shapes,arrows,automata,backgrounds


Stage de Master 2 Recherche en Informatique

Alignement d'ontologies : Passage à l'échelle

Auteur :
Fayçal HAMDI

Maîtres de stage :
Brigitte SAFAR
Haifa ZARGAYOUNA

Organisme d'accueil :
Université Paris-Sud XI, CNRS LRI & INRIA Futurs

Secrétariat - tél : 01 69 15 75 18 Fax : 01 69 15 42 72
courrier électronique : Jacques.Laurent@lri.fr
12 mars – 11 septembre 2008

Table des matières

Résumé

L'explosion du nombre de sources d'informations accessibles via le Web multiplie les besoins de techniques permettant l'intégration de ces sources. L'alignement d'ontologies représente un thème de recherche très important dans les systèmes d'intégration puisqu'il autorise la prise en compte conjointe de ressources décrites par des ontologies différentes. L'apparition de très grandes ontologies pour modéliser des domaines complexes comme la médecine ou l'agronomie apportent aujourd'hui un nouveau défi pour les techniques d'alignement d'ontologies, leur passage à l'échelle. Le travail décrit dans ce rapport s'intègre dans les travaux menés dans l'équipe IASI du LRI autour du système d'alignement *T*axoMap et a pour objectif de permettre son utilisation sur des ontologies de grandes tailles. Notre proposition consiste à partitionner les deux ontologies à aligner en un sous ensemble de blocs de taille plus limitée, puis à appliquer *T*axoMap pour aligner 2 à 2 les paires de blocs jugés les plus proches. Pour partitionner les ontologies, nous proposons deux méthodes inspirées principalement des techniques de partitionnement par co-clustering que nous avons adaptées pour prendre en compte l'objectif d'alignement. L'originalité de notre approche consiste à prendre en compte le plus tôt possible le contexte de l'alignement dans nos algorithmes de partitionnement. Dans la première méthode, nous décomposons l'une des deux ontologies en fonction des partitions obtenues pour l'autre, dans la deuxième méthode, nous décomposons les deux ontologies en fonction l'une de l'autre. Nous testons ensuite ces deux méthodes de partitionnement sur différentes ontologies puis nous analysons les résultats obtenus.

MotsClefs

Ontologies, alignement, partitionnement, similarité

CHAPITRE 1

Introduction

Le développement rapide des technologies internet a engendré un intérêt croissant dans la recherche sur le partage et l'intégration de sources dispersées dans un environnement distribué. Le Web sémantique [8] offre la possibilité à des agents logiciels de comprendre des sources sémantiquement liées dans une architecture décentralisée. Les ontologies ont été reconnues comme une composante essentielle pour le partage des connaissances et la réalisation de cette vision. En définissant les concepts associés à des domaines particuliers, elles permettent à la fois de décrire le contenu des sources à intégrer et d'expliciter le vocabulaire utilisable dans des requêtes par des utilisateurs. Toutefois, il est peu probable qu'une ontologie globale couvrant l'ensemble des systèmes distribués puisse être développée. Dans la pratique, les ontologies de différents systèmes sont développées indépendamment les unes des autres par des communautés différentes. Ainsi, si les connaissances et les données doivent être partagées, il est essentiel d'établir des correspondances sémantiques entre les ontologies qui les décrivent.

La tâche d'alignement d'ontologies (recherche de mappings, appariements ou mises en correspondance entre concepts) est donc particulièrement importante dans les systèmes d'intégration puisqu'elle autorise la prise en compte conjointe de ressources décrites par des ontologies différentes. Ce thème de recherche a donné lieu à de très nombreux travaux [6].

Les méthodes actuelles d'alignement s'appuient en général sur des mesures calculant la similarité de couple de concepts issus des deux ontologies. Ces mesures sont pour la plupart fondées sur les caractéristiques lexicales des labels des concepts et/ou les caractéristiques structurelles des ontologies [4],[5], ce qui implique la comparaison de chaque description de concept d'une ontologie avec les descriptions de tous les concepts de l'autre ontologie. Avec l'apparition de langages de représentations très performants et très expressifs, de grandes ontologies ont été développées dans des domaines complexes (e.g médecine, agronomie) et comportent plusieurs dizaines de milliers de concepts (AGROVOC[1] : 28 439, NALT[2] : 42 326, NCI[3] : 27 652).

Les différentes solutions proposées pour l'alignement sont souvent testées sur des ontologies de petite taille (quelques centaines de concepts) pour lesquelles la plupart des correspondances peuvent être trouvées de manière automatique. Quand les ontologies sont de très grande taille, l'efficacité des méthodes d'alignement automatique diminue considérablement que ce soit en terme de temps d'exécution, de taille mémoire utilisée ou de la précision des mappings obtenus du fait de l'augmentation du bruit.

Une solution possible pour résoudre ce problème est d'essayer de limiter la taille des ensembles de concepts en entrée des algorithmes d'alignement, et pour cela de partitionner les deux ontologies à aligner en plusieurs blocs, afin de n'avoir à traiter que des blocs de taille raisonnable.

Les deux méthodes que nous proposons sont inspirées principalement des techniques de co-clustering, qui consistent à exploiter, en plus des informations exprimées par les relations entre les concepts au sein d'une même ontologie, celles qui correspondent aux relations pouvant exister entre les concepts des deux ontologies. Le fait que des concepts des deux ontologies puissent avoir exactement le même label et puissent être reliés par une relation d'équivalence est un exemple de relation facile à calculer même sur des ontologies de grande taille, et dont nous allons tirer parti dans notre proposition.

[1] http ://www4.fao.org/agrovoc/
[2] http ://agclass.nal.usda.gov/agt/
[3] http ://www.mindswap.org/2003/CancerOntology/

Nous commencerons donc par identifier avec une mesure de similarité stricte et peu couteuse à calculer, les couples de concepts issus des deux ontologies et de label identique, que nous appellerons les *ancres*. Dans la première méthode, nous partitionnons une première ontologie (la mieux structurée), puis nous nous fondons à la fois sur les blocs générés pour cette première ontologie et sur les ancres trouvées auparavant pour partitionner la deuxième ontologie. Dans la deuxième méthode, nous partitionnons les deux ontologies en fonction l'une de l'autre.

Le reste de ce rapport est organisé comme suit : dans la section suivante, nous présentons le contexte de ce travail et quelques travaux dans le domaine du partitionnement puis nous détaillons plus précisément l'algorithme de partionnement FALCON [9] sur lequel nous nous sommes appuyés dans notre proposition. Dans la section 3, nous proposons nos deux méthodes de partitionnement adaptées à la tâche d'alignement. Dans la section 4, nous présentons et analysons les résultats expérimentaux pour démontrer l'efficacité de ces méthodes. Enfin, nous concluons et donnons quelques perspectives en section 5.

CHAPITRE 2

Contexte et état de l'art

Le problème qui nous intéresse ici est celui du passage à l'échelle des méthodes d'alignement d'ontologies.

Dans notre contexte de travail, les ontologies manipulées sont des taxonomies ou ontologies "légères", comprenant un ensemble de concepts C modélisant un domaine d'application et une hiérarchie de subsomption entre concepts H_C. Un concept est défini par son label et les relations de sous-classes qui le relient à d'autres concepts. Le label est un nom (chaîne de caractères) qui décrit des entités en langage naturel et qui peut être une expression composée de plusieurs mots. Les relations de sous-classes établissent des liens entre concepts. Il s'agit de l'unique association sémantique utilisée dans la classification. Une taxonomie est généralement représentée par un graphe acyclique dont les nœuds sont les concepts et les arcs correspondent aux liens de sous-classes.

La tâche d'alignement d'ontologies consiste à générer le plus automatiquement possible des relations ou appariements entre les concepts de deux ontologies. Le système d'alignement *TaxoMap* avec lequel nous travaillons est un système d'alignement orienté : son objectif est de trouver des appariements entre les concepts d'une des deux ontologies dite *Source*, O_S, avec les concepts de l'autre ontologie dite *Cible* ou *Target*, O_T. Les types de relations établies entre les concepts par appariements peuvent être des relations d'équivalence *isEq*, de subsomption *isA* ou de proximité *isClose*. Nous reprendrons ici cette terminologie.

Quand les ontologies sont de très grande taille, l'efficacité des méthodes d'alignement automatique diminue considérablement. La solution que nous envisageons est d'essayer de limiter la taille des ensembles de concepts en entrée de l'outil d'alignement, et pour cela de partitionner les deux ontologies à aligner en plusieurs blocs, afin de n'avoir à traiter que des blocs de taille raisonnable. Bien sûr, cela signifie que les différents blocs obtenus après les partitions devront être ensuite alignés par paires, composées de 2 blocs issus chacun d'une des 2 ontologies initiales et le nouveau problème à résoudre sera d'éviter d'avoir à tester toutes les paires de blocs possibles.

