Définir les catégories de *DBpédia* avec des règles d'associations et des redescriptions

Justine Reynaud*, Esther Galbrun* Mehwish Alam** Yannick Toussaint*, Amedeo Napoli*

*LORIA (CNRS - INRIA - Université de Lorraine)
Campus Scientifique BP 239 – 54506 Vandœuvre-lès-Nancy
prenom.nom@loria.fr,
** Semantic Technology Lab, ISTC-CNR,
Rome, Italy.
mehwish.alam@istc.cnr.it

Résumé. *DBpédia*, qui encode les connaissances de *Wikipédia*, est devenue une base de référence pour le web des données. Les ressources peuvent y être répertoriées par des catégories définies manuellement, dont la sémantique n'est pas directement accessible par des machines. Dans cet article, nous proposons de remédier à cette lacune au moyen de méthodes de fouille de données, à savoir la recherche de règles d'associations et de motifs apparentés. Nous présentons une étude comparative de ces variantes sur une partie de *DBpédia* et discutons le potentiel des différentes approches.

1 Introduction

Le foisonnement des bases de connaissances sur le web pose de nouveaux enjeux quant à leur construction, leur enrichissement et leur interrogation. Nous prenons ici l'exemple de *DBpédia*. Dans cette base de connaissances, les ressources peuvent appartenir à une ou plusieurs *catégories*, générées manuellement. Cependant, les catégories sont définies en extension : on sait *comment* les ressources sont groupées, mais pas *pourquoi* elles sont groupées ainsi. Dans cet article, nous souhaitons définir les catégories en intension. C'est-à-dire que nous nous intéressons à des méthodes permettant d'expliciter les critères de regroupement.

Savoir caractériser ces catégories permettra non seulement d'enrichir *DBpédia* par cette nouvelle connaissance, mais aussi de corriger la base : les ressources assignées à tort à une catégorie ou inversement, les ressources non assignées à une catégorie à laquelle elles sont sensées appartenir pourront être détectées automatiquement.

La base de connaissances de *DBpédia* contient un ensemble de triplets RDF, dont les sujets correspondent à des articles de *Wikipédia*, associés à des paires (`predicat`, `objet`) qui représentent les catégories auxquelles ces articles sont rattachés et d'autres informations.

Afin de caractériser ces catégories en terme des autres informations, nous recherchons des définitions en exploitant des techniques de fouille de données. Chaque paire distincte (`predicat`, `objet`) est représentée par un attribut Booléen, tandis que chaque article est représenté par une entité, associée à un sous ensemble d'attributs représentant les catégories

et informations de l'article correspondant. On dénote Y' l'ensemble d'entités qui possèdent l'ensemble d'attributs Y, aussi appelé le *support* de Y. Une règle d'association entre deux ensembles d'attributs A et B, dénotée $A \to B$, s'interprète comme "si une entité possède A, alors elle possède aussi B". On associe à cette règle une valeur, la *confiance*, indiquant dans quelle mesure cette affirmation est vraie : $\mathrm{conf}(A \to B) = |A' \cap B'| \,/\, |A'|$. A et B sont appelés respectivement l'*antécédent* et la *conséquence* de la règle $A \to B$. Lorsque $\mathrm{conf}(A \to B) = 1$, on parle d'implication, dénotée $A \Rightarrow B$. Si $A \Rightarrow B$ et $B \Rightarrow A$, on a une définition $A \equiv B$.

Comme nous travaillons sur des données incomplètes, nous avons besoin d'une notion qui soit moins restrictive qu'une définition exacte. Nous introduisons donc la notion de *quasi-définition*, qui est à la définition ce que la règle d'association est à l'implication, et que nous dénotons donc $A \leftrightarrow B$. Pour évaluer la qualité de ces règles nous considérons la confiance minimum, $\mathrm{c}^{-}(A \leftrightarrow B) = \min(\mathrm{conf}(A \to B), \mathrm{conf}(B \to A))$, qui garantie une bonne confiance dans les deux directions et est plus adaptée à une recherche de quasi-définitions. Au contraire, la confiance maximum $\mathrm{c}^{+}(A \leftrightarrow B)$ prend une valeur de 1 dès que l'une des deux règles est une implication, même si sa réciproque a une confiance très faible.

