DECOR

Passage à l'échelle des techniques de découverte de correspondances

Workshop EGC 2007

Jérôme Euzenat, Jean-Marc Petit, Marie-Christine Rousset

Mardi 23 janvier 2007
Namur, Belgique

Contents

Préface

Sur un thème très porteur en bases de données et en intelligence artificielle, cet atelier vise à faire le point des travaux menés par la communauté française sur la découverte de correspondances entre schémas ou ontologies. Ces schémas ou ontologies peuvent décrire des bases de données, des documents ou tout ou partie d'un domaine scientifique. Ces schémas et/ou ontologies peuvent être exprimés dans différents langages, plus ou moins formalisés, offrant ainsi une grande diversité dans l'expression des schémas et des ontologies. Ainsi, les correspondances sont elles aussi très variées et dépendent du contexte dans lequel elles s'expriment. De nombreuses techniques de découverte de correspondances ont été proposées dans le passé, sans forcément se soucier *du passage à l'échelle* des techniques sous-jacentes.

Pour cet atelier, un accent particulier a donc été mis sur ce point afin de mieux comprendre les verrous permettant le déploiement à grande échelle de ces techniques, par exemple à l'échelle du futur Web sémantique.

Avec l'objectif d'ouvrir l'atelier aux travaux en cours sur cette thématique, nous avons décidé d'organiser le processus de relecture comme suit : soumission des articles sur la base de résumés limités à 2 pages, puis réception des versions étendues pour l'insertion dans les actes de l'atelier sur la base des remarques faites aux auteurs.

Dix exposés ont été programmés pour cet atelier d'une journée, une place importante étant néanmoins laissée aux questions et aux discussions pour permettre un réel échange entre les participants.

Jérôme Euzenat (INRIA Rhône-Alpes-LIG, Montbonnot),
Jean-Marc Petit (LIRIS, INSA-Lyon),
Marie-Christine Rousset (LIG, Grenoble)

Benchmark d'algorithmes de matching de schémas XML Pour l'automatisation du tuning

Mohamed Boukhebouze*, Rami Rifaieh**, Nabila Benharkat*, Youssef Amghar*
* LIRIS, National Institute of Applied Sciences of Lyon, Lyon, France. Bat. Blaise Pascal 7, av. Jean Capelle 69621 Villeurbanne **San Diego Supercomputer Center, University of California San Diego, USA
{mohamed.boukhebouze, nabila.benharkat, youssef.amghar}@insa-lyon.fr
rrifaieh@sdsc.edu

Résumé. Plusieurs algorithmes de matching ont été proposés en vue d'automatiser ou semi automatiser le processus de découverte des correspondances entre schémas de données différents. Ces algorithmes utilisent différentes techniques. En général, le résultat final combine les différents résultats issus de ces techniques. Souvent ces algorithmes possèdent de nombreux paramètres à ajuster manuellement. Cette tâche, généralement effectuée par des experts humains, impose une maîtrise parfaite de ces systèmes. Notre solution vise à automatiser le tuning des différents paramètres structurels des systèmes de matching de schémas XML afin de réduire l'intervention humaine et aussi d'offrir une bonne précision. Nous avons donc proposé un benchmark qui détermine les meilleures valeurs de paramètres qui servent à calculer la similarité structurelle des outils étudiés en fonction des schémas d'entrée. Puis, nous avons proposé un algorithme pour l'automatisation du choix des meilleures valeurs pour ces paramètres en se basant sur les règles déduites du benchmark.

1 Etat de l'art :

Dans la littérature, on rencontre plusieurs types d'algorithmes de matching. Toutefois, nous nous sommes intéressés plus particulièrement aux approches qui traitent les schémas de données comme COMA (Do et Rahm, 2002), EXSMAL (Rifaieh et al, 2005) et SCIA (Wang, 2004). Dans la littérature actuelle, il existe très peu de solutions pour l'automatisation du tuning. Nous avons répertorié deux travaux : i) Berkovsky (2005) qui utilise les algorithmes génétiques pour le tuning automatique. Notons que le succès des algorithmes génétiques pour trouver une solution optimale dépend fortement d'une fonction physique qui emploie une combinaison de fonctions prédéfinies. Cette solution ne peut pas être appliquée à notre approche puisque nous voulons découvrir ces fonctions entre les paramètres structurels. ii) L'approche de Sayyadian (2005) propose un système de tuning automatique nommé eTuner. Cependant, son algorithme exige le passage par deux étapes (la perturbation des schémas et l'exécution pour effectuer le tuning), et cela engendre un coût considérable en temps.

2 L’approche proposée :

L'étude des différents systèmes de matching dans la partie de l'état de l'art nous a permis de résumer les similitudes et dissimilitudes entre des systèmes de matching. Pour essayer d'automatiser le tuning des différents paramètres des systèmes de matching appliqués aux schémas XML nous avons proposé une solution qui consiste à déterminer les meilleurs paramètres à utiliser suivant le schéma d'entrée d'une façon empirique en lançant une série de tests (benchmark) pour déduire une règle générale de tuning à appliquer pour tous les systèmes de matching.

L’approche proposée est divisée en deux grandes étapes comme le montre la figure 1 : La première étape consiste à effectuer un benchmark. En effet, ce dernier a comme but de déduire empiriquement la relation qui existe entre les caractéristiques des schémas d'entrée et les paramètres structurels des outils étudiés. La deuxième étape consiste à exécuter un algorithme pour l’optimisation des paramètres structurels des outils étudiés et cela en utilisant les relations trouvées dans la première étape.

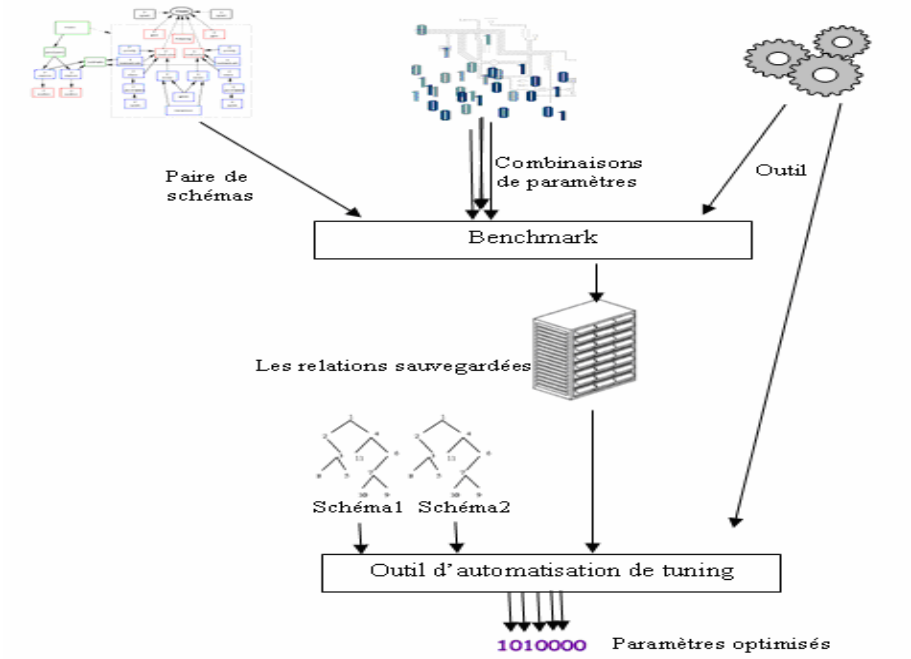


Figure1. Les étapes de L’APPROCHE *proposée*

2.1 Etapes du benchmark :

Nous allons, pour chaque outil choisi et pour chaque paire de schémas choisie, calculer l'ensemble de combinaisons de valeurs des paramètres structurels en incrémentant chacun d'eux d'un certain delta. Nous calculons, ensuite, la précision, le Recall et l'Overall, après avoir effectué un matching automatique pour chaque combinaison de valeurs des paramètres. L'étape suivante consiste à ne garder que les combinaisons de paramètres qui donnent une meilleure précision, le meilleur Recall, et le meilleur Overall puis à appliquer une méthode d'analyse de données. Cette dernière devra nous permettre de trouver une relation entre coefficients structurels des divers "matchers" testés. Pour valider, à la fin de cette phase, la relation trouvée nous allons effectuer une série de tests de signification comme par exemple *le test d'hypothèse* en appliquant le test de *Fisher*. Si les tests sont valides nous sauvegardons les modèles trouvés dans une base, dans le cas contraire nous chercherons à appliquer d'autres méthodes d'analyse de données.