Notre problème est d'élaborer un algorithme de partitionnement adapté au contexte de l'alignement. Partitionner un ensemble E consiste à trouver des sous ensembles $E_1, E_2, ..., E_n$, d'éléments sémantiquement proches c.à.d. liés par un ensemble de relations important.

Une manière d'obtenir un tel partitionnement peut donc consister à maximiser les relations à l'intérieur d'un sous-ensemble et à minimiser les relations entre les différents sous-ensembles. La qualité du résultat d'un partitionnement peut être apprécié selon différents critères :

- La taille des blocs générés : les blocs doivent avoir une taille raisonnable, i.e. inférieure au nombre d'éléments que peut traiter l'outil d'alignement (4 000 concepts maximum pour *TaxoMap*).
- Le nombre de blocs générés : ce nombre doit être le plus faible possible pour limiter le nombre de paires de blocs à aligner.
- Le degré de dépendance entre les blocs : un bloc sera dit faiblement dépendant des autres si les relations (lexicales et structurelles) sont fortes à l'intérieur du bloc et faibles à l'extérieur.

L'objectif est que les éléments des deux ontologies suceptibles d'être appariés se retrouvent finalement concentrés dans un ensemble minimal de blocs qui seront effectivement comparés.

Dans les domaines d'applications réelles, les ontologies devenant de plus en plus volumineuses, de nombreux travaux [7], [1, 2] et [9] se sont intéressés au problème de leur partitionnement.

Ainsi les travaux de [7] visent la décomposition d'une ontologie en sous-blocs (ou *îlots*) indépendants les uns des autres, de façon à faciliter en toute généralité différentes opérations sur les ontologies comme la maintenance, la visualisation, la validation ou le raisonnement. Les travaux de [1] s'intéressent plus particulièrement aux problèmes de raisonnement et cherche à construire des modules centrés autour d'une sous-thématique qui soient cohérents et auto-suffisants pour raisonner. Seul [9] a pour objectif l'alignement d'ontologies mais nous verrons que sa méthode de décomposition ne prend pas complètement en compte toutes les contraintes imposées par cet objectif, en particulier le fait de travailler sur deux ontologies.

2.1 Décomposition d'une ontologie en îlots

L'objectif de [7] est de décomposer une ontologie en blocs indépendants et cohérents. La méthode consiste à trouver des blocs (îlots) à partir d'un graphe de dépendance. Le processus de partitionnement passe par cinq étapes :

1. ***Création du graphe de dépendance*** Le graphe de dépendance est extrait à partir du fichier source de l'ontologie. L'idée est que les éléments de l'ontologie (concepts, relations, instances) sont représentés par des nœuds du graphe. Les liens entre les nœuds sont introduits si les éléments correspondants sont liés dans l'ontologie.
2. ***Déterminer la force de dépendance*** La force des dépendances entre les concepts est déterminée en utilisant des algorithmes d'analyse de réseau : elle est basée sur le nombre de liens auxquels participe un nœud. L'idée est que moins un nœud est relié à d'autres nœuds, plus ses liens sont forts. Des poids peuvent être introduits pour représenter l'importance des différents types de liens, ainsi on peut décider que les liens traduisant les relations de sous-classe ont un impact plus important que les autres relations du domaine.
3. ***Déterminer les modules*** Un module ou îlot ("line island") est défini comme étant un ensemble de nœuds, de taille comprise entre des valeurs minimale et maximale données, tels que la force de chaque connexion interne à l'ensemble est supérieure à la force de chacune des connexions reliant un des nœuds de l'ensemble avec l'extérieur.
4. ***Attribuer les concepts isolés*** Le fait de devoir fixer une borne minimale à la taille des modules est très contraignant. Un mauvais choix de cette borne fait apparaître, d'après les auteurs, de très nombreux concepts isolés dont l'affectation doit être forcée dans le module avec lesquel ils ont la plus forte connexion.
5. ***Fusion*** L'algorithme a aussi tendance à construire beaucoup de petits blocs avec une très forte corrélation interne dont il faut là aussi forcer la fusion. Celle-ci est décidée si certains modules voisins sont assez fortement liés. Dans de nombreux cas, il n'existe qu'un module adjacent pour fusionner. Dans les cas où plus d'un module adjacent existe, la fusion est faite avec le voisin le plus proche, déterminé par la force des dépendances entre les modules.

Cette méthode n'est pas adaptée pour notre contexte, car le processus de génération des modules impose une contrainte sur la taille minimale des modules générés qui n'est pas appropriée pour l'alignement. De plus, elle construit beaucoup trop de petits blocs, ce qui aura un impact négatif sur l'étape d'alignement final.

2.2 Modules autonomes pour le raisonnement

Dans [1, 2] la méthode de partitionnement vise à produire des modules autonomes dans le sens où toutes les inférences à l'intérieur d'un module peuvent être faites uniquement sur la base d'un raisonnement local [2]. La méthode comporte trois étapes de base :

1. ***Safety Check*** Une ontologie n'est partitionnable que si elle ne contient pas d'axiomes dits "dangereux" pour la préservation de la compatibilité sémantique de ses partitions.
2. ***Partitionnement*** L'algorithme débute avec une seule partition. Il crée enuite une nouvelle partition en y introduisant un concept ou un axiome quelconque puis tous les concepts ou axiomes dépendants, i.e. qui y font référence et peuvent être déplacés dans la nouvelle partition sans violer la condition de complétude.
3. ***Génération de module*** Le partitionnement créé dans la deuxième étape est utilisé pour déterminer les modèles. Cela se fait par la fusion des partitions au sein des modules qui se chevauchent. A ce stade, la redondance peut éventuellement être introduite dans l'ontologie.

Cette méthode garantit que tous les concepts reliés par des liens de subsumption seront regroupés dans un seul module. Ceci est un inconvénient majeur pour l'alignement d'ontologies qui comportent des milliers de relations de subsomption (le cas de AGROVOC et NALT). Les auteurs conviennent qu'elle peut conduire à la création de très mauvaises répartitions de tailles des modules dans le cas d'ontologies "monolithiques", qui ne seront pas du tout adaptées pour l'alignement.

2.3 Méthode FALCON

La méthode proposée dans Falcon [9] consiste à partitionner chaque ontologie en blocs en utilisant la méthode de clustering Rock [3], puis à mesurer la proximité de chacun des blocs d'une ontologie avec chaque bloc de l'autre ontologie de façon à n'effectuer l'alignement qu'entre les concepts des paires de blocs les plus proches.

Pour effectuer la partition, alors que Rock considère que les liens entre les concepts ont tous la même valeur, Falcon introduit la notion de *liens pondérés* qui s'appuie sur deux mesures de similarité entre concepts, une mesure linguistique et une mesure structurelle.

2.3.1. Mesures de similarité

Similarité lexicale

Soient d_i (resp. d_j) la chaine de caractère correspondant à la description du concept c_i (resp. c_j) et $comm(d_i, d_j)$ (resp. $diff(d_i, d_j)$) le nombre de caractères communs (resp. différents) dans les chaines d_i et d_j.
La similarité lexicale entre deux concepts c_i et c_j, $Sim_L(c_i,c_j)$, se calcule comme suit :

$$sim_L(c_i, c_j) = comm(d_i, d_j) - diff(d_i, d_j) + Winkler(d_i, d_j)$$

où le terme $Winkler(d_i, d_j)$ est ajouté pour améliorer le résultat en utilisant la méthode de Winkler. Les concepts c_i et c_j sont jugés similaires si $Sim_L(c_i, c_j) > 0.65$.

Similarité structurelle

Soient c_i, c_j deux concepts d'une même ontologie O, c_{ij} leur plus petit ancêtre commun et $depthOf(c)$ la distance en nombre d'arcs entre le concept c et la racine de O. FALCON mesure la similarité structurelle $aff_s(c_i, c_j)$ en utilisant la mesure de Wu et Palmer (ref) qui se calcule comme suit :

$$aff_s(c_i, c_j) = \frac{2 * depthOf(c_{ij})}{depthOf(c_i) + depthOf(c_j)}$$

Le calcul de similarité structurelle entre les concepts d'une ontologie de grande taille peut prendre beaucoup de temps. En considérant que seuls les concepts de profondeurs adjacentes auront des similarités élevées, FALCON ne compare que les concepts qui satisfont la relation suivante :

$$|depthOf(c_i) - depthOf(c_j)| \leq 1$$

Les liens pondérés

Le calcul du lien pondéré entre deux concepts, $link(c_i, c_j)$, s'effectue comme suit :

$$link(c_i, c_j) = \begin{cases} aff(c_i, c_j) & if\ aff(c_i, c_j) > \epsilon_1 \\ 0 & otherwise \end{cases}$$

$$aff(c_i, c_j) = \alpha \cdot aff_s(c_i, c_j) + (1 - \alpha) \cdot Sim_L(c_i, c_j)$$

où ϵ_1 est un seuil donné tel que $\epsilon_1 \in [0, 1]$ et $\alpha \in [0, 1]$ permet à l'utilisateur de faire varier le poids relatif des mesures de similarité.