2 Algorithmes

Pour extraire des quasi-définitions, nous avons recours à des techniques existantes, développées pour la fouille de règles d'associations et de redescriptions.

Règles d'association. Depuis l'introduction par Agrawal et al. (1993) des règles d'association, de nombreux algorithmes ont été développés, dont `Eclat` (Zaki, 2000) que nous utilisons ici. L'objectif est d'énumérer exhaustivement de manière efficace les règles ayant un support et une confiance au-dessus de seuils choisis. Dans la formulation originale du problème, on considère un unique jeu de données et tous les attributs peuvent se trouver aussi bien d'un côté que de l'autre de la règle. Dans notre formulation, au contraire, on considère deux ensembles d'attributs distincts. Nous avons donc ajouté une étape de post-traitement pour ne conserver que les règles qui satisfont la séparation des attributs.

Règles de traduction. Considérant un jeu de données Booléen à deux vues, l'approche proposée par van Leeuwen et Galbrun (2015), avec l'algorithme `Translator`, vise à obtenir un ensemble compact de règles permettant de "traduire" les données d'une vue vers l'autre et vice-versa. On peut effectivement reconstruire une vue en utilisant l'autre vue et un ensemble de règles d'associations. En partant d'un jeu de données vide et en considérant chaque règle tour à tour, on insère les attributs qui constituent la conséquence de la règle dans les lignes correspondant aux entités qui possèdent son antécédent. Une fois toutes les règles parcourues, on applique un masque pour corriger les erreurs restantes et reconstruire ainsi fidèlement la vue originale. Les règles de traduction peuvent être uni-directionnelles, s'appliquant uniquement dans une direction ou dans l'autre, ou bien bi-directionnelles. La sélection des règles s'inspire du principe de longueur de description minimum (MDL). C'est à dire que le critère de sélection est la compression : on cherche un ensemble de règles qui permette de représenter une vue étant donnée l'autre et vice-versa de la manière la plus succincte possible. Cela permet d'obtenir un nombre restreint de règles très informatives, en évitant la redondance.

Redescriptions. La fouille de redescriptions, introduite par Ramakrishnan et al. (2004), a pour but de trouver des descriptions alternatives pour un même ensemble d'entités. La simila-

ERT	Android_(OS)_devices ↔ *(dbo:operatingSysem dbr:Android_OS)*	(R1)
ET	Nokia_mobile_phones ↔ *(dbo:manufacturer dbr:Nokia), (dbp:manufacturer Nokia)*	(R2)
R	Nokia_mobile_phones ↔ *(dbp:manufacturer Nokia)*	(R3)
ER	Nokia_mobile_phones ↔ *(dbo:manufacturer Nokia)*	(R4)
ERT	Samsung_Galaxy ↔ *(dbo:manufacturer Samsung_Electronics), (dbo:operatingSysem Android_OS)*	(R5)
ERT	Sony ↔ *(dbo:manufacturer dbSony), (dbo:operatingSysem Android_OS)*	(R6)
ERT	Mobile_operating_systems ↔ *(rdf:type dbo:Software), (rdf:type dbo:Work)*	(R7)

FIG. 1: Exemples de règles obtenues par les algorithmes `Eclat` (`E`), `Translator` (`T`) et `ReReMi` (`R`). Lorsque le préfixe n'est pas précisé, il s'agit de *dbr*.

rité des descriptions est mesurée par le coefficient de Jaccard des ensembles d'entités décrites :

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

L'objectif est donc de construire des paires de descriptions ayant un support et une similarité au-dessus de seuils choisis. Nous utilisons l'algorithme `ReReMi` (Galbrun et Miettinen, 2012). Considérant un jeu de données à deux vues, incluant potentiellement des variables numériques, l'algorithme `ReReMi` utilise des heuristiques pour construire des descriptions qui peuvent impliquer à la fois conjonctions et disjonctions. Cependant, nous ne considérons ici que des variables Booléennes, et pour permettre la comparaison avec les autres approches, nous avons restreint l'algorithme aux seules conjonctions.

