On remarque qu'au sein d'un jeu de données, très peu de prédicats différents sont utilisés. Il semble donc que le contexte fourni aux algorithmes pourrait être significativement réduit en ne tenant compte que des tops-k prédicats, la méthode pour trouver k restant à déterminer. Ce constat va dans le sens des choix effectués par Alam et al. (2015), qui n'avaient considéré que quelques prédicats pour chacun des jeux de données.

TABLE 7.11 – Nombre de prédicats impliqués dans les définitions extraites par REREMI et ECLAT.

	Prédicats	REREMI	ECLAT
`Turing_Award_laureates`	35	4	5
`Women_Mathematicians`	98	2	3
`Mathematicians`	202	4	5
`Samsung_Galaxy`	33	2	7
`Smartphones`	98	5	8
`Sports_cars`	61	3	5
`Hospital_films`	46	5	2
`Road_movies`	103	3	7
`French_films`	111	4	10

Dans le cas de nos expérimentations, 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.

Synthèse du chapitre

Dans ce chapitre, nous avons proposé des expérimentations de découverte de définitions sur des données issues de DBpedia. Nous avons eu recours à des jeux de données variés tant au niveau de la taille que du domaine. Trois approches ont été utilisées : les règles d'association, les redescriptions et les règles de traduction. Nous avons comparé quantitativement et qualitativement ces approches.

Il en ressort que les règles d'association permettre d'extraire un très grand nombre de définitions, mais présentent de nombreuses redondances. Au contraire, les règles de traduction offrent un faible nombre de définition, avec une très bonne couverture des données. Cependant, les définitions extraites comportent de nombreux attributs. Finalement, les redescriptions sont un compromis en terme de nombre de définitions extraites : leur nombre est bien moins grand que les redescriptions, et elles ont peu d'attributs. Cependant, leur couverture est plus faible que les règles de traduction.

Chapitre 8

Conclusion et perspectives

8.1 Synthèse

Cette thèse s'inscrit dans les domaines de la découverte de connaissances et du web des données. Dans un premier temps, nous avons proposé une *pattern structure* pour appliquer des méthodes de FCA sur des ensembles de triplets, puis nous nos sommes intéressés à trois approches pour découvrir des définitions dans les LOD. Enfin, nous avons mené des expérimentations pour évaluer qualitativement et quantitativement ces trois approches. Nous résumons ces trois contributions dans cette section.

Dans la continuité du travail de Alam et al. (2015), nous avons proposé une *pattern structure* pour le web des données. Cette *pattern structure* permet de structurer les triplets RDF en tenant compte de connaissances du domaine. Notamment, elle permet de tenir compte des deux ordres partiels induits par `rdf:subClassOf` et `rdf:subPropertyOf`. Enfin, les données peuvent être organisées au sein d'un treillis de concepts, facilitant la visualisation et l'exploration de données RDF. Bien que sa pertinence soit mise à mal par une partie des LOD n'utilisant que très peu les taxonomies de prédicats, nous avons montré que cette *pattern structure* est fonctionnelle sur des données RDF.

Nous nous sommes intéressés à trois algorithmes pour découvrir des définitions de classes dans le web des données. Le premier, ECLATZ, extrait des règles d'association. Un post-traitement nous permet d'obtenir des définitions. REREMI, le deuxième, recherche des redescriptions, qui correspondent directement à un ensemble de définitions. Enfin, TRANSLATOR recherche à la fois des associations unidirectionnelles telles que les règles d'association et des associations bidirectionnelles telles que les redescriptions. Cet algorithme s'appuie sur des principes de théorie de l'information, en particulier MDL. Nous avons discuté les spécificités de chaque algorithme, et mis en avant les liens forts existant entre les règles d'association et les redescriptions. Pour chacune des approches, nous avons proposé une méthodologie permettant de l'appliquer au web des données.

Enfin, nous avons proposé une étude expérimentale sur des données de DBpedia, pour vérifier que les résultats reflètent les spécificités des approches. Ainsi, 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'analyste. De nos expérimentations, il ressort qu'ECLATZ est l'algorithme qui définit le plus de catégories, au dépend du grand nombre de règles d'association extraites. TRANSLATOR extrait significativement moins de règles, mais, en conséquence, définit moins de catégories. REREMI, malgré un faible nombre de catégories définies, offre des définitions plus faciles à interpréter qui n'incluent que le minimum d'attributs.

8.2 Perspectives

Le travail présenté dans cette thèse ouvre plusieurs pistes de recherche intéressantes, tant sur le plan de la découverte de connaissances que sur le plan du web des données.

Nous avons proposé une *pattern structure* permettant de tenir compte de deux ordres partiels qui s'appliquent sur les prédicats et sur les objets de triplets. Il serait intéressant de considérer deux ordres partiels sur la même ressource, par exemple sur l'objet. Cette affirmation repose sur l'intuition que certaines relations ont les propriétés requises pour former un ordre partiel et que leur utilisation comme connaissances du domaine enrichirait les définitions. Pour motiver cette affirmation, considérons par exemple le prédicat `situéEn`, qui définit une relation réflexive, transitive et antisymétrique. Actuellement, à partir de l'affirmation *Nancy est située en France*, nous pouvons pas déduire *Nancy est située dans un pays*, mais nous ne pouvons pas déduire *Nancy est situé en Europe*. Or, cette information est pertinente dès lors que l'on traite des données incluant des informations géographiques.