2.3.2. Algorithme de Partitionnement

L'algorithme permettant à FALCON de partitionner une ontologie en blocs s'appuie sur deux notions essentielles : la *cohésion* au sein d'un bloc et le *couplage* entre deux blocs distincts. La cohésion mesure la somme des poids des liens reliant les concepts appartenant à un même bloc et le couplage, la somme des poids des liens reliant les concepts appartenant à deux blocs différents.

Ces deux notions sont représentées au sein d'une même mesure dite *goodness* dont le sens varie suivant qu'elle s'applique sur un bloc unique B_i ou sur deux blocs distincts, B_i et B_j tels que $B_i \neq B_j$:

$$Cohésion(B_i) = goodness(B_i, B_i),$$
$$Couplage(B_i, B_j) = goodness(B_i, B_j)\ avec\ B_i \neq B_j.$$

$$goodness(B_i, B_j) = \frac{\sum_{c_i \in B_i, c_j \in B_j} link(c_i, c_j)}{sizeOf(B_i) \cdot sizeOf(B_j)}$$

L'algorithme prend en entrée l'ensemble B des n blocs à partitionner, où chaque bloc est réduit au départ à un unique concept, et le nombre k de blocs que l'on souhaite obtenir en sortie. Dans chaque itération, l'algorithme choisit tout d'abord le bloc qui a la cohésion maximale, puis le bloc qui a la valeur de couplage maximale avec ce premier bloc, et fusionne ces deux blocs. Le pseudo-code de cet algorithme est présenté ci-dessous :

Falcon(B, k)
Pour chaque bloc B_i dans B **faire**
initialiser la valeur de la cohésion de B_i,
calculer la valeur de couplage de B_i avec tous les autres blocs,
fin
Tant que le nombre courant de blocs $m > k$ **faire**
choisir B_i qui a la valeur de cohésion maximale ;
choisir B_j le bloc qui a la valeur de couplage maximale avec B_i ;
fusionner les blocs B_i et B_j dans B_p ;
mettre à jour les valeurs de cohésion et de couplage de B_p ;
supprimer B_i et B_j ;
Pour chaque bloc B_i sauf B_p **faire**
mettre à jour la valeur de couplage de B_i ;
fin
$m := m - 1$;
fin
fin

2.3.3. Identification des paires de blocs à aligner

Une fois réalisé le partitionnement des deux ontologies, l'évaluation de la proximité des blocs s'effectue en s'appuyant sur des *ancres*, i.e. des appariements préalablement connus entre les termes des deux ontologies, définis par des techniques de comparaison de chaînes de caractères ou par un expert. Plus deux blocs contiennent d'ancres communes, plus ils sont jugés proches.

Soient k le nombre de blocs générés par la partition d'une ontologie O et B_i un de ces blocs, k‘ le nombre de blocs générés par la partition de la deuxième ontologie O‘ et B'_j un de ces blocs. Soit la fonction anchors(B_u, B'_v) qui calcule le nombre d'ancres prédéfinies partagées par deux blocs B_u et B'_v. Le nombre d'ancres contenues par un bloc B_i est donc calculé par la somme $\sum_{v=1}^{k'} anchors(B_i, B'_v)$.

La relation de *Proximité* entre deux blocs B_i et B'_j est finalement définie comme suit :

$$Proximity(B_i, B'_j) = \frac{2.anchors(B_i, B'_j)}{\sum_{u=1}^{k} anchors(B_u, B'_j) + \sum_{v=1}^{k'} anchors(B_i, B'_v)}$$

Les paires de blocs alignées sont toutes les paires ayant une proximité supérieure à un seuil $\epsilon_2 \in [0, 1]$. Un bloc pourra donc être aligné avec plusieurs blocs de l'autre ontologie ou avec aucun.

2.3.4. Exemple

Nous avons appliqué l'algorithme Falcon sur deux petites ontologies jouets pour visualiser son comportement et faciliter la comparaison ultérieure avec nos propres propositions.

FIG.1 présente les deux ontologies O_S et O_T avant et après le processus de partitionnement en utilisant la méthode Falcon. Pour effectuer ces partitionnements, nous avons utilisé une version de l'algorithme trouvée sur le web qui ne se base que sur la similarité structurelle entre concepts. Les labels des concepts n'interviennent donc pas dans le traitement. De plus, la variable de contrôle

de l'algorithme n'est pas, comme il est indiqué dans le papier cité, le nombre de blocs k souhaités en sortie, mais la taille maximale des blocs fusionnables.

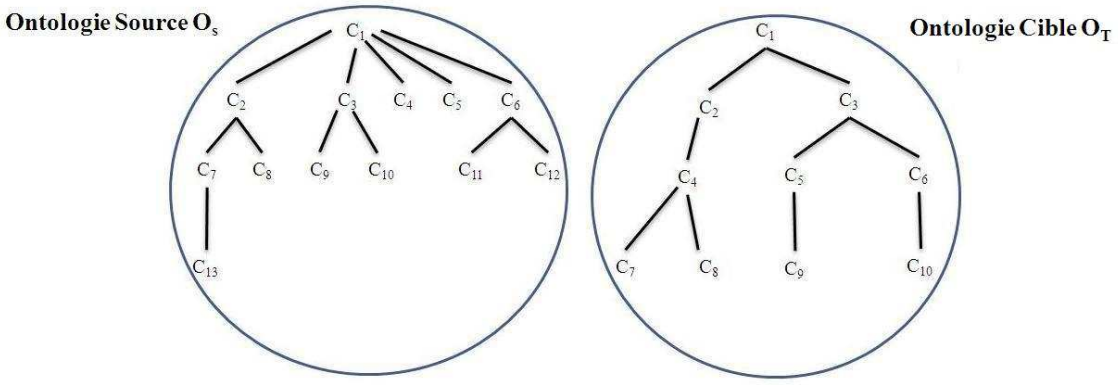

FIG.1a Les ontologies de l'expérimentation test

Pour cette expérimentation, nous avons fixé cette taille limite à 4, ce qui signifie que l'algorithme a le droit de fusionner 2 blocs de taille 3 pour former au plus un bloc de taille 6. Nous sommes ainsi sûrs d'obtenir au moins 2 blocs dans O_T qui contient 10 concepts et 3 blocs dans O_S qui en contient 13. Le partitionnement successif de O_T puis O_S fait effectivement apparaître 2 blocs pour O_T, B_{T1} et B_{T2} contenant 5 concepts chacun, et 3 blocs pour O_S, B_{S1}, B_{S2} et B_{S3} contenant respectivement 4, 3 et 6 concepts.

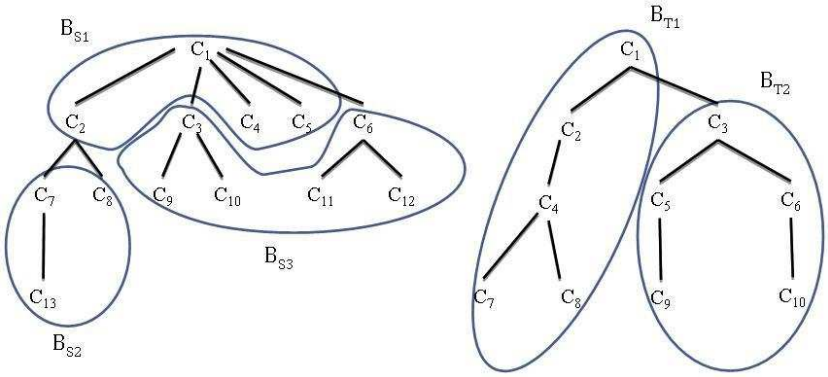

FIG.1b Les blocs construits avec la méthode FALCON

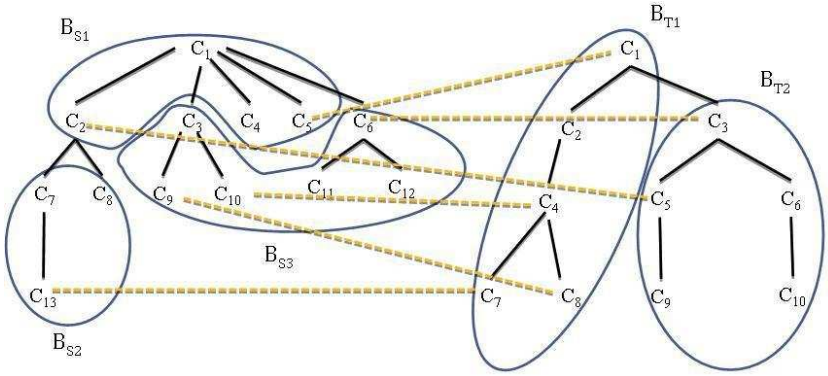

FIG.1c Les ancres partagées par les différents blocs

Le bloc B_{S1} contient 2 ancres, l'une partagée avec B_{T1}, l'autre avec B_{T2}. Le bloc B_{S2} n'en contient qu'une, partagée avec B_{T1}, et le bloc B_{S3} en contient 3, 2 partagées avec B_{T1}, la troisième avec B_{T2}.

