Traitement des incompatibilités de candidats issus d'alignements entre plusieurs bases de connaissances

Fabien Amarger[1,2], Jean-Pierre Chanet[1], Ollivier Haemmerlé[2], Nathalie Hernandez[2], Catherine Roussey[1]

[1] UR TSCF, Irstea, 9 av. Blaise Pascal CS 20085, 63172 Aubière, France
prénom.nom@irstea.fr

[2] IRIT, UMR 5505, Université de Toulouse, UT2J 5 allées Antonio Machado, F-31058 Toulouse Cedex, France
prénom.nom@univ-tlse2.fr

Résumé : De nombreux travaux ont été proposés dans la littérature dans le but de construire des ontologies à partir de sources telles que les thesaurus ou les classifications. Certaines de ces sources sont disponibles sur le Web de données, au format SKOS. Dans nos travaux, nous proposons de construire une base de connaissances destinée à un besoin applicatif particulier, en exploitant un ensemble de sources disponibles sur le domaine considéré. L'originalité de notre approche réside dans le fait d'exploiter la redondance entre les sources afin d'en extraire des candidats (classes, individus, propriétés...). Nous présentons dans cet article la notion d'incompatibilité entre candidats, qui résulte de l'hypothèse de travail selon laquelle nous ne considérons que des relations d'équivalence simple entre les sources. Nous présentons également la génération de sous-ensembles de candidats compatibles afin d'obtenir un consensus cohérent entre les sources. Cette approche a été évaluée sur un cas d'étude réel concernant le domaine de la taxonomie du blé, réalisée en collaboration avec un expert.

Mots-clés : Acquisition de connaissances, candidat d'élément ontologique, traitement d'incompatibilités, taxonomie du blé

1 Introduction

Les données relatives à des domaines particuliers sont, le plus souvent, disponibles sur le Web dans des formes structurées (comme les bases de données ou les thésaurus) et sont consacrées à un usage donné. Les utilisateurs finals peuvent être déroutés face à l'abondance de données disponibles, de qualité différente, provenant de sources différentes, exprimées dans des formalismes différents. L'intérêt du Web de données liées ou Linked Open Data (LOD) est de faciliter l'interrogation de ces données ouvertes en permettant d'établir des liens entre elles. Des approches (Soergel *et al.* (2004); Villazón-Terrazas *et al.* (2010)) ont été proposées afin de formaliser la transformation de données structurées dans le but de les publier sur le LOD. Néanmoins, ces approches nécessitent beaucoup de temps et d'interactions avec l'utilisateur pour être efficaces. De plus, très peu d'approches exploitent l'intérêt de la transformation multi-sources, Amarger *et al.* (2013). Nous avons donc proposé une méthode permettant la construction d'une base de connaissances (BC) pour un domaine spécifique, réutilisant plusieurs sources non-ontologiques, telles que des thésaurus ou des classifications. L'idée principale est de modéliser un domaine d'étude en construisant un module ontologique. Chaque source est ensuite analysée pour enrichir le module et ainsi le peupler avec de nouveaux éléments extraits de la source. Nous obtenons donc plusieurs versions du module, chaque version du module est une BC issue de la source. Les BC sont ensuite alignées par des outils d'alignement. Un regroupement d'éléments alignés provenant de différentes BC est appelé un *candidat*. L'objectif de nos

travaux est d'extraire les connaissances communes à plusieurs BC, Amarger *et al.* (2014).