Dans cette thèse, nous nous sommes intéressés uniquement aux ressources identifiées par une URI, représentées sous la forme de jeux de données Booléens. Il serait intéressant de prendre en compte un vocabulaire plus riche, tant du côté des données que du schéma. Les données contiennent en effet par exemple des valeurs numériques (poids, distances, etc.) ainsi que des dates qu'il serait utile de prendre en compte. Les *interval pattern structures* (Mehdi Kaytoue et al., 2011) existent déjà pour traiter des valeurs numériques, mais des difficultés persistent. En effet, ces *pattern structures* s'appuient des intervalles pour les valeurs numériques. Afin d'éviter de trop augmenter la complexité des *pattern structures*, il est nécessaire de fixer un seuil sur les intervalles, mais ce seuil est difficile à déterminer pour des données RDF. Par exemple, si on considère des intervalles sur des poids, on n'aura pas les mêmes intervalles selon que l'on s'intéresse à des téléphones ou à des mammifères.

Actuellement, nous ne considérons que des triplets, ce qui ne permet pas d'extraire des définitions complexes telles que *Un auteur de romans policiers est un auteur qui écrit des livres qui sont des romans policiers*. Cela pourrait être fait avec des approches telles que l'analyse relationnelle de concepts (Rouane Hacene et al., 2013), ou en fouillant des redescriptions non plus sur des triplets mais sur des graphes, comme initié par Galbrun et Kimmig (2014).

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 logiques, 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.

Dans cette thèse, nous avons comparé les différentes approches avec leurs mesures usuelles, à savoir la confiance et le coefficient de Jaccard. Cependant, de nombreuses autres mesures existent. Cela mérite de s'intéresser à l'impact du choix de la mesure sur les résultats obtenus. De nombreux travaux existent déjà dans le domaine des règles d'association, tels que celui de P. Tan, V. Kumar et Srivastava (2002). Du côté des redescriptions, bien que le coefficient de Jaccard soit motivé par Ramakrishnan, D. Kumar et al. (2004) et Galbrun et Miettinen (2017), il n'existe pas, à notre connaissance, d'étude expérimentale sur le choix des mesures (coefficient de Jaccard et p-value).

Les expérimentations que nous avons menées ne tirent pas pleinement parti des redescriptions. Notamment, en ne considérant qu'une catégorie dans nos définitions, nous réduisons l'approche à un simple problème de classification. Mais la possibilité qu'offrent les redescriptions de considérer des formules logiques de chaque côté de la redescription est un atout dans le web des données, tout particulièrement pour des questions d'alignement de bases de connaissances.

Les clés de liage sont des ensembles de paires de propriétés qui permettent d'identifier les mêmes individus au sein de plusieurs jeux de données (Atencia, Chein et al., 2014). Des approches pour découvrir les clés de liage utilisant l'analyse relationnelle de concepts (Atencia, David et al., 2019) et les *pattern structures* (Abbas, David et Napoli, 2019) existent déjà, et les redescriptions pourraient apporter un complément intéressant à ces travaux.

Enfin, les redescriptions pourraient s'avérer utiles pour la recherche de correspondances complexes dans les bases de connaissances. Ces correspondances s'appuient parfois sur des *competency questions* qui représentent des questions auxquelles une base de connaissances doit pouvoir répondre. Les *competency questions* peuvent s'exprimer sous la forme de requêtes SPARQL. Thiéblin (2018) propose de les utiliser pour faire de l'alignement complexe de bases de connaissances. Un alignement extrait pourrait par exemple être $o_1 : AcceptedPaper \equiv \exists o_2 : hasDecision.o_2 : accept$, où o_1 et o_2 désignent deux bases de connaissances différentes. La fouille de redescriptions permettrait donc de découvrir ces alignements, chaque requête correspondant à une *competency question*.

Liste des publications

Publications internationales

Conférences

J. Reynaud, Y. Toussaint, and A. Napoli. Using redescriptions and formal concept analysis for mining definitions in linked data. In *Formal Concept Analysis - 15th International Conference, ICFCA 2019, Frankfurt, Germany, June 25-28, 2019, Proceedings*, pages 241–256, 2019.

J. Reynaud, Y. Toussaint, and A. Napoli. Redescription mining for learning definitions and disjointness axioms in linked open data. In *Graph-Based Representation and Reasoning - 24th International Conference on Conceptual Structures, ICCS 2019, Marburg, Germany, July 1-4, 2019, Proceedings*, pages 175–189, 2019.

J. Reynaud, M. Alam, Y. Toussaint, and A. Napoli. A proposal for classifying the content of the web of data based on FCA and pattern structures. In *Foundations of Intelligent Systems - 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings*, pages 684–694, 2017.

Q. Brabant, M. Couceiro, A. Napoli, and J. Reynaud. From meaningful orderings in the web of data to multi-level pattern structures. In *Foundations of Intelligent Systems - 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings*, pages 622–631, 2017.

Workshops

J. Reynaud, Y. Toussaint, and A. Napoli. Three approaches for mining definitions from relational data in the web of data. In *Proc. of the 6th International Workshop "What can FCA do for Artificial Intelligence"? co-located with International Joint Conference on Artificial Intelligence and European Conference on Artificial Intelligence (IJCAI/ECAI 2018), Stockholm, Sweden, July 13, 2018.*, pages 21–32, 2018.