Le calcul de proximité basé sur les ancres partagées par les 2 ontologies doit être effectué sur toutes les paires de blocs possibles (ici, 6 paires).

Le nombre de paires effectivement alignées dépend de la valeur fixée pour le seuil : plus celui-ci est bas, plus on multiplie les alignements et les chances de retrouver des appariements, mais plus on augmente aussi le temps d'exécution. Si le seuil est élevé, on passe moins de temps à aligner des blocs éloignés mais on perd peut être des appariements potentiels.

Par exemple, la paire (B_{S1},B_{T1}) ne partageant qu'une ancre alors que les blocs en contiennent respectivement 2 et 4, sa proximité est égale à 0.33. Si le seuil est fixé à 0.5, la paire n'est pas alignée et l'ancre partagée n'est pas retrouvée dans les appariements. Si le seuil est fixé plus bas, on retrouvera toutes les ancres dans les appariements, mais on devra aligner toutes les paires de blocs possibles à l'exception de la paire (B_{S2},B_{T1}) qui ne partage pas d'ancre.

Cette méthode permet donc à Falcon de décomposer des ontologies volumineuses, mais cette décomposition est faite a priori, sans prendre en compte l'objectif d'alignement en s'appliquant sur chaque ontologie indépendamment l'une de l'autre. Pour effectuer l'alignement, Falcon doit d'abord identifier quels sont les blocs les plus proches qui doivent être alignés entre eux et pour cela calculer la proximité des différents blocs en s'appuyant sur les couples d'ancres préalablement connus. Le partitionnement ayant été fait au départ à l'aveugle, certaines ancres pourront ne pas se trouver dans des blocs finalement alignés et l'alignement résultant ne comprendra pas forcément tous les appariements possibles. Enfin, le calcul des blocs pertinents à aligner est coûteux (en temps de traitement).

Malgré ces critiques, l'algorithme de décomposition de Falcon est le plus adapté à la tâche d'alignement puisqu'il permet de contrôler la taille maximale des blocs construits.

Les deux méthodes que nous proposons le réutilisent en modifiant sa façon de générer les blocs. Notre idée est de prendre en considération au plus tôt lors du partitionnement, toutes les informations fournies par les relations existant entre les concepts des deux ontologies et d'essayer de faire, au moins dans la deuxième méthode, du co-clustering.

CHAPITRE 3

Propositions

Notre objectif est donc de prendre en compte au plus tôt l'information fournie par les *ancres* (les concepts des deux ontologies qui ont exactement les mêmes labels) et de l'utiliser dès le départ dans la phase de partition, pour générer les blocs.

Les deux méthodes proposées commencent par la même étape de calcul des ancres en recherchant, à l'aide d'une mesure de similarité facile à calculer, les équivalences lexicales entre les concepts des deux ontologies. Une fois ces ancres identifiées, les deux méthodes diffèrent dans leur exploitation.

Le système d'alignement *TaxoMap* avec lequel nous travaillons est un système d'alignement orienté : son objectif est de trouver des appariements entre les concepts d'une des deux ontologies dite *Source*, O_S, avec les concepts de l'autre ontologie dite *Cible* ou *T*arget, O_T.

La première méthode prend aussi en compte l'orientation du processus et décompose tout d'abord l'ontologie cible O_T, puis décompose ensuite O_S en fonction des blocs obtenus pour O_T, en forcant dès le début de la partition, le regroupement dans un même bloc de O_S de toutes les ancres trouvées dans un même bloc de O_T. Nous sommes ainsi sûrs de retrouver dans la phase d'alignement finale toutes les relation d'équivalence exprimées par les ancres. La deuxième méthode décompose O_T et O_S en fonction l'une de l'autre, en favorisant la construction des blocs qui contiennent le plus d'ancres.

Les sections suivantes présentent successivement la mesure de similarité utilisée pour calculer les ancres à moindre coût, puis le détail des deux méthodes.

3.1 Mesure de similarité lexicale

Nous utilisons pour calculer les ancres une mesure de similarité lexicale stricte. Nous disons que deux concepts sont équivalents seulement si leurs labels sont parfaitement identiques. Aucun prétraitement (filtrage, tokenisation, lemmatisation) ne sera effectué sur ces labels. Nous définissons cette mesure comme suit :

$$sim_{light}(Label(c_i), Label(c_j)) = \begin{cases} 1 & if\ Label(c_i) = Label(c_j) \\ 0 & sinon \end{cases}$$

c_i, c_j : deux concepts tel que $c_i \in O_i$ et $c_j \in O_j$

3.2 Méthode 1

La première méthode force le partitionnement de O_S à suivre celui réalisé pour O_T. Elle comprend quatre étapes en plus du calcul des ancres :

1. Partitionner l'ontologie cible O_T en plusieurs blocs B_{Ti} en utilisant l'algorithme FALCON.
2. Identifier, dans chacun des blocs construits pour O_T, l'ensemble des ancres appartenant à ce bloc. Chacun de ces ensembles constituera le noyau ou *centre* CB_{Si} d'un futur bloc B_{Si} à générer dans l'ontologie source O_S.
3. Partitionner l'ontologie source autour des *centres* CB_{Si} identifiés dans l'étape précédente.
4. Aligner chaque bloc de O_S avec le bloc de O_T correspondant.

3.2.1 Déterminer les blocs de O_T

Pour déterminer les blocs de l'ontologie cible O_T, nous utilisons l'algorithme FALCON tel qu'il est implémenté dans le code que nous avons trouvé sur le web. La différence entre cet algorithme et celui décrit dans la présentation théorique faite en section 2.3 à partir de l'article [9], est que le lien pondéré calculé entre les concepts ne dépend que de la similarité structurelle ($\alpha = 1$ dans l'équation suivante) :

$$aff(c_i, c_j) = \alpha \cdot aff_s(c_i, c_j) + (1 - \alpha) \cdot Sim_L(c_i, c_j)$$

3.2.2 Déterminer les centres de O_S

Les centres des futurs blocs de l'ontologie source O_S sont déterminés en se basant sur deux critères : les couples d'ancres identifiés entre O_S et O_T, et les blocs B_{Ti} construits à partir de l'ontologie cible O_T. Pour chaque bloc B_{Ti} construits à l'étape précédente, nous regroupons les concepts de O_S qui ont des équivalents dans ce bloc en un paquet, CB_{Si}, en utilisant l'algorithme suivant :

Entrées :
$T = \{B_{T1}, B_{T2}, ...B_{Tn}\}$ l'ensemble des blocs de l'ontologie cible O_T.
$E = \{E_1, E_2, ...E_m\}$ l'ensemble des couples de labels identiques trouvés, tel que $E_i = (c_{Si}, c_{Tj})$, où c_{Si} est un concept de l'ontologie source O_S et c_{Tj} est un concept de l'ontologie cible O_T.
Sortie :
$S = \{CB_{S1}, CB_{S2}, ...CB_{Sn}\}$ les centres des futurs blocs de l'ontologie source O_S.

Centre (T, E, S)
  Pour chaque bloc B_{Ti} dans T **faire**
    initialiser $CB_{Si} = \emptyset$
    Pour chaque concept c_{Tk} dans B_{Ti} **faire**
      Pour chaque équivalence E_j dans E **faire**
        Si $c_{Tk} \in E_j$ **alors** $CB_{Si} = CB_{Si} \cup c_{Sk}$
      Fin
    Fin
Fin

3.2.3 Partition de O_S autour des centres CB_{Si}

Après l'identification des centres des futurs blocs de O_S, nous appliquons l'algorithme FALCON avec la différence suivante. Au lieu d'introduire en entrées l'ensemble des m concepts de l'ontologie comme m blocs réduits chacun à un unique concept, nous introduisons les n centres identifiés à l'étape précédente, comme autant de blocs distincts mais regroupant plusieurs concepts, puis les autres concepts de O_S qui n'ont pas d'équivalents dans O_T chacun dans un bloc individuel. La cohésion des blocs représentant les centres de O_S est initialisée avec la valeur de cohésion maximale.