Nous proposons ici une évolution de cette approche en partant d'une hypothèse liée aux alignements entre BC. Un alignement entre deux BC est un ensemble de correspondances entre éléments ontologiques des BC considérées. Nous ne considérons comme correspondances entre éléments ontologiques que des relations d'équivalence de cardinalité 1 :1, dites correspondances simples. Rappelons qu'un regroupement d'éléments ontologiques alignés constitue un candidat. Les outils d'alignements sont capables d'obtenir des correspondances d'équivalence ayant une cardinalité 1 :n ce qui nous amène une ambiguïté. Les correspondances 1 :n peuvent générer au moins n candidats. Nous nommons ces ensembles de candidats des candidats incompatibles. L'objectif du travail présenté est de faciliter la validation des candidats. Nous allons donc chercher à découvrir des sous-ensembles de candidats compatibles entre eux. Cet ensemble est nommé une extension. Une extension ne doit pas être incluse dans une autre. L'objectif est de trouver l'extension la plus grande validée par un expert.

Cet article est organisé de la façon suivante : (1) présentation du processus général, (2) génération d'extension de candidats compatibles, (3) travaux connexes, (4) évaluation sur la taxonomie des blés.

2 Processus général

Notre méthode se compose de trois étapes : (1) "analyse de sources" qui permet de déterminer quelles sources vont être utilisées dans le processus, (2) "transformation des sources" qui permet d'obtenir une base de connaissances source pour chaque source par transformation automatique, en se fondant sur un module ontologique donné et (3) "la fusion des bases de connaissances sources". C'est sur cette dernière étape que se fonde en grande partie l'originalité de notre approche, puisqu'elle permet la fusion de ces différentes bases de connaissances sources en se basant sur l'idée que plus une connaissance apparaît dans plusieurs sources et plus sa confiance augmente. Nous avons détaillé ces processus dans des travaux précédents, Amarger *et al.* (2014).

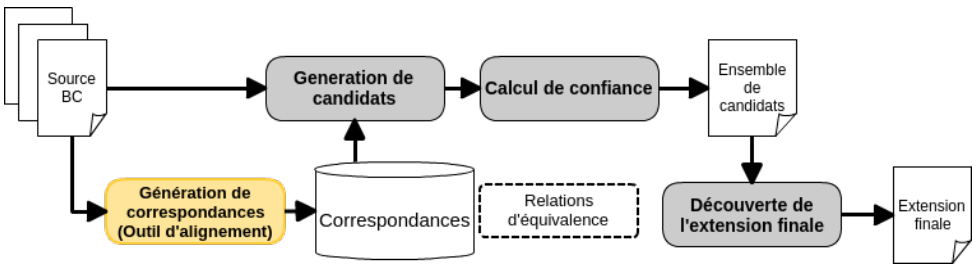

FIGURE 1 – Processus de fusion de bases de connaissances

La figure 1 présente le processus de fusion des bases de connaissances sources en 4 étapes. Les trois premières ayant été décrites dans nos travaux précédents (Amarger *et al.* (2014) et Roussey *et al.* (2013)), nous nous focalisons dans cet article sur la dernière activité (Découverte de l'extension finale).

3 Génération d'extension de candidats compatibles

Partant du constat que les outils d'alignements proposent des correspondances (Euzenat & Shvaiko (2007)) d'équivalence de cardinalité 1 :n, nous présentons dans un premier temps notre méthode afin de générer des candidats. Nous étudierons ensuite comment gérer les incompatibilités entre candidats par rapport au constat précédent et nous verrons enfin comment générer une extension.

Candidats : Un candidat $cand$ représente un élément susceptible d'appartenir à la base de connaissances finale que nous cherchons à construire. Cet élément est le résultat de la fusion d'éléments issus de plusieurs BC, jugés équivalents par des outils d'alignement. Un candidat $cand = (V_{cand}, E_{cand})$ est un graphe non-orienté connexe dont les sommets V_{cand} sont des éléments ontologiques provenant de BC différentes et les arêtes E_{cand} sont les correspondances d'équivalence. Nous considérons ici qu'un candidat n'existe que s'il contient au moins deux éléments ontologiques provenant de deux BC différentes. Les candidats avec un seul élément ontologique ne sont pas considérés car nous cherchons les connaissances communes. De plus nous cherchons à générer les candidats maximaux, ce qui signifie que nous ne considérons pas les candidats inclus dans un autre.