J. Reynaud, Y. Toussaint, and A. Napoli. Contribution to the classification of web of data based on formal concept analysis. In *Proceedings of the 5th International Workshop "What can FCA do for Artificial Intelligence"? co-located with the European Conference on Artificial Intelligence, FCA4AI@ECAI 2016, The Hague, the Netherlands, August 30, 2016.*, pages 69–78, 2016.

Publications nationales

J. Reynaud, Y. Toussaint, and A. Napoli. Trois approches pour classifier les données du web des données. In *Actes de la Conférence Nationale d'Intelligence Artificielle et Rencontres des Jeunes Chercheurs en Intelligence Artificielle (CNIA+RJCIA 2018), Nancy, France, 4-6 Juillet 2018.*, pages 94–101, July 2018.

J. Reynaud, E. Galbrun, M. Alam, Y. Toussaint, and A. Napoli. Définir les catégories de DBpedia avec des règles d'associations et des redescriptions. In *18ème Conférence Extraction et Gestion de Connaissances (EGC 2018)*, January 2018.

Bibliographie

[ADN19] Nacira ABBAS, Jérôme DAVID et Amedeo NAPOLI. "Linkex : A Tool for Link Key Discovery Based on Pattern Structures". In : *Supplementary Proceedings of ICFCA 2019 Conference and Workshops, Frankfurt, Germany, June 25-28, 2019*. 2019, p. 33–38. URL : http://ceur-ws.org/Vol-2378/shortAT1.pdf.

[AIS93] Rakesh AGRAWAL, Tomasz IMIELIŃSKI et Arun SWAMI. "Mining Association Rules Between Sets of Items in Large Databases". In : *ACM SIGMOD Rec.* T. 22. 2. ACM, 1993, p. 207–216.

[AS94] Rakesh AGRAWAL et Ramakrishnan SRIKANT. "Fast Algorithms for Mining Association Rules in Large Databases". In : *VLDB'94*. 1994, p. 487–499.

[Ala+15] Mehwish ALAM et al. "Bridging DBpedia Categories and DL-Concept Definitions Using Formal Concept Analysis". In : *Proceedings of the 4th International Workshop "What can FCA do for Artificial Intelligence ?", FCA4AI 2015, co-located with the International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25, 2015*. 2015, p. 9–16. URL : http://ceur-ws.org/Vol-1430/paper1.pdf.

[Aly+18] El Arby Sidi ALY et al. "Découverte de cardinalité maximale contextuelle dans les bases de connaissances(Mining contextual maximum cardinality in knowledge bases)". In : *Actes de la Conférence Nationale d'Intelligence Artificielle et Rencontres des Jeunes Chercheurs en Intelligence Artificielle (CNIA+RJCIA 2018), Nancy, France, 4-6 Juillet 2018*. 2018, p. 86–93. URL : http://ceur-ws.org/Vol-2133/cnia-paper6.pdf.

[AP11] Simon ANDREWS et Simon POLOVINA. "A Mapping from Conceptual Graphs to Formal Concept Analysis". In : *Conceptual Structures for Discovering Knowledge - 19th International Conference on Conceptual Structures, ICCS 2011, Derby, UK, July 25-29, 2011. Proceedings*. 2011, p. 63–76. DOI : 10.1007/978-3-642-22688-5_5. URL : https://doi.org/10.1007/978-3-642-22688-5%5C_5.

[AGS06] Mark van ASSEM, Aldo GANGEMI et Guus SCHREIBER. *RDF/OWL Representation of WordNet*. W3C Working Draft. http ://www.w3.org/TR/2006/WD-wordnet-rdf-20060619/. W3C, juin 2006.

[Ate+14] Manuel ATENCIA, Michel CHEIN et al. "Defining Key Semantics for the RDF Datasets : Experiments and Evaluations". In : *Graph-Based Representation and Reasoning - 21st International Conference on Conceptual Structures, ICCS 2014, Iaşi, Romania, July 27-30, 2014, Proceedings*. 2014, p. 65–78. DOI : 10.1007/978-3-319-08389-6_7. URL : https://doi.org/10.1007/978-3-319-08389-6%5C_7.

[Ate+19] Manuel ATENCIA, Jérôme DAVID et al. "Link key candidate extraction with relational concept analysis". In : *Discrete Applied Mathematics* (2019), p. 1–19. DOI : 10.1016/j.dam.2019.02.012. URL : https://hal.archives-ouvertes.fr/hal-02196757.

[Aue+07] Sören AUER et al. “DBpedia : A Nucleus for a Web of Open Data”. In : *The Semantic Web*. Sous la dir. de Karl ABERER et al. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 722–735. ISBN : 978-3-540-76298-0.

[BHS09] Franz BAADER, Ian HORROCKS et Ulrike SATTLER. “Description Logics”. In : *Handbook on Ontologies*. 2009, p. 21–43. URL : https://doi.org/10.1007/978-3-540-92673-3%5C_1.

[BM70] Marc BARBUT et Bernard MONJARDET. “Ordre et classification, vols. 1 and 2”. In : *Hachette, Paris, France* (1970).

[Bas+00] Yves BASTIDE et al. “Mining Minimal Non-redundant Association Rules Using Frequent Closed Itemsets”. In : *Proc. of the 1st International Conference on Computational Logic*. Springer, 2000, p. 972–986.

[Ber09] Tim BERNERS-LEE. *Linked Data - Design Issues*. [En ligne ; accédé le 22. Jul. 2019]. Juin 2009. URL : https://www.w3.org/DesignIssues/LinkedData.html.