3 Résultats expérimentaux

Afin de mener une comparaison qualitative fine et détaillée, nous restreignons notre étude à un sous-ensemble de *DBpédia* en sélectionnant tous les triplets dont les sujets appartiennent la catégorie `Smartphones`. Le jeu de données ainsi obtenu contient 566 entités, 330 attributs représentant des catégories et 475 attributs représentant d'autres informations.

Nous avons appliqué les trois algorithmes, `Eclat`, `Translator` et `ReReMi` sur le jeu de données ainsi obtenu. Les règles candidates obtenues sont de la forme $R_{\mathbf{C}} \leftrightarrow R_{\mathbf{I}}$ où $R_{\mathbf{C}}$ et $R_{\mathbf{I}}$ sont des ensembles d'attributs, représentant des catégories et d'autres informations respectivement, interprétés comme des conjonctions.

Chaque algorithme retourne une liste ordonnée de règles candidates. Pour `Translator`, les règles sont retournées dans l'ordre dans lequel elles sont intégrées au modèle de compression. Avec `Eclat` et `ReReMi` elles sont triées par ordre décroissant de confiance maximum (c^+) pour le premier, et de Jaccard (J) pour le second. Pour un algorithme X, on dénote $\mathcal{R}_X^k$ les k premières règles retournées et $\mathcal{R}_X$ l'ensemble des règles retournées. Des exemples de règles obtenues sont présentés dans la Figure 1.

Nous avons évalué manuellement toutes les règles retournées par les algorithmes afin de vérifier si elles correspondent effectivement à une définition. Par exemple, la règle (R7) ne constitue pas une définition, puisque la catégorie *Mobile_operating_systems* est plus spécifique que le type *Smartphone*. La règle (R1) fournit en revanche une définition correcte de la catégorie *Android_(OS)_devices*. L'ensemble des définitions correctes obtenues avec les trois algorithmes forme une base de 20 définitions, dénotée $\mathcal{D}$, qui nous fournit une référence de vérité pour la comparaison des algorithmes.

Étant donné une règle candidate $R_{\mathbf{C}} \leftrightarrow R_{\mathbf{I}}$ et une définition $D_{\mathbf{C}} \equiv D_{\mathbf{I}}$ de la base $\mathcal{D}$, on dit que la règle *couvre* la définition si et seulement si $D_{\mathbf{C}} \subseteq R_{\mathbf{C}}$ et $D_{\mathbf{I}} \subseteq R_{\mathbf{I}}$. On souligne qu'une règle candidate peut couvrir plusieurs définitions de $\mathcal{D}$. Par exemple, la règle (R6) couvre les deux définitions *Sony* ≡ *(dbp:manufacturer dbr:Sony)* et *Sony* ≡ *(dbp:manufacturer dbr:Sony), (dbo:operatingSystem dbr:Android_(OS))*. Étant donné la base $\mathcal{D}$ et un ensemble de règles candidates $\mathcal{R}$ on peut déterminer la *couverture* de $\mathcal{D}$ par $\mathcal{R}$ comme $\mathrm{couv}(\mathcal{D}, \mathcal{R}) = \{D \in \mathcal{D} \mid \exists\, R \in \mathcal{R}, R \text{ couvre } D\}$. Pour chaque algorithme on peut ainsi calculer le pourcentage de définitions couvertes par les règles retournées, ce que nous appelons le *rappel* de l'algorithme et on peut calculer de la même manière la *précision* de chaque algorithme, c'est à dire le pourcentage de règles retournées qui constituent une définition correcte. On a donc

$$\mathrm{rappel}(X) = \frac{|\mathrm{couv}(\mathcal{D}, \mathcal{R}_X)|}{|\mathcal{D}|} \quad \text{et} \quad \mathrm{precision}(X) = \frac{|\{R \in \mathcal{R}_X \mid \exists\, D \in \mathcal{D}, R \text{ couvre } D\}|}{|\mathcal{R}_X|}.$$

Les statistiques des résultats obtenus avec chaque algorithme sont indiquées dans la Table 1a.

On s'intéresse également à l'évolution du nombre de définitions couvertes par les règles candidates à mesure que l'on augmente le nombre de règles considérées. La Figure 1b représente $\left|\mathrm{couv}(\mathcal{D}, \mathcal{R}_X^k)\right|$ en fonction de k. On compare l'ordre original dans lequel les règles candidates sont retournées par chaque algorithme (lignes pleines) et la couverture obtenue en les ordonnant par c^- décroissant (lignes pointillées). Les points pleins et vides représentent respectivement des règles qui sont des définitions correctes et celles qui n'en sont pas.