2.1.1 Le choix des schémas de test utilisés :

Les schémas XML utilisés dans la plupart des évaluations des systèmes de matching se caractérisent par des nombres de chemins différents, des nombres d'éléments différents et des profondeurs maximales différentes, pour cela, nous avons choisi ces trois caractéristiques structurelles pour les schémas de test utilisés dans notre benchmark. Ces schémas choisis sont des schémas réels qui appartiennent au même domaine (bibliographique). En effet, nous avons pris 20 schémas ayant un nombre d'éléments et un nombre de chemins compris entre]0,80], et ayant une profondeur comprise entre [2 ,7], au total 190 paires de schémas possibles.

2.1.2 Les outils étudiés, et leur positionnement :

Parmi les algorithmes de matching, nous avons choisi de travailler avec COMA (Do et Rahm, 2002), EXSMAL (Rifaieh et al, 2005) et SCIA (Wang, 2004) pour le Benchmark, car d'une part nous avons pu récupérer les exécutables des ces derniers, d'autre part, ils apparaissent dans la même catégorie selon la classification de Euzenat et al (2005)

2.1.3 Le calcul de matching et mesures de performances :

Pour pouvoir évaluer la qualité des résultats de matching, cette tâche doit être réalisée d'abord manuellement. Les résultats obtenus sont employés comme standard pour évaluer la qualité du résultat déterminé automatiquement par le système de matching. Pour cela nous avons utilisé les mesures de qualité définies dans la partie "état de l'art" qui sont : la précision, le Recall et l'Overall. En effet, la précision et le Recall proviennent du domaine de la recherche documentaire mais ils sont également employés dans les évaluations des systèmes de matching (Aumueller et al, 2005).

2.1.4 L’analyse des résultats obtenus:

Dans le but d’étudier les résultats obtenus, nous avons choisi parmi les techniques d’analyse de données (Jolliffe, 487p) utilisées la régression linéaire qui semble être la meilleure manière de trouver une relation entre les paramètres et cela en appliquant la méthode des moindres carrés pour trouver les coefficients de régression. Cependant, persiste un problème : comment juger qu’il y a une relation significative entre les paramètres ? Pour remédier à ce problème nous avons effectué, pour chaque relation trouvée, une analyse des résidus pour tester la validité d'un modèle de régression, puis nous avons calculé le coefficient de détermination pour mieux apprécier la contribution de la variable explicative dans l’équation de régression, et finalement effectuer un test de Fisher pour évaluer la signification de la relation.

2.2 L’algorithme d’automatisation du tuning :

Le but de la deuxième étape de notre proposition consiste à automatiser la tâche d’ajustement des différents paramètres structurels des systèmes de matching étudiés. Pour cela, nous avons proposé un algorithme d’automatisation du tuning qui utilise les résultats de benchmark. L’algorithme d’automatisation du tuning a comme entrée deux schémas et le nom de l’outil pour lequel nous voulons optimiser les paramètres structurels. L’algorithme va d’abord calculer le nombre d’éléments, nombre de chemins et la profondeur de chaque schéma. Ensuite il va choisir, parmi les relations sauvegardées après l’exécution du benchmark, la relation à appliquer qui correspond aux caractéristiques des schémas. Finalement l’algorithme calcule et affiche les valeurs suggérées.

Dans le but d’évaluer la précision de cet outil nous avons mené une expérience qui consiste à prendre COMA pour effectuer un maching entre deux schémas donnés. Nous avons calculé la précision et le Recall des résultats obtenus. Nous avons refait la même opération après avoir ajusté les paramètres de COMA en utilisant les valeurs suggérées. Nous pouvons déduire que notre outil a permis de trouver des valeurs optimales pour les paramètres structurels de COMA, suivant les caractéristiques des schémas d’entrée. L’utilisation de ces paramètres a permis l’obtention d’une meilleure précision et d’un meilleur Recall.

3 Conclusion et perspectives :

Dans le but d’automatiser le tuning des différents paramètres des systèmes de matching, nous avons proposé une approche qui comporte deux étapes : un benchmark évolutif et un algorithme de tuning automatique.

Nous pensons que si notre proposition contribue à résoudre le problème de tuning automatique des paramètres structurels des algorithmes de matching (appliqués aux schémas XML), il n’en demeure pas moins qu’il y a des points à améliorer. Nous estimons qu’une solution robuste pour ce problème devrait prendre en considération, en supplément, le tuning

des paramètres linguistiques et de prendre en considération aussi le tuning des algorithmes de matching appliqués aux schémas relationnels.

4 Bibliographie

H.H.Do, E.Rahm (2002). COMA - A System for Flexible Combination of Schema Matching Approaches. VLDB 2002: 610-621

R.Rifaieh, U.Chukmol, A.N.Benharkat (2005). A Matching Algorithm for Electronic Data Interchange. TES 2005: 34-47

G.Wang, Y.W.Nam, and K.Lin (2004). Critical Points for Interactive Schema Matching. Technical Report CS2004-0779, UCSD Department of Computer Science, January 2004;

S.Berkovsky, Y.Eytani (2005). Measuring the Relative Performance of Schema Matchers. Proceedings

of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)

M.Sayyadian, Y.Lee, A.H.Doan, A.Rosenthal (2005). Tuning Schema Matching Software using Synthetic Scenarios. VLDB 2005: 994-1005.

P.Shvaiko, and J.Euzenat (2005). A Survey of Schema-based Matching Approaches. Technical Report DIT-04-087, Informatica e Telecomunicazioni, University of Trento

D.Aumueller, H.H.Do, S.Massmann, E.Rahm (2005). Schema and ontology matching with COMA++. SIGMOD Conference 2005: 906-908

I.T. Jolliffe. Principal Component Analysis. Springer (ed), ISBN: 0387954422, 487p

5 Summary

Several matching algorithms were recently developed in order to automate or semi-automate the process of correspondences discovery between XML schemas. These algorithms use a wide range of approaches and matching techniques covering linguistic similarity, structural similarity, constraints, etc. The final matching combines arithmetically different results stemmed from these techniques. The aggregation of the results uses often many parameters and weights to be adjusted manually. Generally, this task is achieved by human experts and requires a perfect understanding of the matching algorithm. In order to reduce the human intervention and improve matching quality, we suggest automating the tuning of the various structural parameters used within XML-Schema matching algorithms. In this work, we offer a benchmark, for

three tools, that seeks mathematical relations between parameters values and schema topology. In consequent, we propose an algorithm for the tuning of these parameters for studied tools.

Découverte de correspondances sémantiques par inférences dans un environnement P2P

Lionel Médini, Nicolas Lumineau, Catarina Ferreira da Silva,
Patrick Hoffmann, Parisa Ghodous

Université de Lyon, Lyon, F-69003, France ; université Lyon 1, CNRS UMR5205, LIRIS, Villeurbanne, F-69622, France.
{prenom.nom}@liris.cnrs.fr
http://liris.cnrs.fr/prenom.nom/

Résumé. Cet article présente une approche de découverte automatique de correspondances sémantiques dans un réseau pair à pair en utilisant des méthodes sémantiques fondées sur des logiques de description. Pour cela, chaque pair inclut un moteur d'inférences et collabore avec les autres pairs pour échanger des parties d'ontologies et des axiomes nécessaires à la découverte de nouvelles correspondances. Une particularité de cette approche est qu'elle permet un enrichissement des axiomes en utilisant la connaissance et les correspondances relatives aux voisins communs de deux pairs cherchant à s'aligner sémantiquement. Notre objectif principal est de réduire le temps de calcul lié à l'utilisation du moteur d'inférences. Pour cela, nous explorons plusieurs stratégies complémentaires, notamment pour réduire la taille des ontologies à traiter, et distribuer les opérations à réaliser sur les différents pairs.

1 Introduction

De plus en plus d'applications industrielles accessibles sur le Web ou par d'autres réseaux, offrent des services fondés sur le partage et l'interopérabilité des structures de connaissances issues de diverses organisations, et représentant divers aspects d'un domaine modélisé. Plusieurs approches existent pour mettre en œuvre cette interopérabilité. Parmi ces approches, on distingue l'alignement d'ontologies, qui consiste à identifier des relations entre éléments de différentes ontologies. Euzenat *et al.* (2004) identifient différents types de méthodes d'alignement d'ontologies : terminologiques, structurelles, extensionnelles et sémantiques, qui proviennent de disciplines variées, telles que la fouille de données, les sciences du langage, les statistiques ou la représentation des connaissances. Nous nous intéressons ici à la mise en correspondance d'ontologies à l'aide de méthodes sémantiques fondées sur des logiques de description. De telles méthodes permettent, à l'aide d'une représentation formelle des concepts des ontologies et d'un

moteur d'inférences, de découvrir automatiquement des correspondances sémantiques entre concepts de deux ontologies.