3.2.4 Identification des blocs à aligner

Le principe de l'algorithme FALCON étant de fusionner entre eux les blocs qui ont un couplage élevé, deux blocs constitués à partir de deux centres distincts peuvent être fusionnés par l'algorithme s'ils ont un couplage important. Si un bloc B_{Sk} est le résultat de la fusion de deux blocs

formés autour des centres CB_{Si} et CB_{Sj}, il devra être aligné avec les deux blocs de O_T correspondants. Un bloc de O_T ne sera aligné qu'une seule fois. L'identification des blocs de O_T devant être finalement alignés avec ceux de O_S n'est donc pas immédiate et consiste à retrouver pour chaque centre initial, i.e. pour un élément quelconque de ce centre, le bloc B_{Sk} auquel il a été finalement affecté. Ce calcul est effectué par l'algorithme suivant :

Entrées :
$T = \{B_{T1}, B_{T2}, ...B_{Tn}\}$ l'ensemble des blocs de l'ontologie cible O_T.
$S = \{CB_{S1}, CB_{S2}, ...CB_{Sn}\}$ les centres des futurs blocs de O_S générés à partir des blocs B_{Ti}
$BS = \{B_{S1}, B_{S2}, ...B_{Sm}\}$ les blocs effectivement constitués à partir de O_S.
Sortie :
$P = \{(B_{S1}, P_1), (B_{S2}, P_2)...(B_{Sm}, P_m)\}$ où les P_i représentent l'ensemble des blocs B_{Tj} devant être alignés avec un bloc B_{Si}, ensemble éventuellement vide si le bloc B_{Si} n'a pas été constitué à partir d'un centre.

Align (S, T, BS, P)
Pour chaque bloc B_{Si} dans BS **faire**
initialiser $P_i = \emptyset$
Pour chaque centre CB_{Sj} dans S **faire**
Soit c_j le premier élément de CB_{Sj},
Si $c_j \in B_{Si}$
Alors $P_i = P_i \cup B_{Tj}$, Supprimer CB_{Sj} de S
Fin
Fin

Dans l'état courant de notre implémentation, un bloc B_{Si} qui n'a pas été constitué à partir d'un centre ne sera pas pris en compte dans le processus d'appariement. En effet, ce bloc ne contenant pas d'ancre, nous n'avons pas encore étudié d'heuristiques permettant de choisir les blocs B_{Ti} avec lequel on pourrait essayer de le rapprocher.

3.2.5 Exemple

Nous avons appliqué la méthode 1 sur les ontologies de l'exemple précédent présenté en FIG. 1. La génération des blocs de la cible s'effectue comme dans la méthode FALCON (FIG. 2a).

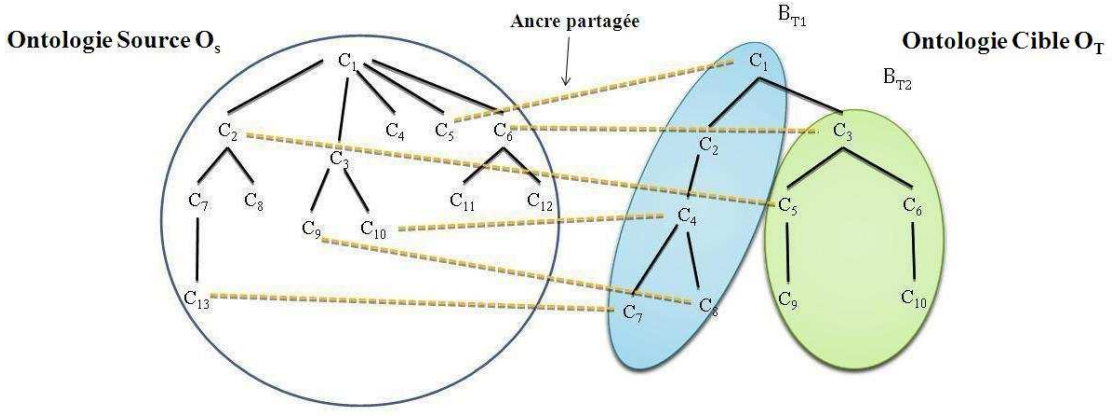

FIG. 2a Génération des blocs de la cible identique à celle de FALCON

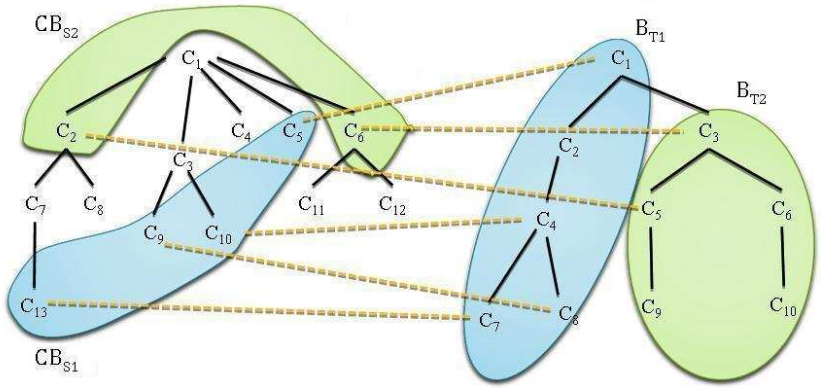

FIG. 2b Identification des centres des blocs de l'ontologie source

A partir des blocs générés pour la cible B_{T1} et B_{T2}, l'algorithme identifie les centres des futurs blocs de l'ontologie source ; CB_{S1} : {c_5, c_9, c_{10}, c_{13}} et CB_{S2} : {c_2, c_6} (FIG. 2b).
La partition de O_S autours des centres constitués se poursuit ensuite comme dans l'algorithme de Falcon (FIG. 2c).

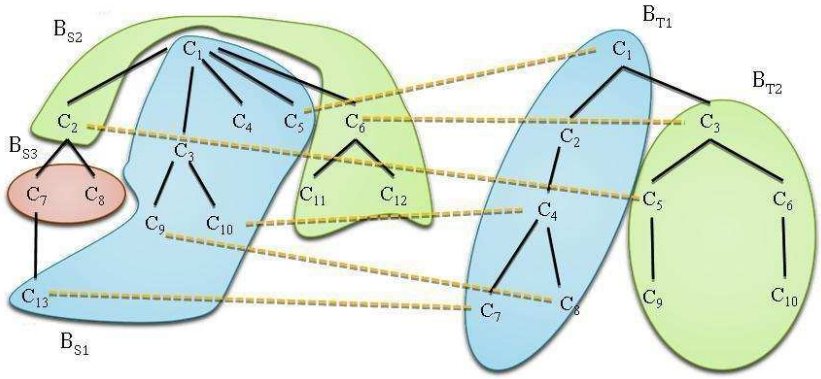

FIG. 2c Génération des blocs de la source à partir des centres constitués précédemment

Pour l'identification des paires de blocs à aligner, l'algorithme réaffecte à chaque bloc de O_S les centres (donc les blocs de O_T correspondant) qui lui appartiennent, en testant l'appartenance du premier élément de chaque centre non encore réaffecté. Pour B_{S1}, le 1^{er} élément c_5 du centre correspondant à B_{T1} lui appartient : la paire (B_{S1}, B_{T1}) devra donc être alignée. Le 1^{er} élément c_2 du centre correspondant à B_{T2} ne lui appartient pas, et il n'y a plus d'autre centre à affecter : B_{S1} ne sera donc aligné qu'avec B_{T1}. Pour B_{S2} le centre associé à B_{T1} déjà affecté n'est plus à prendre en compte, ne reste que celui correspondant à B_{T2}. Son 1^{er} élément c_2 appartenant à B_{S2}, la paire (B_{S2}, B_{T2}) sera alignée. Il ne reste plus de centre à affecter à B_{S3} qui ne participera pas à l'alignement.

3.3 Méthode 2

L'idée de cette méthode est de partitionner les deux ontologies en même temps, c.à.d. de faire du co-clustering. Le problème est que nous ne pouvons pas traiter réellement ces ontologies en parallèle du fait de leur grande taille. Pour simuler le parallélisme, nous partitionnons l'ontologie cible en nous fondant sur toutes les équivalences identifiées avec la source, et nous partitionnons la source en nous fondant sur une partie des équivalences trouvés dans les blocs générés pour la cible.
Prendre en compte les relations d'équivalence identifiées entre les ontologies dès le partitionnement de O_T, devrait permettre par la suite de faciliter la recherche des paires de blocs les plus proches et d'améliorer les résultats de l'alignement. Nous pouvons ainsi dire que ce partitionnement, contrairement à celui de FALCON ou à celui implémenté dans la méthode 1, est orienté alignement pour les deux ontologies à partitionner.

La deuxième méthode comprend ainsi trois étapes :

1. Partitionner la première ontologie O_T en utilisant l'algorithme FALCON mais en prenant en compte lors de la génération des blocs, l'ensemble des ancres.
2. Partitionner la deuxième ontologie O_S de la même manière mais en se basant à chaque fois sur une partie des ancres qui appartiennent à un bloc de l'ontologie O_T.
3. Aligner les paires de blocs partageant le plus d'ancres.