Les résultats obtenus avec `Eclat` sont proches des résultats obtenus avec `ReReMi`. Cela n'est pas surprenant car `ReReMi` est limité ici aux conjonctions et les mesures de confiance et de coefficient de Jaccard sont similaires. Par contre, `ReReMi` ne fait pas une énumération exhaustive et retourne donc moins de règles. L'algorithme `Translator` retourne des résultats nettement différents des autres algorithmes, ce qui s'explique par un critère de sélection des règles reposant sur la compression.

On observe de la redondance dans les ensembles de règles : certaines règles, bien que n'étant pas identiques, n'apportent pas d'information supplémentaire. Cela se manifeste par des règles prises en compte qui ne permettent pas de couvrir de nouvelle définition, c'est à dire que la valeur de k est incrémentée mais celle de la couverture n'augmente pas (cf. Figure 1b).

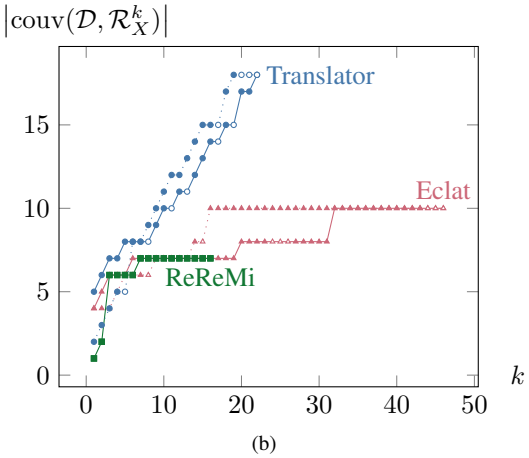
Les différents algorithmes utilisent différentes mesures de qualité pour sélectionner et trier les résultats : c^+ pour l'algorithme `Eclat`, J pour l'algorithme `ReReMi` et la compression `Translator`. On observe qu'utiliser c^- pour ordonner les règles candidates est particulièrement utile avec les résultats d'`Eclat`, les redondances se retrouvant en fin de liste. Par contre, comme `Translator` équilibre la qualité et la diversité des résultats, réordonner les résultats n'apporte pas de gain notable.

4 Discussion

Sélectionner et interpréter les règles. Les approches considérées ici recherchent toutes des associations entre deux ensembles d'attributs, mais elles ne favorisent pas les mêmes motifs, car elles ne partagent pas le même objectif. En particulier, `ReReMi` favorise les règles courtes dans le but de faciliter l'interprétation tandis que `Translator` favorise des règles plus longues qui contiennent un maximum d'information afin d'obtenir une représentation

	E	T	R
$\|\mathcal{R}\|$	52	23	17
supp min.	10	5	29
supp moy.	32	65	69
supp max.	183	566	183
$\|R_{\mathbf{I}}\|$ moy.	3.6	2.9	1.2
$\|R_{\mathbf{C}}\|$ moy.	2.6	3.7	1.6
c^- moy.	.70	.72	.76
c^+ moy.	.90	.90	.93
J moy.	.65	.67	.72
precision	84%	70%	100%
rappel	50%	90%	35%
redondance	70%	8%	76%

(a)



(b)

TAB. 1: Statistiques des ensembles de règles obtenus par les différents algorithmes (1a) et nombre de définitions couvertes en fonction du nombre de règles prises retournées (1b).

compacte. Par exemple, `Translator`, retourne la règle (R2) tandis que `ReReMi` retourne les deux règles (R3) et (R4). Ici, le prédicat *dbp :manufacturer* résulte de l'extraction automatique des données et le prédicat *dbo :manufacturer* – créé manuellement pour compléter le schéma de l'ontologie – forment un doublon. Les triplets contenant le prédicat *dbp :manufacturer* que nous avions extraits ont d'ailleurs été récemment supprimés de *DBpédia*. Cette observation soulève la question de l'interprétation de la conjonction. Celle-ci ne joue pas le même rôle dans (R2) et dans la règle (R5) : alors que les attributs associés par la conjonction dans (R2) sont redondants et qu'il est possible d'en éliminer un sans altérer la validité de la définition, ce n'est pas le cas dans (R4) où la présence des deux attributs est nécessaire.