Incompatibilités : Nous posons une incompatibilité entre les candidats partageant un élément ontologique, ce qui implique qu'un seul de ces candidats peut faire partie de la base de connaissances finale. Nous obtenons ici un graphe d'incompatibilité dont les sommets sont des candidats et les arêtes des incompatibilités.

Extensions : Une extension est le sous-ensemble de candidats n'ayant pas de lien d'incompatibilités entre eux. Trouver toutes les extensions revient à résoudre un problème connu en théorie des graphes : MCE (Maximum Clique Enumeration). Il faut pour cela considérer le graphe complémentaire au graphe d'incompatibilité et donc rechercher toutes les cliques maximales. Pour résoudre ce problème de MCE, plusieurs algorithmes existent. Le plus courant est le Bron Kerbosch (Bron & Kerbosch (1973)). Il existe néanmoins un certain nombre d'améliorations de cet algorithme, tel que l'algorithme de Tomita (Tomita *et al.* (2006)) ou, plus récemment, l'algorithme de Eppstein et Strash (Eppstein & Strash (2011)). Ce problème étant NP-difficile, il est inenvisageable de découvrir toutes les cliques maximales possibles. Notre problème est donc de trouver un moyen d'obtenir une extension et de la faire valider par un utilisateur dans ces conditions de complexité. Pour résoudre ce problème, nous utilisons un solver CSP (Constraint Satisfaction Problem), GLPK (GNU Linear Programming Kit – `https://www.gnu.org/software/glpk/`), pour utiliser la technique de Branch and Bound qui nous permettra de trouver une solution au problème en maximisant une fonction objective. Nous cherchons ici à maximiser le nombre de candidats dans une extension. De cette manière, la première extension rencontrée qui maximise le nombre de candidat sera retournée en un temps fini.

Validation : Une phase de validation des candidats est utile pour ajouter des contraintes au problème afin de converger vers une solution optimale, autrement dit, pour ce qui nous concerne, vers une extension dont tous les candidats sont validés. L'idée étant de présenter à un expert les candidats d'une extension, un à un, pour validation. Si l'expert valide le candidat présenté, alors on peut ajouter la contrainte selon laquelle l'extension doit contenir ce candidat. Inversement, si l'expert ne valide pas le candidat, alors on ajoute la contrainte selon laquelle l'extension ne doit pas contenir ce candidat. À chaque candidat non validé, l'algorithme est relancé afin de trouver

une nouvelle extension avec les nouvelles contraintes. L'algorithme s'arrête quand il n'y a plus de candidats à valider dans l'extension. Grâce à cette méthode, nous pouvons être assurés de ne présenter à l'expert qu'un minimum de candidats à valider en déduisant automatiquement que tous les candidats incompatibles avec un candidat validé ne peuvent pas apparaître dans l'extension finale. Ceci permet de réduire le nombre d'interactions pour faciliter le travail d'un expert. Le temps nécessaire pour la validation par l'expert des candidats est, dans le pire des cas, égal, en nombre d'interactions, au nombre de candidats générés. En d'autres termes, dans le pire des cas, l'expert aura à valider tous les candidats s'ils sont tous compatibles les uns avec les autres. Dès qu'une incompatibilité apparaît, ce nombre d'interactions est forcément diminué.