3.3.1 Déterminer les blocs de O_T

Pour déterminer les blocs de l'ontologie cible O_T, nous utilisons l'algorithme FALCON en modifiant la définition de la mesure de *goodness* pour prendre en compte les liens entre les deux ontologies. Nous lui ajoutons un coefficient qui représente la proportion d'ancres présentes dans un bloc de O_T sur le nombre d'ancres total identifiées avec l'ontologie O_S.
De ce fait, au cours de la génération des blocs, le choix du bloc qui a la valeur maximale de cohésion ou de couplage ne dépend pas seulement des relations des concepts à l'intérieur ou à l'extérieur des blocs d'une même ontologie, mais aussi des relations d'équivalences identifiées avec l'autre ontologie.

L'équation de goodness devient :

$$goodness\,(B_i, B_j) = \alpha \left(\frac{\sum_{c_i \in B_i, c_j \in B_j} link\,(c_i, c_j)}{sizeOf\,(B_i) \cdot sizeOf\,(B_j)} \right) + (1-\alpha) \left(\frac{\sum_{c_j \in B_j, c_k \in O_S} sim_{light}\,(c_j, c_k)}{\sum_{c_n \in O_T, c_m \in O_S} sim_{light}\,(c_n, c_m)} \right) \quad (3.1)$$

où $\alpha \in [0,1]$, B_i et B_j sont 2 blocs de O_T, $\sum_{c_j \in B_j, c_k \in O_S} sim_{light}(c_j, c_k)$ représente le nombre d'ancres présentes dans B_j et $\sum_{c_n \in O_T, c_m \in O_S} sim_{light}(c_n, c_m)$, le nombre d'ancres total.

3.3.2 Déterminer les blocs de O_S

Les blocs de l'ontologie source O_S sont générés en se basant sur les poids des liens entre les concepts de O_S, les relations d'équivalences entre les deux ontologies et les blocs de l'ontologie O_T. Si nous considèrons un bloc B_i dans O_S avec une valeur de cohésion maximale, le calcul de goodness pour trouver le bloc ayant la valeur de couplage maximale avec B_i se base non seulement sur

les relations structurelles dans O_S, mais aussi sur les relations d'équivalences avec le bloc de O_T qui a le maximum d'équivalents avec B_i.

L'équation de goodness devient :

$$goodness\left(B_i, B_j\right) = \alpha \left(\frac{\sum_{c_i \in B_i, c_j \in B_j} link\left(c_i, c_j\right)}{sizeOf\left(B_i\right) \cdot sizeOf\left(B_j\right)} \right) + (1-\alpha) \left(\frac{\sum_{c_j \in B_j, c_k \in B_k} sim_{light}\left(c_j, c_k\right)}{\sum_{c_n \in O_T, c_m \in O_S} sim_{light}\left(c_n, c_m\right)} \right) \quad (3.2)$$

où $\alpha \in [0,1]$, B_i et B_j sont 2 blocs distincts de O_S et où B_k est le bloc de O_T qui partage le plus d'ancres avec B_i.
Lors des opérations de fusion, nous conservons pour chaque bloc de O_S l'ensemble des ancres appartenant au bloc, ce qui facilitera lors de la recherche des blocs à aligner, l'identification des blocs partageant le plus d'ancres.

3.3.3 Identification des blocs à aligner

La conservation des ancres introduites dans chaque bloc facilite la phase de calcul des paires de blocs partageant le plus d'ancres, un bloc de O_S ne s'alignant qu'avec un seul bloc de O_T. Une fois identifiés les blocs de O_S devant être alignés avec le même bloc de O_T nous pouvons éventuellement les fusionner si leur taille le permet, afin de diminuer le nombre de combinaisons à faire lors du processus d'alignement.

3.3.4 Exemple

Les figures 3a et 3b montrent le résultat du partitionnement de nos ontologies jouets avec la méthode 2.

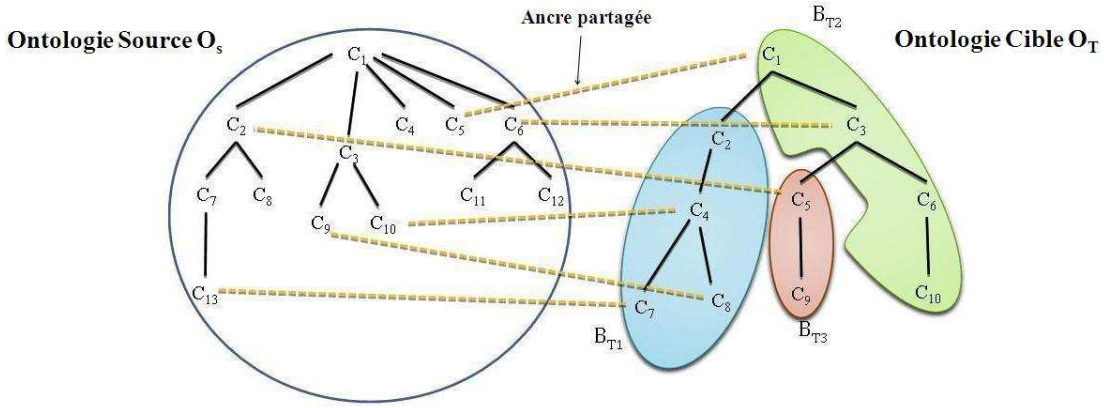

FIG. 3a Génération des blocs de la cible

L'alignement s'effectuera entre les blocs qui partagent le plus d'ancres : B_{S2} et B_{S3} seront alignés avec B_{T1} et B_{S1}, avec B_{T2}. B_{T3} ne participe pas à l'alignement puisque qu'il ne contient qu'une ancre partagé, avec B_{S2}, qui partage plus d'ancres avec B_{T2} qu'avec lui. Nous perdons donc ici l'appariement d'une ancre, (c_2, c_5), mais nous diminuons les temps d'alignement et pouvons espérer avoir construits par co-clustering des blocs prenant plus en considération les relations partagées entre les deux ontologies.

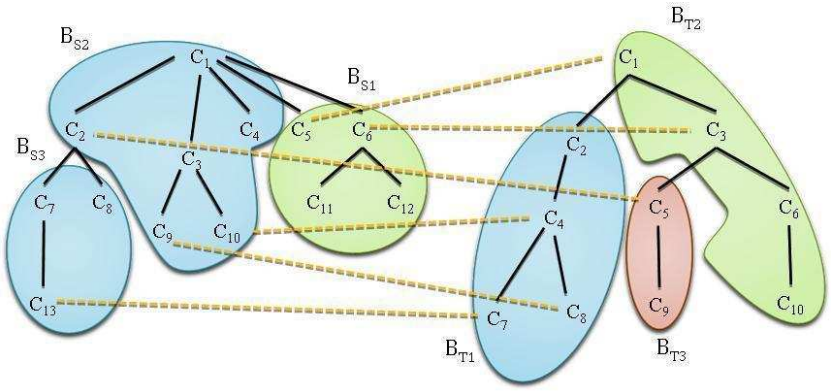

FIG. 3b Génération des blocs de la source

CHAPITRE 4

Expérimentations

Nous avons implémenté les deux méthodes présentées précédemment sous forme de modules java en réutilisant un prototype de la méthode FALCON disponible sur le web. Des expérimentations ont été faites sur différentes ontologies afin de comparer les méthodes de partitionnement et leur efficacité pour l'alignement. Les blocs constitués ont été alignés par paires à l'aide du logiciel d'alignement développé au sein de l'équipe, *TaxoMap*.

Les expérimentations ont tout d'abord été réalisées sur des ontologies dans le domaine géographique, fournies par le COGIT[1]. Ces ontologies sont bien connues des chercheurs de l'équipe et de tailles limitées, ce qui permet de les aligner directement avec *TaxoMap* sans avoir besoin de les partitionner. Ceci nous a permis d'analyser la pertinence sémantique des blocs générés et d'utiliser les résultats des alignements directs (sans partitions) comme référence pour les résultats obtenus aprés les partitionnements. D'autres expérimentations ont été faites ensuite sur une paire d'ontologies de grandes tailles.

4.1 Expérimentations sur les ontologies géographiques

La première ontologie, la *Cible* BDTopo, est composée de 612 concepts reliés par des liens de subsomption jusqu'à 7 niveaux de profondeur. La deuxième ontologie, la *Source* BDCarto, est composée de 506 concepts reliés par des liens de subsomption jusqu'à 7 niveaux de profondeur.

Les résultats de l'alignement direct (sans partitions) effectué par *TaxoMap* et qui vont nous servir de référence pour les autres alignements sont présentés dans le tableau 1 ci-dessous. Ils décomptent les appariements obtenus suivant le type de relations établies entre les concepts : relation d'équivalence (*i*sEq), de subsomption (*i*sA) ou de proximité (*i*sClose).