Par ailleurs, les systèmes pair à pair (P2P) sont devenus des solutions incontournables pour permettre le passage à l'échelle des applications (Sigmod record, 2003). Il nous a semblé intéressant d'étendre l'approche de découverte de correspondances sémantiques afin de l'adapter aux environnements P2P. Les verrous scientifiques liés à l'utilisation de notre approche dans un environnement P2P sont principalement liés au volume total d'informations manipulées et à l'impossibilité de stocker une quelconque information de manière centralisée[1]. Il est également nécessaire de considérer l'évolution des correspondances découvertes face aux connexions et déconnexions des pairs. Enfin, il est possible de tirer partie du paradigme P2P pour améliorer le fonctionnement de cette approche, en termes d'efficacité et de nombre de correspondances découvertes.

La suite de l'article est organisé comme suit : la Section 2 présente les différents travaux concernant la découverte de correspondances sémantiques et des systèmes P2P. Dans la Section 3, nous présentons notre approche de traitement collaboratif des ontologies. La Section 4 présente brièvement l'architecture des pairs permettant la mise en œuvre de cette approche. La section 5 présente une conclusion et quelques perspectives.

2 Travaux connexes

Les travaux présentés dans cet article concernent la découverte de correspondances en environnement pair à pair à l'aide de méthodes sémantiques. Nous présentons successivement dans cette section le principe des méthodes sémantiques, celui du pair à pair, puis des exemples de systèmes réalisant la découverte de correspondances sémantiques en environnement pair à pair.

2.1 Découverte de correspondances à l'aide de méthodes sémantiques

De manière générale, les méthodes sémantiques permettent d'effectuer des déductions par l'application de techniques de raisonnement, lesquelles s'appuient sur des formalismes de représentation des connaissances fondées sur des logiques formelles. Parmi ces méthodes sémantiques, on trouve les techniques de satisfiabilité propositionnelle (Giunchiglia *et al.*, 2004) ou celles fondées sur les Logiques de Description (LD) : Baader *et al.* (2003). Les premières permettent notamment de retranscrire les problèmes d'appariement d'arbres – présentés sous la forme de deux hiérarchies de concepts et d'un ensemble de correspondances – en logique propositionnelle et d'en vérifier la validité. Les secondes sont capables d'appliquer les algorithmes de subsomption inclus dans certains moteurs d'inférences pour déduire des correspondances entre des concepts d'ontologies, à partir de leurs représentations formelles.

[1] Nous nous plaçons ici dans un contexte où chaque pair a pour objectif de collaborer dans un projet industriel, en y apportant des connaissances validées, et ce, dans un but constructif. Par conséquent, nous nous affranchissons ici des problèmes liés à la sécurité, la confiance et la malveillance pouvant survenir sur des réseaux pair à pair classiques.

En combinant ces deux types de méthodes, il est possible de découvrir automatiquement des correspondances sémantiques entre ontologies, à partir d'un ensemble d'axiomes[2] initiaux (Bouquet *et al.*, 2004). Ce processus, qui teste en fait la subsomption de toutes les paires de concepts des deux ontologies d'entrée, est décrit en détail dans Ferreira Da Silva *et al.* (2006). Il permet d'identifier quatre types de correspondances : (i) : subsomption : $C_1 \sqsubseteq C_2$; (ii) : équivalence : $C_1 \sqsubseteq C_2$ et $C_2 \sqsubseteq C_1$; (iii) : conjonction : $C_1 \sqcap C_2 \sqsubseteq C_1 \sqcap C_3$; (iv) : transitivité : $C_1 \sqsubseteq C_2$ et $C_2 \sqsubseteq C_3 . \Rightarrow C_1 \sqsubseteq C_3$.

Un module de découverte de correspondances sémantiques a été implanté dans le moteur ONDIL (Le Duc, 2004). ONDIL est fondé sur le moteur d'inférences FaCT[3].En pratique, à partir de quatre ontologies du domaine de la construction (contenant plus de 20 000 concepts et relations), et d'environ cinq axiomes reliant chaque paire d'ontologies, environ 100 correspondances ont été obtenues.

Suite à l'exploitation de cet outil, Ferreira Da Silva *et al.* (2006) établit que : (i) cette méthode ne nécessite qu'une faible intervention des experts humains, limitée à l'élaboration des axiomes et à la validation des résultats par rapport aux ontologies et aux axiomes de départ ; (ii) dans la mesure où l'algorithme des tableaux, implanté dans le moteur d'inférence, a été prouvé décidable et complet par Baader et Sattler (2001), les correspondances sémantiques obtenues peuvent être considérées comme exactes ; (iii) le processus de découverte de correspondances impose un temps de calcul relativement important (de l'ordre de plusieurs minutes), ce qui pose problème pour la fourniture d'un service de découverte de correspondances sémantiques en temps réel.

2.2 Pair à Pair

Les architectures pair à pair se présentent actuellement comme une solution viable pour permettre le partage de ressources à l'échelle de l'Internet. En effet, aussi bien d'un point de vue commercial que scientifique, les architectures pair à pair suscitent un véritable engouement. Le paradigme du P2P garantit un fonctionnement à large échelle (Oram, 2001). Un très grand nombre de pairs peut interagir dans le réseau, de manière à permettre le partage d'une grande quantité de ressources. Aussi appelé d'égal à égal, chaque participant à un système P2P peut être à la fois client et serveur. Le fonctionnement du système ne repose sur aucune coordination centralisée. Ainsi, le comportement global du réseau résulte uniquement des interactions locales entre les pairs qui se connectent et se déconnectent.

On distingue communément trois types d'architectures P2P. Les architectures non structurées basées sur le tout distribué et respectant entièrement le paradigme P2P, les architectures hybrides

[2] Les axiomes sont des relations binaires entre deux concepts, et servent à initier le processus de découverte automatique ; ils sont en général fournis par les experts, ou peuvent être découverts à l'aide d'autres types de méthodes d'alignement.

[3] Fast Classification of Terminologies : Horrocks (1997). FaCT a été conçu pour évaluer la faisabilité de l'utilisation d'algorithmes de subsomption optimaux basés sur la technique des tableaux. Il permet de raisonner sur des descriptions de concepts, de rôles et d'attributs, et de maintenir une hiérarchie de concepts fondée sur les relations de subsomption.

qui hiérarchisent fonctionnellement les pairs selon leur capacité et les architectures structurées basées sur un index distribué des données (DHT : Stoica, 2001).

On distingue de nombreuses applications P2P comme (a) des systèmes de partage de fichiers comme Gnutella[4], (b) des moteurs de recherche P2P, (c) des messageries instantanées comme ICQ[5], (d) des plateformes de calcul distribué intensif tel Seti@home[6], (e) des plateformes collaboratives comme Oceanstore (Kubiatowicz, 2000) ou Ivy (Muthitacharoen, 2001) et (f) des systèmes de partage de données définies selon un schéma ou une ontologie Edutella (Nejdl, 2002).

2.3 Gestion de correspondances en Pair à Pair

Que ce soit au niveau d'un schéma ou d'une ontologie, de nombreuses solutions basées sur des correspondances sémantiques entre les pairs ont été développées. Différents procédés sont utilisés pour générer ces correspondances de manière plus ou moins automatique.

L'approche PIAZZA (Tatarinov, 2003) remplace un simple schéma logique de système d'intégration de données par un ensemble de correspondances sémantiques entre les schémas de pairs. Les correspondances entre les schémas sont spécifiées *via* une extension de Datalog. Ces travaux se focalisent sur la nature distribuée des correspondances entre schémas définis sur différents pairs et la réécriture de requêtes. Dans SomeWhere (Adjiman, 2005), un pair se connectant au réseau construit les correspondances entre sa propre ontologie et les ontologies des pairs servant de point d'entrée dans le réseau. Pour traiter une requête, l'utilisateur doit choisir le pair par lequel sa requête sera initiée dans le réseau. Le routage des requêtes est guidé vers les pairs dont les correspondances sont pertinentes. Cette pertinence est établie selon un algorithme distribué de raisonnement en logique des propositions. Le projet Hyperion (Kementsietsidis, 2003) propose la construction semi-automatique de tables de correspondances entre les schémas définis sur des pairs distants. Ces correspondances sont définies via des règles. L'utilisation de triggers permet de maintenir la cohérence de ces règles en fonction de l'évolution des données stockées dans les SGBD. Une approche agent est proposée dans PeerDB (Ooi, 2003).