[BH01] Tim BERNERS-LEE et James HENDLER. “Publishing on the semantic web”. In : *Nature* 410.6832 (avr. 2001), p. 1023–1024. ISSN : 1476-4687. DOI : 10.1038/35074206.

[Bir40] Garrett BIRKHOFF. *Lattice theory*. T. 25. American Mathematical Soc., 1940.

[Bre+84] Leo BREIMAN et al. *Classification and Regression Trees*. Wadsworth, 1984. ISBN : 0-534-98053-8.

[BG14] Dan BRICKLEY et Ramanathan GUHA. *RDF Schema 1.1*. W3C Recommendation. W3C, fév. 2014. URL : http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[BMS97] Sergey BRIN, Rajeev MOTWANI et Craig SILVERSTEIN. “Beyond Market Baskets : Generalizing Association Rules to Correlations”. In : *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’97. event-place : Tucson, Arizona, USA. New York, NY, USA : ACM, 1997, p. 265–276. ISBN : 9780897919111. DOI : 10.1145/253260.253327. URL : http://doi.acm.org/10.1145/253260.253327.

[Bri+97] Sergey BRIN, Rajeev MOTWANI, Jeffrey D. ULLMAN et al. “Dynamic Itemset Counting and Implication Rules for Market Basket Data”. In : *SIGMOD Rec.* 26.2 (juin 1997), p. 255–264.

[CR04] Claudio CARPINETO et Giovanni ROMANO. *Concept Data Analysis : Theory and Applications*. Chichester, UK : John Wiley & Sons, 2004.

[CK04] Jeremy CARROLL et Graham KLYNE. *Resource Description Framework (RDF) : Concepts and Abstract Syntax*. W3C Recommendation. W3C, fév. 2004. URL : http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[Cer+08] Loic CERF et al. “Data Peeler : Contraint-Based Closed Pattern Mining in n-ary Relations”. In : *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*. 2008, p. 37–48. DOI : 10.1137/1.9781611972788.4. URL : https://doi.org/10.1137/1.9781611972788.4.

[CYH14] Hong CHENG, Xifeng YAN et Jiawei HAN. “Mining Graph Patterns”. In : *Frequent Pattern Mining*. 2014, p. 307–338. DOI : 10.1007/978-3-319-07821-2_13. URL : https://doi.org/10.1007/978-3-319-07821-2%5C_13.

[CN14] Vıctor CODOCEDO et Amedeo NAPOLI. “A Proposition for Combining Pattern Structures and Relational Concept Analysis”. In : *Formal Concept Analysis - 12th International Conference, ICFCA 2014, Cluj-Napoca, Romania, June 10-13, 2014. Proceedings*. 2014, p. 96–111. DOI : 10.1007/978-3-319-07248-7_8. URL : https://doi.org/10.1007/978-3-319-07248-7%5C_8.

[DP02] Brian A. DAVEY et Hilary A. PRIESTLEY. *Introduction to Lattices and Order (2. ed.)* Cambridge University Press, 2002.

[DS04] Mike DEAN et Guus SCHREIBER. *OWL Web Ontology Language Reference*. W3C Recommendation. W3C, fév. 2004. URL : http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[Dis11] Felix DISTEL. "Learning description logic knowledge bases from data using methods from formal concept analysis". Thèse de doct. Dresden University of Technology, 2011. URL : http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-70199.

[ER97] Leo EGGHE et Ronald ROUSSEAU. "Duality in information retrieval and the hypergeometric distribution". In : *Journal of Documentation* 53.5 (1997), p. 488–496.

[FdE10] Nicola FANIZZI, Claudia D'AMATO et Floriana ESPOSITO. "Induction of Concepts in Web Ontologies through Terminological Decision Trees". In : *Machine Learning and Knowledge Discovery in Databases*. Sous la dir. de José Luis BALCÁZAR et al. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, p. 442–457.

[FPS96] Usama M. FAYYAD, Gregory PIATETSKY-SHAPIRO et Padhraic SMYTH. "From Data Mining to Knowledge Discovery : An Overview". In : *Advances in Knowledge Discovery and Data Mining*. 1996, p. 1–34.

[Fer15] Sébastien FERRÉ. "A Proposal for Extending Formal Concept Analysis to Knowledge Graphs". In : *Formal Concept Analysis - 13th International Conference, ICFCA 2015, Nerja, Spain, June 23-26, 2015, Proceedings*. 2015, p. 271–286. DOI : 10.1007/978-3-319-19545-2_17. URL : https://doi.org/10.1007/978-3-319-19545-2%5C_17.

[FR00] Sébastien FERRÉ et Olivier RIDOUX. "A Logical Generalization of Formal Concept Analysis". In : *Conceptual Structures : Logical, Linguistic, and Computational Issues*. Sous la dir. de Bernhard GANTER et Guy W. MINEAU. Berlin, Heidelberg : Springer Berlin Heidelberg, 2000, p. 371–384.

[Flo10] Luciano FLORIDI. *Information : A Very Short Introduction*. Oxford : Oxford University Press, 2010, p. 152. URL : ttps://www.veryshortintroductions.com/10.1093/actrade/9780199551378.001.0001/actrade-9780199551378.

[Gal+13] Luis Antonio GALÁRRAGA et al. "AMIE : association rule mining under incomplete evidence in ontological knowledge bases". In : *WWW'13*. 2013, p. 413–422.