Ontologies	Taille Cible	Taille Source	isEq	isClose	isA	$\Sigma =$
Topo-Carto	612	505	193	24	143	360

Tab.1. Les relations identifiées par l'alignement de BDCarto vers BDTopo

Pour effectuer les partitions, nous avons fixé à 100 concepts la taille maximale des blocs fusionnables, ce qui veut dire qu'un bloc dépassant cette taille ne peut plus être fusionné mais que deux blocs comprenant au plus 99 concepts chacun pourront l'être. Les blocs générés contiennent donc au plus 198 concepts chacun. Le tableau 2 ci-dessous donne le nombre de blocs générés à partir des deux ontologies par chaque méthode.

		BDTopo 612 concepts			BDCarto 505 concepts		
Méthodes	ancres	blocs générés	concepts isolés	plus grand bloc	blocs générés	concepts isolés	plus grand bloc
FALCON	184	5	0	151	22	25	105
Méthode 1	184	5	0	151	9	16	143
Méthode 2	184	6	0	123	9	16	153

Tab.2. Partitionnement de BDTopo et BDCarto par méthode

[1]Le laboratoire COGIT (Conception Objet et Généralisation de l'Information Topographique), Institut Géographique National

Remarquons que le nombre d'ancres identifiées, 184, est plus faible que le nombre de concepts jugés équivalents par *TaxoMap*, 193. En effet, la mesure de similarité utilisée pour calculer les ancres est plus stricte (mais plus rapide à calculer), que celle de *TaxoMap*.

L'ontologie cible BDTopo est l'ontologie de référence du COGIT, elle est donc bien construite, compacte et très structurée . La racine ne comporte que deux fils directs de profondeur 1, eux-mêmes pères directs d'un nombre limité de noeuds. A l'inverse, l'ontologie source BDCarto est moins structurée, très dispersée. La racine est reliée à presque une trentaine de fils directs, et de très nombreux sous-arbres ne contiennent pas plus d'une dizaine d'éléments.

L'ontologie cible est ainsi facile à partitionner en blocs sémantiquement pertinents que ce soit par la méthode FALCON, méthode reprise dans la méthode 1 pour la décomposition de la cible et qui s'appuie essentiellement sur les relations structurelles entre concepts, ou par la méthode 2. Les deux décompositions proposées, composées respectivement de 5 et 6 blocs, ont toutes les deux leur cohérence.

La décomposition de la source, plus dispersée, est plus délicate pour FALCON. Elle génère un nombre important de petits blocs ne comprenant pas plus de 5 ou 6 concepts et dont 16 ne contiennent pas d'ancres, plus 25 concepts isolés. En utilisant l'information sur les ancres partagées, nos méthodes permettent en revanche d'aggréger à des blocs plus importants plus de la moitié de ces petits blocs ainsi que de nombreux concepts isolés, tout en gardant la cohérence sémantique de ces derniers. Le partionnement construit, moins dispersé, est donc plus lisible pour l'humain et plus efficace pour la phase suivante d'alignement des blocs.

Le choix des paires de blocs à aligner diffère suivant les méthodes :

- pour FALCON, (cf. tableau 3) les auteurs alignent les paires de blocs qui ont un ratio, nombre d'ancres partagées sur le nombre de concepts présents dans les deux blocs, supérieur à un seuil donné ϵ, ici fixé à 0.1, ce qui veut dire qu'un même bloc de la source pourra être aligné avec plusieurs blocs de la cible (ici, les paires 4 et 5 alignent le même bloc source) et qu'un bloc de la cible pourra participer à plusieurs alignements avec plusieurs blocs de la source (cas des paires 1 et 7, puis 2 et 3).

- Pour la méthode 1, un bloc de la source est aligné avec tous les blocs de la cible qui contiennent ses ancres, mais un bloc de la cible ne participe qu'à un unique alignement (cf. tableau 4 où les paires 3 et 4 alignent le même bloc source sur 2 blocs cibles distincts).

- Pour la méthode 2, les paires choisies sont celles qui maximisent le nombre d'ancres partagées d'un bloc source, qui ne participe donc qu'à un unique alignement(cf. tableau 5).

Paires	Taille Source	Taille Cible	isEq	isClose	isA	Σ (Relations) =
paire1	2	83	1	0	0	1
paire2	81	111	43	0	7	50
paire3	102	111	18	3	16	37
paire4	105	151	10	6	18	34
paire5	105	144	14	7	17	38
paire6	69	123	15	3	21	39
paire7	3	83	3	0	0	3
		Σ =	104	19	79	202

Tab.3. Les relations identifiées par l'alignement des blocs générés par la méthode FALCON

Parmi les 22 blocs générés par FALCON, seuls 6 blocs sources contiennent des ancres et ils sont chacun alignés avec le ou les blocs cibles pour lesquels le ratio d'ancres partagées dépasse le seuil limite.

Paires	Taille Source	Taille Cible	isEq	isClose	isA	Σ (Relations) =
paire1	105	123	45	4	23	72
paire2	101	111	75	0	8	83
paire3	111	83	12	3	8	23
paire4	111	151	23	5	20	48
paire5	143	144	29	7	17	53
		Σ =	184	19	76	279

Tab.4. Les relations identifiées par l'alignement des blocs générés avec la méthode 1

Parmi les 9 blocs sources générés par la méthode 1, les 4 blocs contenant les ancres (le bloc de taille 111 résulte de la fusion des blocs construits autour des centres initiés à partir des ancres des 2 blocs de la cible de taille 83 et 151) sont alignés aux 5 blocs de la cible.

Paires	Taille Source	Taille Cible	isEq	isClose	isA	Σ (Relations) =
paire1	113	111	70	1	11	82
paire2	43	105	9	1	7	17
paire3	152	123	31	9	35	75
paire4	27	62	11	1	5	17
paire5	6	110	4	0	1	5
paire6	134	101	14	4	13	31
		Σ =	139	16	72	227

Tab.5. Les relations identifiées par l'alignement des blocs générés avec la méthode 2

Même en appariant moins de paires de blocs que dans la méthode FALCON, l'appariement des blocs générés par les méthodes 1 et 2 donnent de meilleurs résultats. En effet, nos méthodes prennent en considération l'information fournie par les ancres avant d'affectuer les partitions, ce qui permet de regrouper les concepts qui ont des relations entre eux dans des blocs qui seront considérés par la suite comme des paires à aligner, alors que la méthode FALCON partitionne les ontologies indépendamment l'une de l'autre.

La méthode 1 produit les meilleurs résultats et permet en particulier, par construction, de retrouver tous les appariements correspondants aux ancres, ce que ne permet pas la méthode 2. Les résultats de cette dernière nous semble décevants, puisque la méthode était pensée pour prendre en compte dès le départ les relations partagées par les deux ontologies. La mauvaise structuration de l'ontologie source est peut être à l'origine du problème. Une analyse plus poussée des qualités relatives de ces deux méthodes demanderait beaucoup plus de temps et des expérimentations sur d'autres ontologies bien connues, de structures plus équilibrées.

Nous avons calculé les deux mesures classiquement utilisées en alignement pour comparer l'efficacité des méthodes, la *précision* (le nombre d'appariements corrects identifiés après partition par rapport au nombre total d'appariements trouvés aprés partition) et le *rappel* (le nombre d'appariements corrects identifiés après partition par rapport au nombre total d'appariements de référence directement trouvés par *TaxoMap*). Les résultats[2] sont présentés dans le tableau suivant et montrent bien l'apport de nos méthodes :

[2]Ces résultats ont été calculés automatiquement par l'API d'évaluation d'alignements disponible sur le Web, http ://oaei.ontologymatching.org/2008/align.html, en fournissant en référence le fichier généré par l'alignement direct sans partition.

Méthodes	Précision	Rappel
FALCON	0.81	0.45
Méthode 1	0.87	0.61
Méthode 2	0.88	0.57

Tab.6. Précision et Rappel par méthode

Le fait que les différentes méthodes aient une précision inférieure à 1 signifie qu'elles trouvent toutes les trois des appariements qui n'avaient pas été trouvés par l'alignement direct des deux ontologies non partitionnées. Bien que comptabilisés ici comme incorrects, ces appariements ne sont pas forcément faux. En effet, *TaxoMap* ne produit pour chaque concept de la source, qu'un unique appariement avec un seul concept de la cible, celui qu'il juge le meilleur, même si plusieurs concepts de la cible pouvaient en être rapprochés. Si les deux concepts intervenant dans un appariement de référence ne sont plus comparés entre eux parce qu'ils sont répartis dans des blocs non alignés, un autre appariement, qui ne sera pas forcément inintéressant, peut être trouvé pour le concept source. Nous n'avons pas eu le temps d'examiner la qualité de ces nouveaux mappings.