3 Traitement collaboratif des ontologies

Cette partie présente le fonctionnement de notre approche de découverte de correspondances fondée sur des méthodes sémantiques, dans un réseau pair à pair. Pour la mettre en œuvre, chacun des pairs est structuré comme décrit en section 4 : il incorpore notamment une ontologie locale, des listes de correspondances avec certains de ses voisins et un moteur d'inférences. Le moteur d'inférences réalise la détection de correspondances en utilisant l'algorithme des tableaux, et à partir des deux ontologies à aligner et d'un ensemble d'axiomes, formalisées en OWL-DL.

[4] Gnutella : http://www.gnutella.org
[5] ICQ : http://www.icq.com
[6] Seti@home : http://setiathome.ssl.berkeley.edu/

Notre approche se fonde sur une architecture P2P non structurée, afin de garantir l'autonomie des pairs et permettre l'échange d'information de structure et de sémantique complexes. Elle consiste à exploiter des moteurs d'inférences dans le processus de découverte des correspondances sémantiques entre les pairs et leur voisinage. L'exploitation de ces moteurs d'inférences dans un contexte de large échelle induit certaines contraintes, principalement liées au coût du processus de découverte des correspondances.

L'originalité de l'algorithme présenté ci-dessous est de mettre en œuvre une stratégie de partage de correspondances entre les pairs, et de leur permettre de collaborer pour découvrir des correspondances en utilisant la connaissance dont dispose leur voisinage. Pour cela, il s'appuie sur deux stratégies complémentaires. D'une part, afin de réduire le nombre de concepts en entrée du moteur d'inférences associé à un pair, nous appliquons une politique de *segmentation* des ontologies, et ne fournit au moteur que des sous-ensembles d'ontologies exploitables pour la découverte. D'autre part, afin d'augmenter le nombre d'axiomes fournis au moteur d'inférences pour la découverte, nous proposons l'utilisation d'une approche par *composition*, qui permet de découvrir des correspondances entre deux pairs « à travers » un troisième pair du réseau, voisin commun des deux précédents.

3.1 Principe de découverte de correspondances

Pour illustrer notre processus de découverte de correspondances, nous prenons l'exemple d'un pair A souhaitant connaître ses correspondances avec un pair C. On suppose qu'il existe un pair B, qui est un « voisin logique » de A, et qui possède éventuellement des correspondances avec C. Ainsi, le processus général de découverte des correspondances de A vers C se décompose en trois étapes :

Génération des axiomes : les axiomes sont obtenus à l'aide de deux processus distincts : l'identification des correspondances existantes entre A et C, et l'ajout de connaissances relatives au pair B : Dans le premier cas, A demande à C sa liste de correspondances vers A[7] ; C renvoie à A une liste, éventuellement vide, que A importe, en l'ajoutant à sa propre liste ; cette liste complétée constitue la première partie des axiomes. Dans le second cas, A envoie à B sa liste de correspondances vers B ; A demande à B de lui renvoyer des axiomes entre A et B, *via* le processus de composition (voir plus loin) ; A reçoit la réponse de B et ajoute d'éventuelles informations complémentaires aux axiomes précédents.

Génération des ontologies : le processus de détection de correspondances par le moteur d'inférences sera lancé par A sur des parties des ontologies locales de A et de C, obtenues par segmentation (voir plus loin) et non sur leurs totalités. Pour cela, A met en œuvre un processus de segmentation sur son ontologie locale, en fonction des concepts apparaissant dans les axiomes. D'autre part, A envoie à C la liste des URI des concepts de C apparaissant dans les axiomes, et lui demande de mettre en œuvre une segmentation de son ontologie locale. C met en œuvre ce processus de segmentation et renvoie à A la partie minimale de son ontologie locale.

[7] L'absence d'hypothèse sur la symétrie des listes de correspondances nécessite que les listes des deux pairs concernés par ces correspondances soient fusionnées avant chaque traitement des correspondances, d'où cette requête de A à C.

Découverte de correspondances : A met en œuvre la recherche de correspondances sémantiques à l'aide de son moteur d'inférences, en utilisant les axiomes identifiés, et sur les parties des ontologies de A et de C concernées.

3.2 Segmentation

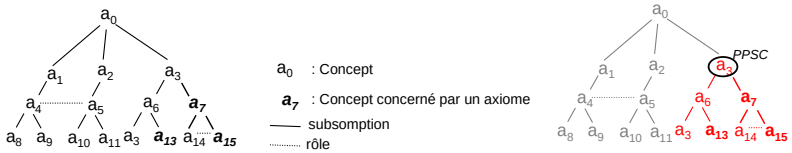
L'objectif est ici de minimiser la quantité d'informations fournies au moteur d'inférences lors d'une recherche de correspondances, afin de diminuer le temps de calcul. Il s'agit pour cela de réduire autant que possible les tailles des ontologies d'entrée, en ne communiquant pas au moteur d'inférences les parties d'une ontologie dans lesquelles aucune correspondance ne peut être découverte. De telles parties n'ont pas de lien avec les axiomes et n'apparaissent pas non plus (sous forme de rôles) dans les définitions des concepts des sous-parties en lien avec ces axiomes. Inversement, aucun rôle dans les définitions de leurs concepts ne pointe vers des concepts apparaissant dans les axiomes ou appartenant à l'autre ontologie à aligner. Il est à noter que cette stratégie est indépendante du paradigme P2P, et peut être utilisée pour la découverte de correspondances à l'aide de méthodes sémantiques dans d'autres contextes.

L'algorithme mis en œuvre fonctionne de la façon suivante : il reçoit en entrée un ensemble d'URI de concepts de l'ontologie locale, utilise en ressource l'ontologie locale, et produit en sortie la plus petite partie de l'ontologie locale permettant de « relier » tous les concepts d'entrée. Son fonctionnement est le suivant :

Marquage des concepts : L'algorithme parcourt les axiomes à la recherche des URI des concepts de l'ontologie à segmenter ; lorsqu'il en trouve une, le concept correspondant de l'ontologie est marqué comme « alignable ». Il fait alors la même recherche dans les définitions des concepts marqués, ainsi que dans celles de leurs sous-concepts, et marque également les concepts qui y apparaissent en tant que co-domaines de rôles. Il parcours enfin les définitions des concepts non marqués, à la recherche de rôles pointant soit vers des concepts marqués, soit vers l'autre ontologie à aligner ; le cas échéant, il marque les concepts correspondants.

Construction de l'ontologie segmentée : l'algorithme recherche ensuite le Plus Petit Subsumeur Commun (PPSC) de tous les concepts marqués. Le résultat retourné est constitué du PPSC, de tous les concepts situés entre le PPSC et les concepts marqués, des concepts marqués eux-mêmes et des sous-concepts des concepts marqués.

Pour illustrer ce processus, nous considérons l'ontologie représentée sur la Figure 1.a dont seuls les concepts a_7, a_{13} et a_{15} sont concernés par les axiomes définis localement. L'application du processus de segmentation permet d'identifier a_3 comme le PPSC des trois concepts précédents. Ainsi, seul le sous-arbre de racine a_3 sera fourni en entrée du moteur d'inférences.



1.a *1.b*

FIG. 1 – *Segmentation d'une ontologie.*

En termes de complexité, la segmentation nécessite une itération sur l'ensemble des concepts de l'ontologie concernée, ainsi que sur les axiomes. Cette étape supplémentaire est donc linéaire, et à comparer avec les différentes étapes du processus de découverte par le moteur d'inférences dont la complexité peut – dans le pire des cas – être exponentielle pour la première phase (Le Pham et Le Than, 2006). Si cette étape aboutit à un « allègement » des ontologies en entrée du moteur d'inférences, elle peut donc vite s'avérer rentable dans le cas d'ontologies volumineuses.

3.3 Composition

Cette section présente la façon dont un pair fournit des correspondances entre ses voisins, en utilisant celles qu'il possède avec chacun d'entre eux, comme le montre la Figure 2 où le pair A connaît déjà le pair B et souhaite connaître le pair C. Sur cet exemple, comme le concept c_3 est subsumé par le concept b_5, lui-même subsumé par le concept b_2, il est possible de déduire grâce à la subsomption entre b_2 et a_6, que par transitivité, le concept a_6 subsume c_3.