[Gal+15] Luis Antonio GALÁRRAGA et al. "Fast rule mining in ontological knowledge bases with AMIE+". In : *VLDB Journal* 24.6 (2015), p. 707–730.

[GK14] Esther GALBRUN et Angelika KIMMIG. "Finding relational redescriptions". In : *Machine Learning* 96.3 (2014), p. 225–248. DOI : 10.1007/s10994-013-5402-3. URL : https://doi.org/10.1007/s10994-013-5402-3.

[GM12a] Esther GALBRUN et Pauli MIETTINEN. "Siren : an interactive tool for mining and visualizing geospatial redescriptions". In : *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. 2012, p. 1544–1547. DOI : 10.1145/2339530.2339776. URL : https://doi.org/10.1145/2339530.2339776.

[GM12b] Esther GALBRUN et Pauli MIETTINEN. "From Black and White to Full Color : Extending Redescription Mining Outside the Boolean World". In : *Statistical Analysis and Data Mining* 5.4 (2012), p. 284–303.

[GM17] Esther GALBRUN et Pauli MIETTINEN. *Redescription Mining*. Springer Briefs in Computer Science. Springer, 2017. ISBN : 978-3-319-72888-9. DOI : 10.1007/978-3-319-72889-6. URL : https://doi.org/10.1007/978-3-319-72889-6.

[Gan+06] Aldo GANGEMI et al. "Modelling Ontology Evaluation and Validation". In : *The Semantic Web : Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*. 2006, p. 140–154. DOI : 10.1007/11762256_13. URL : https://doi.org/10.1007/11762256%5C_13.

[GK01] Bernhard GANTER et Sergei O. KUZNETSOV. "Pattern Structures and Their Projections". In : *Conceptual Structures : Broadening the Base, 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, July 30-August 3, 2001, Proceedings*. 2001, p. 129–142.

[GW99] Bernhard GANTER et Rudolf WILLE. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999. ISBN : 978-3-540-62771-5. DOI : 10.1007/978-3-642-59830-2. URL : https://doi.org/10.1007/978-3-642-59830-2.

[GFC04] Asunción GÓMEZ-PÉREZ, Mariano FERNÁNDEZ-LÓPEZ et Óscar CORCHO. *Ontological Engineering : With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web*. Advanced Information and Knowledge Processing. Springer, 2004. ISBN : 978-1-85233-551-9. DOI : 10.1007/b97353. URL : https://doi.org/10.1007/b97353.

[Gru93] Thomas R. GRUBER. "A translation approach to portable ontology specifications". In : *Knowledge Acquisition* 5.2 (1993), p. 199–220. ISSN : 1042-8143. DOI : https://doi.org/10.1006/knac.1993.1008. URL : http://www.sciencedirect.com/science/article/pii/S1042814383710083.

[Grü04] Peter GRÜNWALD. "A tutorial introduction to the minimum description length principle". In : *CoRR* math.ST/0406077 (2004).

[GD86] Jean-Louis GUIGUES et Vincent DUQUENNE. "Familles minimales d'implications informatives résultant d'un tableau de données binaires". In : *Mathématiques et Sciences humaines* 95 (1986), p. 5–18.

[GH07] Fabrice GUILLET et Howard J. HAMILTON. *Quality Measures in Data Mining (Studies in Computational Intelligence)*. Springer-Verlag, fév. 2007. ISBN : 978-354044911-9. URL : http://dl.acm.org/citation.cfm?id=1207196.

[HS13] Steven HARRIS et Andy SEABORNE. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C, mar. 2013. URL : http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[Hor05] Ian HORROCKS. "OWL : A Description Logic Based Ontology Language". In : *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*. 2005, p. 5–8. DOI : 10.1007/11564751_2. URL : https://doi.org/10.1007/11564751%5C_2.

[Hor+12] Ian HORROCKS et al. *OWL 2 Web Ontology Language Profiles (Second Edition)*. W3C Recommendation. W3C, déc. 2012. URL : http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/.

[Hos+12] M. Shahriar HOSSAIN et al. "Connecting the Dots between PubMed Abstracts". In : *PLOS ONE* 7.1 (jan. 2012), p. 1–23. DOI : 10.1371/journal.pone.0029509. URL : https://doi.org/10.1371/journal.pone.0029509.

[Hot+06] Andreas HOTHO et al. “BibSonomy : a social bookmark and publication sharing system”. In : *Proceedings of the First Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*. 2006.

[IBM18] IBM. *IBM Knowledge Center - Lift in an association rule*. [Online ; accessed 1. Mar. 2018]. Mar. 2018. URL : https://www.ibm.com/support/knowledgecenter/en/SSEPGG_9.5.0/com.ibm.im.model.doc/c_lift_in_an_association_rule.html.

[Jäs+06] Robert JÄSCHKE et al. “TRIAS - An Algorithm for Mining Iceberg Tri-Lattices”. In : *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*. 2006, p. 907–911. DOI : 10.1109/ICDM.2006.162. URL : https://doi.org/10.1109/ICDM.2006.162.

[Kay97] Daniel KAYSER. *La représentation des connaissances*. Hermes Paris, 1997.

[Kay+11] Mehdi KAYTOUE et al. “Mining gene expression data with pattern structures in formal concept analysis”. In : *Information Sciences* 181.10 (2011), p. 1989–2001. URL : https://hal.archives-ouvertes.fr/hal-00541100.