L'interprétation des règles n'est valable que dans le domaine dans lequel elles ont été obtenues. Par exemple, (R3) n'est pas vraie en général, puisqu'il existe des appareils manufacturés par Nokia qui ne sont pas des téléphones mobiles. Elle a été obtenue et s'applique dans le monde clos des smartphones. Ce point ouvre des pistes de réflexion intéressantes, sur l'interprétation en monde ouvert, mais ce n'est pas notre priorité pour le moment : nous considérons les définitions obtenues uniquement dans un monde clos – celui qui est défini lors de l'extraction du jeu de données.

Utiliser des motifs plus expressifs. Nous nous sommes intéressés ici 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. L'algorithme `ReReMi` permet déjà de traiter des valeurs numériques, mais ce n'est pas le cas de `Translator`. Nous n'avons pas tenu compte du schéma de *DBpédia*, à savoir les hiérarchies de classes et de prédicats. Intégrer ces connaissances dans le processus de fouille permettrait d'affiner les définitions ou d'en trouver de nouvelles. En utilisant les outils de la FCA, cela pourrait se traduire par un *scaling*, c'est-à-dire l'ajout d'attributs liés à la hiérarchie. Mais nous pourrions également tirer

profit des structures de patrons, qui sont une généralisation de la FCA permettant d'intégrer une hiérarchie sur les attributs. Les algorithmes de fouille pourrait aussi être adaptés pour manipuler la hiérarchie directement. La formalisation de ces différentes approches et l'étude de leurs relations fournissent des pistes pour nos travaux à venir.

Dans cette étude, nous avons limité l'algorithme `ReReMi` aux conjonctions, mais il permet aussi d'utiliser des disjonctions, et ainsi d'obtenir par exemple la redescription *Samsung_mobile_phones* $\leftrightarrow$ *(dbo:manufacturer dbr:Samsung)* $\vee$ *(dbo:manufacturer dbr:Samsung_Electronics)*. Nous pensons que les disjonctions peuvent contribuer à identifier les doublons, en les regroupant dans une même règle. Tirer parti de l'expressivité des redescriptions permise par `ReReMi` offrirait des résultats plus fins. Cependant, l'utilisation de disjonctions apporte son lot de difficultés, en terme de complexité de la fouille comme de subtilités d'interprétation.

5 Conclusion

Dans cet article, nous avons comparé trois algorithmes pour caractériser les catégories de *DBpédia* : `Eclat`, `Translator` et `ReReMi`. `Translator` atteint le meilleur rappel, mais les règles sont difficilement interprétables, limitant leur incorporation à la base de connaissances. Les algorithmes `Eclat` et `ReReMi` atteignent un rappel similaire. Cependant `Eclat` retourne beaucoup plus de règles redondantes. L'algorithme `ReReMi` est prometteur car il permet d'utiliser des disjonctions. Plusieurs pistes sont envisageables pour la suite, dont notamment l'intégration de données plus riches et la prise en compte du schéma de l'ontologie.

Références

Agrawal, R., T. Imieliński, et A. Swami (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD Rec.*, Volume 22, pp. 207–216. ACM.

Galbrun, E. et P. Miettinen (2012). From Black and White to Full Color : Extending Redescription Mining Outside the Boolean World. *Stat Anal Data Min 5*(4), 284–303.

Ramakrishnan, N., D. Kumar, B. Mishra, M. Potts, et R. F. Helm (2004). Turning CARTwheels : an Alternating Algorithm for Mining Redescriptions. In *KDD'04*, pp. 266–275.

van Leeuwen, M. et E. Galbrun (2015). Association Discovery in Two-View Data. *TKDE 27*(12), 3190–3202.

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

Summary

DBpedia, which encode the knowledge of Wikipedia, has become a reference for the web of data. Resources can be indexed by manually-defined categories which are not accessible by a machine. In this article, we take a step towards addressing this accessibility issue by means of data mining techniques for mining association rules and redescriptions. We compare these approaches on a dataset from DBpedia and present our results.