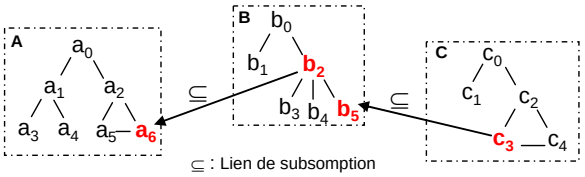


Fig.2 – *Ajout de correspondances par composition.*

L'algorithme de composition s'exécute sur le pair intermédiaire B, de manière à répartir la charge de calcul. Il reçoit en entrée une liste de correspondances entre A et B (cf. point « composition » du processus général de découverte de correspondances), utilise comme ressource l'ontologie locale de B, et fournit en sortie : une liste de correspondances entre A et B (qui peut avoir été modifiée), une liste de correspondances entre B et C (connaissance locale à B et à C), et une partie de l'ontologie de B (obtenue par segmentation).

Le fonctionnement de cet algorithme est le suivant : (i) B identifie sa liste de correspondances vers A, et la fusionne avec celle de A ; (ii) B identifie sa liste de correspondances vers C, demande à C sa liste de correspondances avec B, et fusionne ces deux listes ; (iii) B met en œuvre un processus de segmentation de son ontologie locale à partir des deux listes précédentes.

4 Architecture d'un pair

Cette section présente l'architecture globale d'un système P2P capable de découvrir automatiquement des correspondances sémantiques entre les pairs, en fonction de l'algorithme présenté en 3. L'architecture d'un pair est définie comme décrite en Figure 3.

La couche de communication d'un pair va gérer la communication entre un pair et son voisinage. La connaissance de ce voisinage est stockée dans la table de voisinage. Au-delà des aspects purement associés au fonctionnement du réseau, cette couche de communication va permettre de

traiter l'importation et l'exportation de segments d'ontologie ainsi que l'importation de listes de correspondances fournies par ses voisins. Les modules d'importation ont pour finalité de fournir les entrées nécessaires au moteur d'inférences pour découvrir les correspondances entre l'ontologie locale et les ontologies distantes ciblées. L'ensemble des correspondances sémantiques découvertes localement est géré par un module d'actualisation qui permet de maintenir la cohérence entre la table de voisinage d'un pair et les informations sémantiques stockées (un pair ne conserve que des correspondances concernant ses voisins).

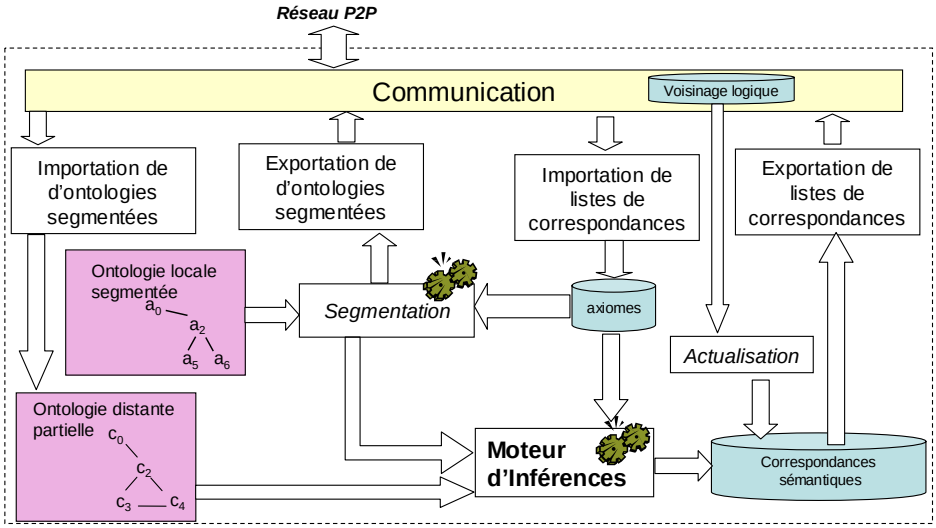


FIG. 3 – *Architecture d'un pair*

5 Conclusion et perspectives

Ce document présente une approche d'extension des résultats de découverte automatique de correspondances entre ontologies à l'aide de méthodes sémantiques, par et pour leur adaptation à des environnements pair à pair. Ce travail présente l'avantage de tirer partie de la connaissance distribuée entre les pairs, et vise le partage de connaissances et de temps de calcul. Il est actuellement en cours de réalisation et n'a pas encore été validé. Une perspective à court terme est donc l'implémentation et la mise en œuvre d'un prototype.

Dans un deuxième temps, nous envisageons d'accroître la répartition du raisonnement en expérimentant l'utilisation d'algorithmes plus appropriés au raisonnement distribué à large échelle. En effet, nous souhaitons tirer parti de travaux sur les logiques de description distribuées (Borgida et Serafini, 2003) pour répartir autant que possible le calcul de découverte de correspondances sur les différents pairs y prenant part.

Enfin, il est possible d'organiser le réseau P2P afin de valoriser les liens logiques entre les pairs en fonction des correspondances dont ils disposent. A cette connaissance, il serait possible d'ajouter la connaissance des centres d'intérêt des utilisateurs pour disposer d'un *a priori* sur la proximité sémantique entre les pairs. La considération de cette proximité sémantique entre pairs aboutirait selon le principe montré par Lumineau et Doucet (2004) à une meilleure interaction entre les pairs et une réduction de la charge globale du réseau.

Références

Adjiman P., Chatalic P., Goasdoué F., Rousset M-C., Simon L. (2005). *SomeWhere in the Semantic Web*. In proc. PPSWR 2005: 1-16.

Baader, F., Borgida, A., Brachman, R. J. et al. (2003). *The Description Logic Handbook, Theory, Implementation and Applications.* In: Baader, F., Calvanese D., McGuinness D., Nardi D. and Patel-Schneider P. (eds.): Cambridge University Press.

Baader F. Sattler U. (2001). *An Overview of Tableau Algorithms for Description Logics.* Studia Logica, 69:5–40.

Borgida, A., L. Serafini (2003). *Distributed description logics: Assimilating information from peer sources*. Journal of Data Semantics, pages 153–184.

Bouquet, P., Euzenat, J., Franconi, F., Serafini, L., Stamou, G., Tessaris, S. (2004). *D2.2.1 Specification of a common framework for characterizing alignment*. Knowledge Web project, realizing the semantic web. IST-2004-507482 Programme of the Commission of the European Communities.

Euzenat, J., T. Le Bach, J. Barrasa, P. Bouquet, J. De Bo, R. Dieng *et al.* (2004). *D2.2.3: State of the art on ontology alignment. Knowledge Web project, realizing the semantic web*, EU-IST-2004-507482.

Ferreira Da Silva, C., L. Médini, S. Abdul Ghafour, P. Hoffmann, P. Ghodous, C. Lima (2006). *Semantic Interoperability of Heterogeneous Semantic Resources*. Electronic Notes in Theoretical Computer Science 150(2):71-85, Elsevier.

Giunchiglia, F., P. Shvaiko, M. Yatskevich (2004). *S-Match: an algorithm and an implementation of semantic matching.* Proceedings of ESWS 2004. Heraklion, Greece 61–75.

Horrocks, I. (1997). *Optimising Tableaux Decision Procedures for Description Logics*. PhD Thesis, University of Manchester.

Kementsietsidis A., Arenas M., Miller R. J. (2003). *Managing Data Mappings in the Hyperion Project*. In proc. ICDE: 732-734.

Kubiatowicz J., Bindel D., Chen Y., Czerwinski S., Eaton P., Geels D., Gummadi R., Rhea S., Weatherspoon H., Wells C., Zhao B. (2000). *OceanStore: an architecture for global-scale persistent storage.* ACM SIGARCH Computer Architecture News, v.28 n.5, p.190-201.

Le Duc, C. (2004). *Transformation d'ontologies basées sur la Logique de Description - Application dans le Commerce Electronique*, Thèse de doctorat de l'Université de Nice.

Le Pham, T. A., N. Le Than (2006). *Raisonnement sur des ontologies décomposées.* Rapport de recherche INRIA ISRN I3S/RR-2006-15-FR, Projet Mainline.

Lumineau, N., Doucet, A. (2004) *Sharing Communities Experiences for Query Propagation in Peer-to-Peer Systems.* In 8th International Database Engineering and Applications Symposium (IDEAS'04), Coimbra, Portugal, 7-9 July 2004.