[Kay+10] M. KAYTOUE et al. “The Coron System”. In : *8th International Conference on Formal Concept Analsis (ICFCA) - Supplementary Proceedings*. Sous la dir. de L. BOUMEDJOUT et al. (demo paper). 2010, p. 55–58. URL : http://www.loria.fr/~kaytouem/publi/ICFCA10-demo-coron.pdf.

[Kle+94] Mika KLEMETTINEN et al. “Finding Interesting Rules from Large Sets of Discovered Association Rules”. In : *CIKM’94*. 1994, p. 401–407.

[KM02] Marja-Riitta KOIVUNEN et Eric MILLER. “W3C Semantic Web Activity”. In : *Semantic Web Kick-Off in Finland - Vision, Technologies, Research, and Applications*. Sous la dir. d’Eero HYVÖNEN. Helsinki Institute for Information Technology (HIIT) Publications, 2002, p. 27–44. URL : https://www.w3.org/2001/12/semweb-fin/w3csw.

[Köt15] Jens KÖTTERS. “Concept Lattices of RDF Graphs”. In : *Proceedings of the International Workshop on Formal Concept Analysis and Applications, FCA&A 2015, co-located with 13th International Conference on Formal Concept Analysis (ICFCA 2015), Nerja, Málaga, Spain, June 23-26, 2015*. 2015, p. 81–91. URL : http://ceur-ws.org/Vol-1434/paper6.pdf.

[Kum+08] Deept KUMAR et al. “Algorithms for Storytelling”. In : *IEEE Trans. Knowl. Data Eng.* 20.6 (2008), p. 736–751. DOI : 10.1109/TKDE.2008.32. URL : https://doi.org/10.1109/TKDE.2008.32.

[LS18] Jonathan LAJUS et Fabian M. SUCHANEK. “Are All People Married ? Determining Obligatory Attributes in Knowledge Bases”. In : *International Conference WWW*. Lyon, France, 2018.

[LG15] Matthijs van LEEUWEN et Esther GALBRUN. “Association Discovery in Two-View Data”. In : *TKDE* 27.12 (2015), p. 3190–3202.

[LW95] Fritz LEHMANN et Rudolf WILLE. “A Triadic Approach to Formal Concept Analysis”. In : *Conceptual Structures : Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS ’95, Santa Cruz, California, USA, August 14-18, 1995, Proceedings*. 1995, p. 32–43. DOI : 10.1007/3-540-60161-9_27. URL : https://doi.org/10.1007/3-540-60161-9%5C_27.

[Lux91] Michael LUXENBURGER. “Implications partielles dans un contexte”. In : *Mathématiques, informatique et sciences humaines* 29.113 (1991), p. 35–55.

[McC80] John MCCARTHY. "Circumscription - A Form of Non-Monotonic Reasoning". In : *Artif. Intell.* 13.1-2 (1980), p. 27–39. DOI : 10.1016/0004-3702(80)90011-9. URL : https://doi.org/10.1016/0004-3702(80)90011-9.

[MDŠ18] Matej MIHELČIĆ, Sašo DŽEROSKI et Tomislav ŠMUC. "Extending Redescription Mining to Multiple Views". In : *Discovery Science - 21st International Conference, DS 2018, Limassol, Cyprus, October 29-31, 2018, Proceedings*. 2018, p. 292–307. DOI : 10.1007/978-3-030-01771-2_19. URL : https://doi.org/10.1007/978-3-030-01771-2%5C_19.

[Mil98] George A MILLER. *WordNet : An electronic lexical database*. MIT press, 1998. Site associé : https://wordnet.princeton.edu/.

[Nij06] Siegfried NIJSSEN. "Mining Structured Data". Thèse de doct. Leiden Institute of Advanced Computer Science, 2006.

[Pia91] Gregory PIATETSKY-SHAPIRO. "Discovery, Analysis, and Presentation of Strong Rules". In : *Knowledge Discovery in Databases*. 1991.

[Ram+04] Naren RAMAKRISHNAN, Deept KUMAR et al. "Turning CARTwheels : an Alternating Algorithm for Mining Redescriptions". In : *KDD'04*. 2004, p. 266–275.

[RZ09] Naren RAMAKRISHNAN et Mohammed Javeed ZAKI. "Redescription Mining and Applications in Bioinformatics". In : sous la dir. de Jake Y CHEN et Stefano LONARDI. Chapman & Hall/CRC, 2009. DOI : 10.1201/9781420086850.ch22. URL : http://www.crcpress.com/product/isbn/9781420086843.

[Rij79] C. J. Van RIJSBERGEN. *Information Retrieval*. 2nd. Newton, MA, USA : Butterworth-Heinemann, 1979. ISBN : 0408709294.

[Ris78] Jorma RISSANEN. "Modeling by shortest data description". In : *Automatica* 14.5 (1978), p. 465–471. DOI : 10.1016/0005-1098(78)90005-5. URL : https://doi.org/10.1016/0005-1098(78)90005-5.

[Ris08] Jorma RISSANEN. "Minimum description length". In : *Scholarpedia* 3.8 (2008), p. 6727.

[Riz+17a] Giuseppe RIZZO, Claudia D'AMATO et al. "Terminological Cluster Trees for Disjointness Axiom Discovery". In : *Proceedings of ESWC*. 2017, p. 184–201.

[Riz+17b] Giuseppe RIZZO, Claudia D'AMATO et al. "Tree-based models for inductive classification on the Web Of Data". In : *J. Web Semant.* 45 (2017), p. 1–22. DOI : 10.1016/j.websem.2017.05.001. URL : https://doi.org/10.1016/j.websem.2017.05.001.