4 Travaux connexes

Les travaux connexes concernant la fusion de bases de connaissances et plus particulièrement la gestion des incompatibilités (ou incohérences) sont assez récents. La plupart des travaux (Trojahn *et al.* (2011); Abbas & Berio (2013); Raunich & Rahm (2014)) cherchent à détecter des incompatibilités entre les différentes correspondances établies entre deux sources par une ou plusieurs approches d'alignement. L'idée sous-jacente est de déterminer quelles correspondances peuvent être ignorées dans le but de lever l'incompatibilité. Dans ces travaux, une incompatibilité est détectée lorsque les correspondances établies rendent la base de connaissances inconsistantes d'une point de vue logique. Certains travaux (Abbas & Berio (2013),Trojahn *et al.* (2011)) considèrent également les préférences des deux agents utilisant chacun des ontologies pour établir ces incompatibilités. Le traitement des incompatibilités se fait soit par l'utilisation de règles (Raunich & Rahm (2014)), soit en cherchant des sous-ensembles compatibles, notamment en utilisant la théorie de l'argumentation (Trojahn *et al.* (2011)). Notre approche se place très clairement dans cette deuxième catégorie : nous pouvons assimiler nos candidats à des arguments et nos incompatibilités à des attaques entre arguments en suivant la théorie de Dung (1995). La différence est ici que nous manipulons des incompatibilités entre candidats d'éléments ontologiques composés de correspondances entre plusieurs sources. Nous ne remettons en question que les correspondances $1 : n$ ayant mené à la génération de plusieurs candidats incompatibles. Nous identifions le candidat valide en exploitant les correspondances.

5 Évaluation sur la taxonomie des blés

Les jeux de données que nous avons utilisés proviennent d'un projet de construction d'une base de connaissances sur les céréales Roussey *et al.* (2013) et Amarger *et al.* (2014). Nous avons utilisé uniquement les données concernant la taxonomie des blés. Pour ce faire, nos experts ont sélectionné les sources suivantes :
Agrovoc (http ://aims.fao.org/standards/agrovoc/about),
TaxRef (http ://inpn.mnhn.fr/programme/referentiel-taxonomique-taxref),
NCBI Taxonomy (http ://www.ncbi.nlm.nih.gov/).

En utilisant le module ontologique AgronomicTaxon [1](Roussey *et al.* (2013)) et des patrons de transformations adaptés à chacune des trois sources, nous avons utilisé notre approche pour générer 3 BC. Ces BC ont été alignées avec LogMap, Jiménez-Ruiz & Grau (2011). Nous avons

1. https://sites.google.com/site/agriontology/home/irstea/agronomictaxon

généré un ensemble de candidats. Comme présenté précédemment, nous générons le graphe d'incompatibilités des candidats. Le tableau 1 présente quelques données sur ce graphe.

Sources	nb_{eo}	nb_{cand}	nb_{incomp}
Agrovoc	11	150	1555
TaxRef	19		
NCBI	130		

TABLE 1 – Graphe d'incompatibilités

$nb_{ext_{max}}$	$nb_{ext_{finale}}$	$nb_{interaction}$	$Ratio$
25	23	62	0.41

TABLE 2 – Résultats

eo : élément ontologique, ext_{max} *: extension maximum possible,* ext_{finale} *: extension finale,* $Ratio = \frac{nb_{interaction}}{nb_{cand}}$

En utilisant notre méthode, nous avons demandé à un expert de valider ces candidats en comptant le nombre d'interactions qui ont été nécessaires pour obtenir l'extension finale. Nous obtenons les résultats présentés dans le tableau 2.

Nous pouvons observer plusieurs faits notables dans ce tableau. Tout d'abord, la taille de l'extension maximale possible au début de l'exercice est de 25 candidats, alors que la taille de l'extension validée est de 23 candidats. Ceci vient du fait que 2 candidats n'ont pas été validés par l'expert ainsi que tous les candidats incompatibles avec ceux-ci. De plus, il a fallu 62 interactions de l'expert. Il a donc validé ou invalidé 62 candidats sur les 150 présents initialement. On observe donc un ratio d'interactions de 0.41, ce qui signifie que 41% des candidats ont dû être observés par l'expert afin d'obtenir l'extension finale. Cette technique est donc avantageuse puisque l'on gagne plus de 50% des interactions nécessaires à la validation de tous les candidats générés.