Muthitacharoen A., Morris R., Gil T.G., Chen B. (2002). *Ivy: A Read/Write Peer-to-Peer File System.* In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA.

Nejdl W., Boris Wolf B., Staab S., Tane J.(2002). *EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network.* Semantic Web Workshop, Hawaï.

Ooi B., Shu Y., Tan K-L. (2003). *Relational data sharing in peer-based data management systems.* SIGMOD Record, 32(3).

Oram A. (2001). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly.

Schmidt-Schaub, M., G. Smolka (1991). *Attributive concept descriptions with complements*. Artificial Intelligence, Vol. 48(1) 1-26.

Sigmod Record (2003). *Special section on peer-to-peer data management*. Volume 32 (3).

Stoica, I., R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan (2001). *Chord: Scalable Peer-To-Peer lookup service for internet applications*. In Proc. ACM SIGCOMM Conference, pages 149–160.

Tatarinov I., Ives Z., Madhavan J., Halevy A., Suciu D., Dalvi N., Dong X., Kadiyska Y., Miklau G., Mork P. (2003). *The Piazza peer data management project*. SIGMOD Rec. (32) pp:47-52.

Summary

This paper presents an approach for automatic discovery of semantic mappings between ontologies in a P2P network. This process is based on semantic methods and description logics. It requires each peer to embed an inference engine, and to collaborate with other peers to exchange ontology parts as well as axioms over the network. A particular aspect of this approach is that it takes advantage of knowledge and mappings related to other peers to enrich the set of axioms that will be used to perform the mappings discovery process. The main challenge to solve is to significantly the computation time needed for semantic methods to perform this process. For this, we explore several strategies, such as reducing the size of the input ontologies, and distributing the operations to be performed by the algorithm upon the different peers involved in the alignment process.

Appariement de schémas de BD géographiques à l'aide d'ontologies déduites des spécifications

Sébastien Mustière, Nathalie Abadie, Frédéric Laurens

Institut Géographique National
Laboratoire COGIT
2 av. pasteur
94160 SAINT-MANDE
sebastien.mustiere@ign.fr, nathalie-f.abadie@ign.fr
http://recherche.ign.fr/labos/cogit/

Résumé. L'intégration de base de données géographiques peut être facilitée par l'utilisation d'ontologies du domaine. Ces ontologies peuvent être déduites de l'analyse semi-automatique des spécifications textuelles de ces bases de données. Cet article présente des travaux en cours et les principales difficultés qui apparaissent pour réaliser cela.

1 Introduction

De nombreuses bases de données géographiques coexistent pour représenter un même espace. Ces bases ont été réalisées pour répondre à différents besoins (urbanisme, navigation…) et possèdent différents niveaux d'analyse (échelle du pays, de la ville…). Une gestion relativement indépendante de ces bases pose divers problèmes pour le producteur comme pour l'utilisateur des données. Tout d'abord il peut y avoir des incohérences entre les bases. Ensuite les efforts de saisie, de maintenance et de mise à jour sont multipliés. Enfin, il est difficile de réaliser des analyses combinant des données avec différents points de vue. Une solution possible à ces problèmes est de rendre explicites les relations entre les bases de données. Cette intégration soulève deux problèmes, l'appariement des schémas des bases, et l'appariement des données elles-mêmes rendu complexe par l'absence d'identifiant universel sur les données géographiques. Les travaux présentés ici portent sur l'appariement des schémas de bases de données géographiques, l'appariement de données faisant l'objet de recherches parallèles (Olteanu et al. 2005).

2 Des ontologies pour apparier des schémas de BD

Pour apparier deux schémas, une comparaison directe de ceux-ci se révèle souvent difficile, et ceci d'autant plus que le monde représenté dans les bases est complexe, conduisant à des choix de modélisation variés, ce qui est le cas pour les données géographiques. Une approche de plus en plus privilégiée, autant dans le monde des bases de données en général (Partridge 2002), que dans celui des systèmes d'information géographiques (Uitermark 2001, Gesbert 2005), est d'appuyer l'appariement sur une

ontologie du domaine concerné. Chaque schéma de données est alors relié à l'ontologie (voir un exemple en figure 1), et des relations schéma/schéma peuvent être ensuite déduites des deux ensembles de relations schéma/ontologie. C'est la voie que nous privilégions dans nos travaux.

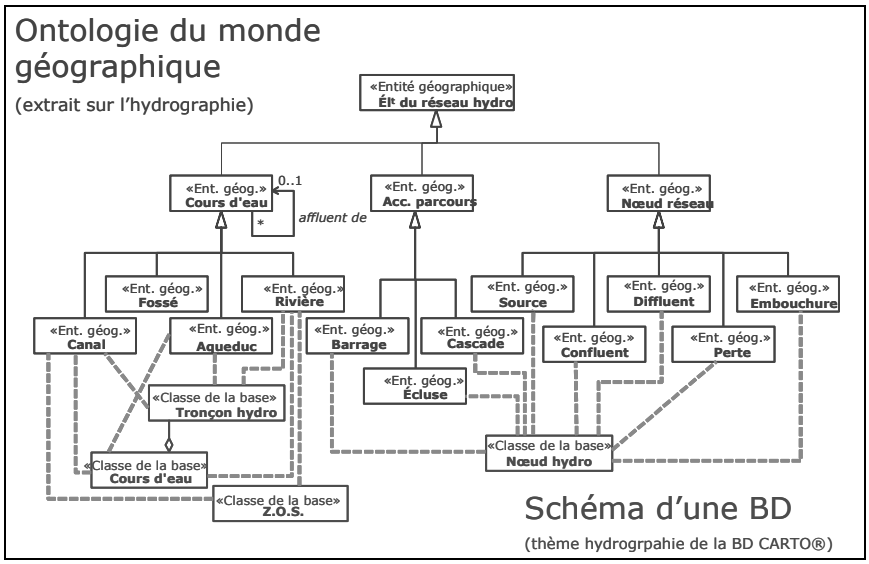
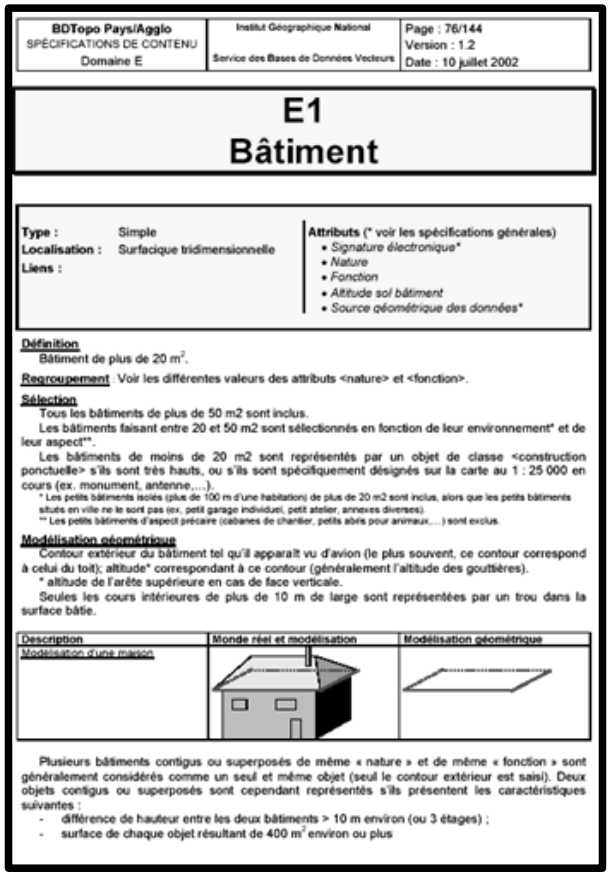


Fig. 1 - Liens entre une ontologie et un schéma de BD, d'après Gesbert (2005)

3 Les spécifications comme sources des ontologies

Deux questions se posent alors : comment constituer une ontologie du domaine, et comment représenter et instancier le lien entre un schéma et une ontologie ? Comme proposé par Gesbert (2005), nos recherches aspirent à répondre à ces deux questions pour les données géographiques à partir d'une même source de connaissances : les spécifications textuelles des bases de données. En raison de la complexité du monde géographique, et surtout en raison de la diversité des choix réalisés pour modéliser et instancier les bases de données géographiques, les producteurs associent généralement à leurs bases de volumineuses spécifications (typiquement d'une centaine de pages, voir un extrait en figure 2). Prenons l'exemple de la classe *Tronçon hydrographique* d'une base de données, pour laquelle les spécifications précisent que « Un *tronçon hydrographique* correspond à l'axe du lit d'une rivière, d'un ruisseau ou d'un canal », mais aussi que la base contient « tous les axes principaux, y compris dans la zone d'estran et dans les zones de marais, à l'exception des "culs-de-sac" d'une longueur inférieure à un kilomètre […] ». Ces spécifications font donc de fait référence à une ontologie du domaine (concepts de *rivière*, *ruisseau*, *canal*, *estran*, *marais*…), et explicitent le lien entre les classes et l'ontologie (la classe *Tronçon hydrographique* représente *toutes les rivières, à l'exception de*….). Ces spécifications sont des documents assez fortement structurés, mais principalement rédigés en langage naturel.