[Riz+18] Giuseppe RIZZO, Nicola FANIZZI et al. "Approximate classification with web ontologies through evidential terminological trees and forests". In : *Int. J. Approx. Reasoning* 92 (2018), p. 340–362. DOI : 10.1016/j.ijar.2017.10.019. URL : https://doi.org/10.1016/j.ijar.2017.10.019.

[Rou+13] Amine Mohamed ROUANE HACENE et al. "Relational Concept Analysis : Mining Concept Lattices From Multi-Relational Data". In : *Annals of Mathematics and Artificial Intelligence* 67.1 (jan. 2013). Sous la dir. de Martin C. GOLUMBIC, p. 81–108. DOI : 10.1007/s10472-012-9329-3. URL : https://hal-lirmm.ccsd.cnrs.fr/lirmm-00816300.

[SA99] Guus SCHREIBER et Hans AKKERMANS. "Knowledge Engineering and Management : The Commonkads Methodology". In : 1999.

[Ser10] Baris SERTKAYA. "A Survey on how Description Logic Ontologies Benefit from FCA". In : *Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, October 19-21, 2010*. 2010, p. 2–21. URL : http://ceur-ws.org/Vol-672/paper2.pdf.

[Sow08] John F. SOWA. "Conceptual Graphs". In : *Handbook of Knowledge Representation*. 2008, p. 213–237. DOI : 10.1016/S1574-6526(07)03005-2. URL : https://doi.org/10.1016/S1574-6526(07)03005-2.

[SKW07] Fabian SUCHANEK, Gjergji M KASNECI et Gerhard M WEIKUM. "Yago : A Core of Semantic KnowledgeUnifying WordNet and Wikipedia". In : *16th international conference on World Wide Web*. Proceedings of the 16th international conference on World Wide Web. Banff, Canada, mai 2007, p. 697–697. DOI : 10.1145/1242572.1242667. URL : https://hal.archives-ouvertes.fr/hal-01472497.

[Sza06] Laszlo SZATHMARY. "Méthodes symboliques de fouille de données avec la plate-forme Coron". Thèse de doct. Henri Poincaré University, Nancy, France, 2006. URL : https://tel.archives-ouvertes.fr/tel-01754284.

[TKS02a] Pang-Ning TAN, Vipin KUMAR et Jaideep SRIVASTAVA. "Selecting the right interestingness measure for association patterns". In : *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2002, p. 32–41.

[TKS02b] Pang-Ning TAN, Vipin KUMAR et Jaideep SRIVASTAVA. "Selecting the right interestingness measure for association patterns". In : *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. 2002, p. 32–41. DOI : 10.1145/775047.775053. URL : https://doi.org/10.1145/775047.775053.

[Thi18] Élodie THIÉBLIN. "Do Competency Questions for Alignment Help Fostering Complex Correspondences ?" In : *Proceedings of the EKAW Doctoral Consortium 2018 co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France, November 13, 2018*. 2018. URL : http://ceur-ws.org/Vol-2306/paper8.pdf.

[VFS15] Johanna VÖLKER, Daniel FLEISCHHACKER et Heiner STUCKENSCHMIDT. "Automatic acquisition of class disjointness". In : *J. Web Sem.* 35 (2015), p. 124–139.

[VN11] Johanna VÖLKER et Mathias NIEPERT. "Statistical Schema Induction". In : *The Semantic Web : Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*. 2011, p. 124–138. DOI : 10.1007/978-3-642-21034-1_9. URL : https://doi.org/10.1007/978-3-642-21034-1%5C_9.

[Vou02] George VOUTSADAKIS. "Polyadic Concept Analysis". In : *Order* 19.3 (2002), p. 295–304. DOI : 10.1023/A:1021252203599. URL : https://doi.org/10.1023/A:1021252203599.

[Wil02] Rudolf WILLE. "Why can concept lattices support knowledge discovery in databases ?" In : *J. Exp. Theor. Artif. Intell.* 14.2-3 (2002), p. 81–92. DOI : 10.1080/09528130210164161. URL : https://doi.org/10.1080/09528130210164161.

[YHA03] Xifeng YAN, Jiawei HAN et Ramin AFSHAR. "CloSpan : Mining Closed Sequential Patterns in Large Datasets". In : *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*. 2003, p. 166–177. DOI : 10.1137/1.9781611972733.15. URL : https://doi.org/10.1137/1.9781611972733.15.

[Zak00] Mohammed Javeed ZAKI. "Scalable algorithms for association mining". In : *TKDE* 12.3 (2000), p. 372–390.

[ZH02] Mohammed Javeed ZAKI et Ching-Jui HSIAO. “CHARM : An Efficient Algorithm for Closed Itemset Mining”. In : *Proceedings of the 2002 SIAM International Conference on Data Mining*. Proceedings. Society for Industrial et Applied Mathematics, avr. 2002, p. 457–473. ISBN : 9780898715170. DOI : 10.1137/1.9781611972726.27. URL : https://epubs.siam.org/doi/10.1137/1.9781611972726.27.

[ZR05] Mohammed Javeed ZAKI et Naren RAMAKRISHNAN. “Reasoning about sets using redescription mining”. In : *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*. 2005, p. 364–373. DOI : 10.1145/1081870.1081912. URL : https://doi.org/10.1145/1081870.1081912.