6 Conclusion

Nous avons présenté dans cet article un moyen de générer des incompatibilités entre des candidats provenant de l'extraction multi-source d'éléments ontologiques. Ces incompatibilités peuvent être utilisées pour générer des extensions, des sous-ensembles cohérents de candidats. Nous avons aussi présenté un moyen de valider les candidats en exploitant ces incompatibilités pour limiter les interactions de l'expert afin d'obtenir l'extension optimale. Cette méthode a été validée sur un jeu de données réel provenant de plusieurs sources pour la création d'une base de connaissances sur la taxonomie des plantes.

Il serait intéressant de faire évoluer ces travaux en utilisant des scores associés aux candidats Amarger *et al.* (2014) et des pondérations des correspondances proposés par les outils d'alignement, pour améliorer la fonction objective à optimiser. L'utilisation des scores des candidats dans la fonction objective permettrait de présenter en premier les candidats les plus pertinents.

Durant les évaluations, un phénomène est apparu qui pourrait permettre de réduire considérablement le nombre d'interactions de l'expert pour la validation des candidats. Parmi les candidats incompatibles issus d'une même correspondance $1 : n$, les candidats non validés par l'expert comportent moins de labels communs que le candidat validé par l'expert. Il serait donc intéressant de pouvoir présenter en premier les candidats partageant le plus de label. Cette idée est généralisable en considérant tout le voisinage des candidats (et pas seulement les labels) dans la fonction objective à maximiser, ce qui reviendrait à privilégier les candidats partageant le plus de voisins communs (sommets ou arcs) : par exemple des sommets labels alignés ou des candidats sommets liés par la même relation.

Références

ABBAS M. & BERIO G. (2013). Creating ontologies using ontology mappings : Compatible and incompatible ontology mappings. *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, p. 143–146.

AMARGER F., ROUSSEY C., CHANET J., HAEMMERLÉ O. & HERNANDEZ N. (2013). Etat de l´art : Extraction d´information à partir de thésaurus pour générer une ontologie. *INFORSID*, p. 29–44.

AMARGER F., ROUSSEY C., CHANET J., HAEMMERLÉ O. & HERNANDEZ N. (2014). Skos sources transformations for ontology engineering : Agronomical taxonomy use case. *MTSR*.

BRON C. & KERBOSCH J. (1973). Algorithm 457 : finding all cliques of an undirected graph. *Communications of the ACM*, p. 575–577.

DUNG P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, p. 321–357.

EPPSTEIN D. & STRASH D. (2011). Listing all maximal cliques in large sparse real-world graphs. *Experimental Algorithms*, p. 364–375.

EUZENAT J. & SHVAIKO P. (2007). *Ontology matching*.

JIMÉNEZ-RUIZ E. & GRAU B. C. (2011). Logmap : Logic-based and scalable ontology matching. *The Semantic Web–ISWC 2011*, p. 273–288.

RAUNICH S. & RAHM E. (2014). Target-driven merging of taxonomies with atom. *Information Systems*, p. 1–14.

ROUSSEY C., CHANET J., CELLIER V. & AMARGER F. (2013). Agronomic taxon. In *WOD*, p.5̃.

SOERGEL D., LAUSER B., LIANG A., FISSEHA F., KEIZER J. & KATZ S. (2004). Reengineering thesauri for new applications : The AGROVOC example. *Journal of Digital Information*, p. 1–23.

TOMITA E., TANAKA A. & TAKAHASHI H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, p. 28–42.

TROJAHN C., EUZENAT J., TAMMA V. & PAYNE T. R. (2011). Argumentation for reconciling agent ontologies. p. 89–111.

VILLAZÓN-TERRAZAS B., SUÁREZ-FIGUEROA M. C. & GÓMEZ-PÉREZ A. (2010). A pattern-based method for re-engineering non-ontological resources into ontologies. *Int. J. Semantic Web Inf. Syst.*, p. 27–63.