BDTopo Pays/Agglo
SPÉCIFICATIONS DE CONTENU
Domaine E

Institut Géographique National
Service des Bases de Données Vecteurs

Page : 76/144
Version : 1.2
Date : 10 juillet 2002

E1
Bâtiment

Type : Simple
Localisation : Surfacique tridimensionnelle
Liens :

Attributs (* voir les spécifications générales)
- *Signature électronique**
- *Nature*
- *Fonction*
- *Altitude sol bâtiment*
- *Source géométrique des données**

Définition
Bâtiment de plus de 20 m².

Regroupement : Voir les différentes valeurs des attributs <nature> et <fonction>.

Sélection
Tous les bâtiments de plus de 50 m2 sont inclus.
Les bâtiments faisant entre 20 et 50 m2 sont sélectionnés en fonction de leur environnement* et de leur aspect**.
Les bâtiments de moins de 20 m2 sont représentés par un objet de classe <construction ponctuelle> s'ils sont très hauts, ou s'ils sont spécifiquement désignés sur la carte au 1 : 25 000 en cours (ex. monument, antenne,...).
* Les petits bâtiments isolés (plus de 100 m d'une habitation) de plus de 20 m2 sont inclus, alors que les petits bâtiments situés en ville ne le sont pas (ex. petit garage individuel, petit atelier, annexes diverses).
** Les petits bâtiments d'aspect précaire (cabanes de chantier, petits abris pour animaux,...) sont exclus.

Modélisation géométrique
Contour extérieur du bâtiment tel qu'il apparaît vu d'avion (le plus souvent, ce contour correspond à celui du toit); altitude* correspondant à ce contour (généralement l'altitude des gouttières).
* altitude de l'arête supérieure en cas de face verticale.
Seules les cours intérieures de plus de 10 m de large sont représentées par un trou dans la surface bâtie.

Description	Monde réel et modélisation	Modélisation géométrique
Modélisation d'une maison		

Plusieurs bâtiments contigus ou superposés de même « nature » et de même « fonction » sont généralement considérés comme un seul et même objet (seul le contour extérieur est saisi). Deux objets contigus ou superposés sont cependant représentés s'ils présentent les caractéristiques suivantes :
- différence de hauteur entre les deux bâtiments > 10 m environ (ou 3 étages) ;
- surface de chaque objet résultant de 400 m² environ ou plus

Fig. 2 - Extrait de spécifications textuelles d'une base de données géographiques

4 Automatisation de la construction des ontologies

4.1 Traitement semi-automatique du langage naturel

Afin de passer à l'échelle, c'est-à-dire d'apparier l'intégralité des schémas de nombreuses bases, il nous semble nécessaire d'automatiser autant que possible la construction d'une ontologie à partir de ces spécifications en langage naturel, ainsi que la détermination des liens entre le schéma et l'ontologie. Il peut être objecté à notre approche que si à chaque spécification correspond une ontologie sous-jacente, le problème de l'appariement des schémas sera uniquement déplacé et ramené à un problème d'alignement des ontologies sous-jacentes. Cependant, nous constatons que ce nouveau problème, s'il reste entier, est plus facile à résoudre que le problème initial, car les ontologies déduites des spécifications sont beaucoup plus proches les unes des autres que ne le sont les schémas des bases de données correspondantes. Par exemple, les aqueducs sont représentés dans deux bases

distinctes de l'IGN, mais ceux-ci sont représentés dans la classe *Tronçon hydrographique* du thème *Hydrographie* dans une base, et dans la classe *Canalisation* du thème *Transport d'énergie et de fluides* dans l'autre base : il est difficile de faire a priori le lien entre ces deux classes. Mais a contrario, on remarque que le concept d'*Aqueduc* est présent et similaire dans les ontologies sous-jacentes aux deux bases. Seule une analyse des spécifications textuelles met en valeur le lien entre les deux classes mentionnées à travers le concept d'aqueduc présent dans les deux ontologies sous-jacentes.

Nous avons donc réalisé une analyse des spécifications par traitement automatique du langage naturel pour, dans un premier temps, découvrir les ontologies sous-jacentes aux spécifications (voir figure 3, Laurens 2006). Ceci est nécessaire avant de, dans un deuxième temps, détecter et formaliser les relations entre schémas et ontologies.

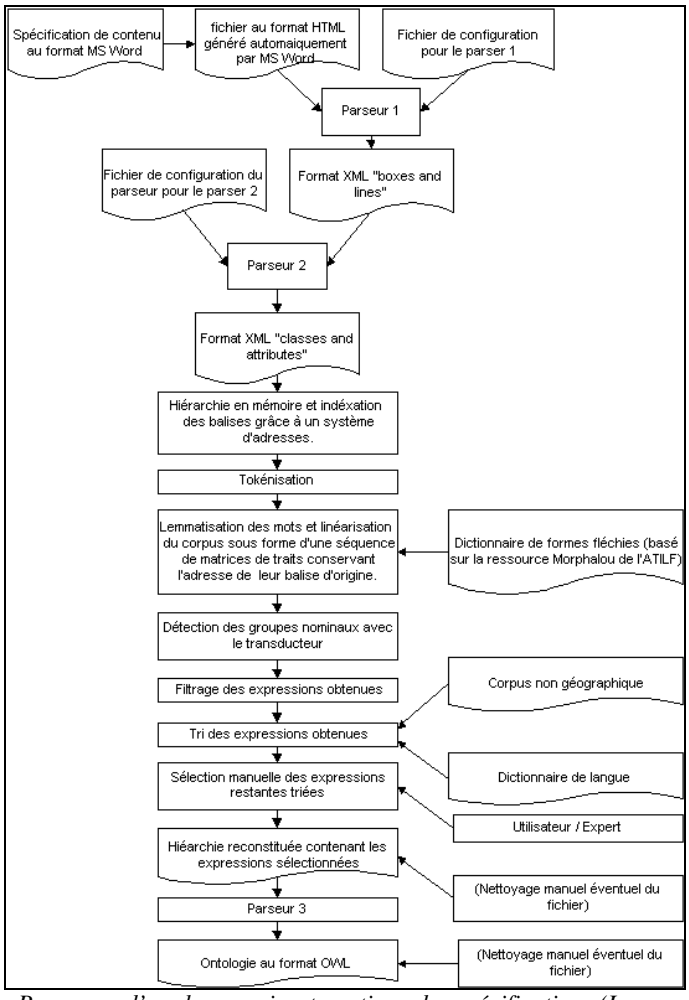


Fig. 3 - Processus d'analyse semi-automatique des spécifications (Laurens 2006)

4.2 Résultats de l'automatisation

Les résultats obtenus (cf. figure 4) montrent en premier lieu qu'une détermination semi-automatique d'ontologie est possible, même si un post-traitement interactif se révèle toujours nécessaire : concrètement, on évalue à une journée de travail le post-traitement interactif nécessaire à la finalisation de l'ontologie obtenue automatiquement.