[Zav+13] Amrapali ZAVERI, Dimitris KONTOKOSTAS et al. “User-driven quality evaluation of DBpedia”. In : *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems, ISEM '13, Graz, Austria, September 4-6, 2013*. 2013, p. 97–104. DOI : 10.1145/2506182.2506195. URL : https://doi.org/10.1145/2506182.2506195.

[Zav+16] Amrapali ZAVERI, Anisa RULA et al. “Quality assessment for Linked Data : A Survey”. In : *Semantic Web* 7.1 (2016), p. 63–93. DOI : 10.3233/SW-150175. URL : https://doi.org/10.3233/SW-150175.

Résumé

Dans cette thèse, nous nous intéressons au web des données et aux “connaissances” que potentiellement il renferme. Le web des données se présente comme un très grand graphe constitué de bases de triplets RDF connectées entre elles. Un triplet RDF, dénoté (sujet, prédicat, objet), représente une relation (le prédicat) qui existe entre deux ressources (le sujet et l'objet). Les ressources peuvent appartenir à une ou plusieurs classes, où une classe regroupe des ressources partageant des caractéristiques communes. Ainsi, ces bases de triplets RDF peuvent être vues comme des bases de connaissances interconnectées.

La plupart du temps ces bases de connaissances sont construites de manière collaborative par des utilisateurs. C'est notamment le cas de DBpedia, une base de connaissances centrale dans le web des données, qui encode le contenu de Wikipédia au format RDF. DBpedia est construite à partir de deux types de données de Wikipédia : d'une part, des données (semi-)structurées telles que les *infoboxes* et d'autre part les *catégories*, qui sont des regroupements thématiques de pages générés manuellement. Cependant, la sémantique des catégories dans DBpedia, c'est-à-dire la raison pour laquelle un agent humain a regroupé des ressources, n'est pas explicite. De fait, en considérant une classe, un agent logiciel a accès aux ressources qui y sont regroupées — il dispose de la définition dite *en extension* — mais il n'a généralement pas accès aux “motifs” de ce regroupement — il ne dispose pas de la définition dite *en intension*.

Dans cette thèse, nous cherchons à associer une définition à une catégorie en l'assimilant à une classe de ressources. Plus précisément, nous cherchons à associer une intension à une classe donnée en extension. La paire (extension, intension) produite va fournir la définition recherchée et va autoriser la mise en œuvre d'un raisonnement par classification pour un agent logiciel. Cela peut s'exprimer en termes de conditions nécessaires et suffisantes : si x appartient à la classe C, alors x a la propriété P (condition nécessaire), et si x a la propriété P, alors il appartient à la classe C (condition suffisante). Deux méthodes de fouille de données complémentaires nous permettent de matérialiser la découverte de définitions, la fouille de règles d'association et la fouille de redescriptions.

Dans le mémoire, nous présentons d'abord un état de l'art sur les règles d'association et les redescriptions. Ensuite, nous proposons une adaptation de chacune des méthodes pour finaliser la tâche de découverte de définitions. Puis nous détaillons un ensemble d'expérimentations menées sur DBpedia, où nous comparons qualitativement et quantitativement les deux approches. Enfin les définitions découvertes peuvent potentiellement être ajoutées à DBpedia pour améliorer sa qualité en termes de cohérence et de complétude.

Mots-clés: Découverte de connaissances • Analyse de Concepts Formels • Fouille de Redescriptions • Fouille de Règles • Construction de Définitions • Classification dans le Web des Données

Abstract

In this thesis, we are interested in the web of data and knowledge units that can be possibly discovered inside. The web of data can be considered as a very large graph consisting of connected RDF triple databases. An RDF triple, denoted as (subject, predicate, object), represents a relation (i.e. the predicate) existing between two resources (i.e. the subject and the object). Resources can belong to one or more classes, where a class aggregates resources sharing common characteristics. Thus, these RDF triple databases can be seen as interconnected knowledge bases.

Most of the time, these knowledge bases are collaboratively built thanks to human users. This is particularly the case of DBpedia, a central knowledge base within the web of data, which encodes Wikipedia content in RDF format. DBpedia is built from two types of Wikipedia data : on the one hand, (semi-)structured data such as infoboxes, and, on the other hand, categories, which are thematic clusters of manually generated pages. However, the semantics of categories in DBpedia, that is, the reason a human agent has bundled resources, is rarely made explicit. In fact, considering a class, a software agent has access to the resources that are regrouped together, i.e. the class extension, but it generally does not have access to the "reasons" underlying such a cluster, i.e. it does not have the class intension.

Considering a category as a class of resources, we aim at discovering an intensional description of the category. More precisely, given a class extension, we are searching for the related intension. The pair (extension, intension) which is produced provides the final definition and the implementation of classification-based reasoning for software agents. This can be expressed in terms of necessary and sufficient conditions : if x belongs to the class C, then x has the property P (necessary condition), and if x has the property P, then it belongs to the class C (sufficient condition). Two complementary data mining methods allow us to materialize the discovery of definitions, the search for association rules and the search for redescriptions.

In this thesis, we first present a state of the art about association rules and redescriptions. Next, we propose an adaptation of each data mining method for the task of definition discovery. Then we detail a set of experiments applied to DBpedia, and we qualitatively and quantitatively compare the two approaches. Finally, we discuss how discovered definitions can be added to DBpedia to improve its quality in terms of consistency and completeness.

Keywords: Knowledge Discovery • Formal Concept Analysis • Redescription Mining • Rule Mining • Definition Construction • Classification in the Web of Data