4.2 Expérimentations sur les ontologies de grande taille

Nous avons testé les méthodes 1 et 2 sur deux ontologies de grande taille, AGROVOC et NALT, qui sont composées respectivement de 28 439 et 42 326 concepts et sont utilisées comme ontologies de référence dans le challenge OAEI'08 (*Ontology Alignment Evaluation Initiative 2008 Campaign*) qui fait concourir chaque année les outils d'alignement sur des couples d'ontologies de taille et de domaine variés. AGROVOC est une ontologie multilingue construite par la FAO (Food and Agriculture Organization) et qui couvre les domaines de l'agriculture, les forêts, la pêche, l'environnement et l'alimentation. NALT est le thésaurus de la NAL (National Agricultural Library) sur les mêmes domaines.

Nous avons choisi l'ontologie NALT, la plus importante, comme cible, l'ontologie AGROVOC comme source, et fixé la taille maximale des blocs fusionnables à 2 000 concepts puisque *TaxoMap* n'est plus du tout efficace à partir de 4 000 concepts.

Le temps d'exécution de l'alignement d'une paire de blocs par *TaxoMap* pouvant être important (plusieurs heures quand les blocs contiennent plusieurs milliers de concepts), nous n'avons pas pu tester la méthode FALCON sur ces ontologies de grande taille, puisque le processus de choix des paires de blocs à aligner dans FALCON demande de calculer les ancres partagées par toutes les combinaisons possibles de paires de blocs, puis multiplie les alignements entre les paires pour maximiser le nombre d'appariements. Le temps total nécessaire pour cette expérimentation était donc bien plus important que pour nos méthodes.

Le tableau ci-dessous donne le nombre de blocs générés pour les deux ontologies :

		AGROVOC			NALT		
Méthodes	ancres	blocs générés	concepts isolés	plus grand bloc	blocs générés	concepts isolés	plus grand bloc
FALCON	14 787	318	492	2830	47	4	3356
Méthode 1	14 787	220	199	2939	47	4	3356
Méthode 2	14 787	95	199	3 534	47	4	3118

Tab.7. Partitionnement de AGROVOC et NALT

L'alignement se fait seulement sur les paires de blocs générés par les partitions puisque *TaxoMap* ne peut pas aligner d'ontologies dépassant les 4 000 concepts. Nous n'avons donc pas cette fois-ci

d'alignement de référence. Les tableaux 8 et 9 suivants présentent les différentes relations trouvées pour chaque méthode.

On remarquera que dans cette expérimentation, toutes les ancres identifiées au départ ne se retrouvent pas, et de loin, dans les relations d'équivalence identifiées par la méthode 1. Ce résultat surprenant a priori, s'explique par l'utilisation dans AGROVOC et NALT, de labels multiples pour chaque concept. Par exemple, le concept *Caribbean* de NALT dispose de 3 autres labels, *Leeward Islands, Windward Island,* et *Lesser Antilles* qui tous désignent le même concept, alors que dans AGROVOC, ces 3 labels correspondent à 3 concepts distincts.

Le problème vient du traitement (défectueux !) des concepts à labels multiples dans *TaxoMap*. Celui-ci compare les différents labels d'un concept à tous ceux des concepts de l'autre ontologie, puis fixe pour le concept un label "officiel", celui pour lequel il a trouvé la plus forte similarité lexicale avec un label de l'autre ontologie. Les 3 labels de *Caribbean* ayant tous les 3 leur équivalent dans l'ontologie source, *TaxoMap* a choisi le premier label et n'a plus pris en compte les 2 autres. De ce fait, nous avions identifiés 3 ancres, mais nous ne retrouvons plus qu'un seul appariement avec *TaxoMap* !

Paires	Taille Source	Taille Cible	isEq	isClose	isA	Σ (Relations) =
paire1	851	2 371	679	10	96	785
paire2	2 794	408	182	17	126	325
paire3	2 794	3 356	962	54	325	1 341
paire4	2 939	2 581	888	21	420	1 329
paire5	2 939	2 392	648	28	571	1 247
paire6	2 200	2 465	1 335	5	331	1 671
paire7	2 199	2 098	731	85	363	1179
paire8	2 199	3 020	1 056	90	397	1 543
paire9	2 021	3 170	1 163	46	510	1 719
paire10	2	5	2	0	0	2
paire11	2 586	2 686	1 099	131	490	1 720
paire12	2 586	2 417	838	133	535	1 506
paire13	2 724	231	329	14	118	461
paire14	2 724	2 182	725	29	400	1 154
paire15	2 724	2 306	953	24	313	1 290
paire16	864	1 361	712	17	52	781
		Σ =	12 302	704	5 047	18 053

Tab.8. Les relations identifiées par l'alignement des blocs générés avec la méthode 1

On remarquera aussi qu'un même bloc source pouvant être aligné par la méthode 1 avec plusieurs blocs cibles, on peut trouver pour un bloc source, un nombre d'appariements supérieurs à la taille du bloc (exemple les paires 13, 14 et 15 alignent le même bloc de 2 724 concepts avec 3 blocs distincts et totalisent 2 905 appariements). Ce qui signifie que plusieurs concepts du même bloc ont été alignés différemment dans les différents blocs cibles. Ici encore, nous n'avons pas eu le temps de juger de la validité de ces appariements en surnombre. Mais ils seront tous envoyés à la fin du mois au challenge OAEI'08 où l'équipe est vraiment contente de pouvoir concourir cette année sur des ontologies aussi volumineuses !

Paires	Taille Source	Taille Cible	isEq	isClose	isA	Σ (Relations) =
paire1	1 157	2 861	518	213	154	885
paire2	2 143	2 102	414	187	395	996
paire3	1 501	2 214	930	61	132	1 123
paire4	889	2 148	299	215	73	587
paire5	72	2 391	55	5	5	65
paire6	686	2 011	413	76	78	567
paire7	661	2 143	527	41	27	595
paire8	2 064	2 556	852	99	227	1 178
paire9	2 242	3 118	443	328	360	1 131
paire10	2	5	2	0	0	2
paire11	626	231	317	36	35	388
paire12	7	43	3	0	0	3
paire13	415	2 182	337	32	16	385
paire14	2 693	3 118	286	396	509	1 191
paire15	419	2 102	247	43	61	351
paire16	121	535	69	2	5	76
paire17	298	1 441	61	143	32	236
paire18	56	2 556	41	2	5	48
paire19	865	1361	712	44	26	782
paire20	3 534	3 118	214	577	726	1 517
paire21	351	1 542	104	11	66	181
paire22	2 051	2 703	913	95	187	1 195
paire23	2 060	2 371	701	84	206	991
paire24	72	55	22	2	2	26
paire25	1 652	3 118	84	216	243	543
		Σ =	8 564	2 908	3 570	15 042

Tab.9. Les relations identifiées par l'alignement des blocs générés avec la méthode 2

Ces résultats montrent qu'on peut aligner des ontologies de grande taille si on introduit au début du processus d'alignement une étape de partitionnement, et que cette dernière est plus efficace si elle prend en compte les contraintes de la tâche d'alignement, i.e. le fait qu'elle doit s'appliquer à deux ontologies.

Chapitre 5

Conclusion et perspectives

(A compléter)

Bibliographie

[1] Evren Sirin Bernardo Cuenca Grau, Bijan Parsia and Aditya Kalyanpur. Automatic partitioning of owl ontologie using e-connections. In *DL 2005, Proceedings of 18th International Workshop on Description Logics, Edinburgh, UK*, 2005.

[2] Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings of KR2006 : the 20th International Conference on Principles of Knowledge Representation and Reasoning, Lake District, UK, June 2–5, 2006*, pages 198–209, 2006.

[3] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK : A robust clustering algorithm for categorical attributes. *Information Systems*, 25 :345–366, 2000.

[4] Hassen Kefi, Chantal Reynaud, and Brigitte Safar. Techniques structurelles pour l'alignement de taxonomies sur le web. In *Atelier Fouille du Web - EGC 2006*, 2006.

[5] Natalya Fridman Noy and Mark A. Musen. PROMPT : Algorithm and tool for automated ontology merging and alignment. In *AAAI/IAAI*, pages 450–455, 2000.

[6] Shvaiko Pavel and Euzenat Jérôme. A survey of schema-based matching approaches. *Journal on Data Semantics IV*, pages 146–171, 2005.

[7] Heiner Stuckenschmidt and Michel Klein Vrije. Structured-based partitioning of large concept hierarchies. In *The Semantic Web - ISWC.*, pages 289–303, June 2004.

[8] James Hendler Tim Berners-Lee and Ora Lassila. The semantic web : A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. by tim berners-lee, james hendler and ora lassila. scientific american. In *The Scientific American 284(5)*, pages 34–43, May 2001.

[9] Yuanyuan Zhao Wei Hu and Yuzhong Qu. Partition-based block matching of large class hierarchies. In *The Semantic Web - ASWC*, pages 72–83, 2006.