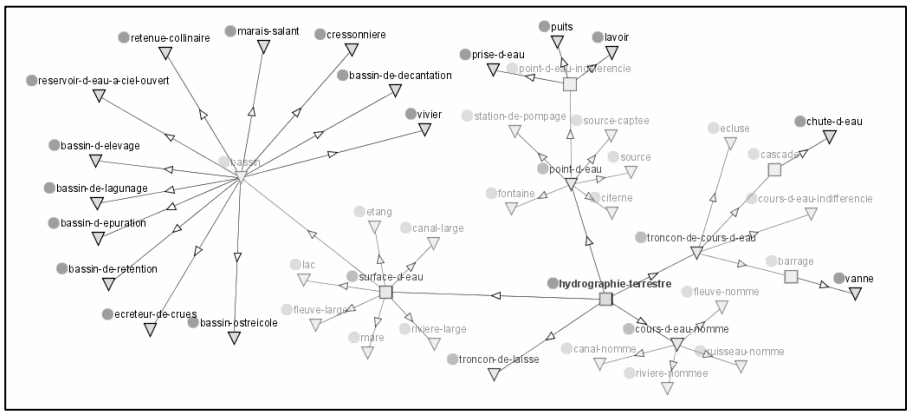


Fig. 4 - Extrait de l'ontologie issue de l'analyse semi-automatique de spécifications (extrait de Laurens 2006, Visualisation : Protégé-OWL)

Nos résultats montrent également que l'analyse automatique du langage naturel des spécifications est plus efficace en suivant une approche par règles qu'une approche statistique, ceci en raison de la faible répétition des concepts de l'ontologie dans les spécifications. En effet, certaines parties caractéristiques des spécifications telles que les noms de classe ou de thèmes qui les regroupent renferment presque toujours des concepts intéressants, et il est donc plus avantageux de se baser sur une analyse de la structure du document et des heuristiques linguistiques que sur une analyse fréquentielle ou distributionnelle des termes utilisés. Nos résultats mettent enfin en valeur les principales difficultés rencontrées pour une analyse automatique des spécifications.

En premier lieu, les concepts pertinents peuvent se situer à divers endroits et il est donc difficile de les détecter tous. Ils peuvent apparaître soit dans le nom même des classes (ex : *Tronçon hydrographique*), soit dans la définition associée (ex : « lit d'une *rivière*, *ruisseau*, *canal*… »), soit dans les règles de sélection associées (ex : « …dans les zones de *marais*… »), soit dans les valeurs possibles d'un attribut (ex : valeur « *aqueduc* » possible pour l'attribut *Nature*), soit dans la définition associée à cette valeur (ex : « *Tuyau* ou *chenal* artificiel… »). Ajoutons à cela une difficulté supplémentaire du au fait que derrière le titre « définition » rencontré dans les spécifications on trouve différentes acceptions du terme : il peut s'agir d'une réelle définition (ex : définition de la valeur d'attribut *cap :* « proéminence dans le contour d'une côte »), mais il peut aussi s'agir en fait d'une précision de ce qui est saisi dans la base (ex : définition de *carrière* : « carrière à ciel ouvert »).

En second lieu, certaines hiérarchies sont difficiles à expliciter. Tout d'abord, de nombreuses hiérarchies naturelles de concepts sont mises à plat dans les spécifications et il est donc impossible de les détecter automatiquement sans source de connaissances extérieure (ex : un attribut peut avoir pour valeurs « sommet », « pic », « vallée » ou « plage », sans que la proximité naturelle entre sommet et pic soit explicitée). Ensuite, il existe des concepts difficiles à nommer autrement que par un des sous-concepts représentatifs qui peuvent être mentionnés (ex : la valeur *dépression* représente « une cuvette, un bassin fermé, une *dépression*, ou une doline » et il est difficile de trouver un terme plus général). Enfin, les sous-concepts mentionnés sont parfois des réelles spécialisations (« Vallée : combe, ravin, val, talweg… »), parfois des termes qui peuvent être plutôt vus comme des synonymes (« Isthme : isthme, cordon littoral »).

5 Conclusion

Nos travaux en cours nous encouragent à poursuivre dans cette voie de l'analyse des spécifications pour apparier les schémas, mais montrent aussi la limite à une complète automatisation de l'extraction d'ontologie pour réaliser cela. D'autres voies sont aussi explorées, comme l'induction de règles de correspondances entre schémas à partir d'un appariement de données réalisé sur des critères géométriques par exemple (Sheeren 2005).

Références

Gesbert N. (2005). *Formalisation des spécifications de bases de données géographiques en vue de leur intégration*, Thèse de doctorat, Université de Marne La Vallée.

Laurens F. (2006). *Création d'une ontologie à partir de textes en langage naturel*. Stage de Master 1 Linguistique-Informatique, Université Paris 7.

Olteanu A.M., Mustière S., Ruas A. (2006). Matching Imperfect Data. Dans les actes de International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Science (Accuracy'2006), Lisbon, pp.694-704.

Partridge C. (2002). *The role of ontology in integrating semantically heterogeneous databases*. Technical Report 05/02 LADSEB-CNR, Padoue.

Sheeren D. (2005). *Méthodologie d'évaluation de la cohérence inter-représentations pour l'intégration de bases de données spatiales – Une approche combinant l'utilisation de métadonnées et l'apprentissage automatique*. Thèse de doctorat de l'université Paris 6.

Uitermark H. (2001). *Ontology-Based Geographic Data Set Integration*. PhD thesis, Universiteit Twente, Pays-Bas.

Summary

Integration of geographic databases may be facilitated by the use of domain ontologies. These one can be deduced from the processing of textual specifications of the databases. This paper presents ongoing works on this subject and the main difficulties encountered.

Utilisation de connaissances supplémentaires pour la découverte de mappings dans le système *TaxoMap*

C. Reynaud, B. Safar

Université Paris-Sud, CNRS (L.R.I.) & INRIA (Futurs), Orsay, France
{chantal.reynaud, brigitte.safar}@lri.fr

1 Introduction

La recherche de documents pertinents sur le Web est une tâche encore souvent laborieuse. Le Web sémantique devrait faciliter ce travail en réalisant un appariement sémantique entre les requêtes des utilisateurs et les documents auxquels sont associées des méta-données issues d'ontologies décrivant leur contenu. Notre travail contribue à faciliter cet appariement en proposant des techniques de mise en correspondance entre ontologies, mises en œuvre dans le système *TaxoMap* (Kéfi (2006)).

Les schémas en entrée du processus de mise en correspondance sont des taxonomies. Une taxonomie (C, H_C) comprend un ensemble de concepts C et une hiérarchie de subsomption entre concepts H_C. Un concept est ainsi défini uniquement par son label et les relations de sous-classes qui le relient à d'autres concepts.

L'approche que nous proposons est particulièrement bien adaptée aux taxonomies ayant les caractéristiques suivantes :

- des taxonomies dissymétriques : une taxonomie étant très peu structurée, voire pas du tout (simple ensemble de termes), alors que la seconde l'est davantage.
- des taxonomies comportant exclusivement des relations de subsomption entre concepts.
- des taxonomies très spécifiques comportant des descriptions très fines de domaines d'application, ce qui se traduit au niveau des concepts par : (1) des concepts appartenant à un même domaine circonscrit, (2) des labels correspondant à des expressions composées de plusieurs mots, (3) des labels des concepts généraux inclus dans les labels des concepts plus spécifiques.

Le processus d'alignement a pour objectif de mettre en correspondance des concepts de la taxonomie la moins structurée, la taxonomie source (T_{Source}), avec les concepts de la taxonomie la plus structurée, la taxonomie cible (T_{Cible}). Ce processus génère des mappings 1-1 qui sont des relations d'équivalence ou des relations de sous-classes. Une relation d'équivalence *isEq* est un lien entre un concept dans T_{Source} et un concept dans T_{Cible} dont les noms sont considérés comme étant similaires. Les relations de spécialisation *isA* sont des liens usuels de sous-classe/super-classe.

Cette approche a des applications dans le contexte du Web. A titre d'illustration, nous présenterons lors de l'exposé deux cas d'utilisation :

1. L'interrogation élargie de portails Web .

Le système *TaxoMap* peut être utilisé pour aligner la taxonomie associée à des documents externes (T_{Source}) avec celle d'un portail Web (T_{Cible}) de façon à augmenter le nombre de documents accessibles à partir de ce portail. La recherche de documents s'appuie en général sur des ontologies très simples, des taxonomies. Les taxonomies de concepts sur lesquelles se basent les portails Web sont en général bien structurées, celles des autres documents auxquels on souhaiterait avoir accès via ce portail ne le sont pas toujours.

2. L'annotation sémantique de documents Web

L'approche peut être utile pour lier des termes isolés, extraits de documents Web, à ceux d'une ontologie, et générer des annotations sémantiques basées sur cette ontologie.

Après avoir décrit l'approche générale mise en œuvre dans *TaxoMap*, nous présenterons tout d'abord les aspects favorisant son application à grande échelle. Deux caractéristiques seront plus particulièrement mises en avant.

- l'application séquentielle de différentes techniques (terminologiques et structurelles) complémentaires, exécutables indépendamment les unes des autres. L'ordre d'exécution a été déterminé de façon à ce que le processus d'alignement soit le plus efficace possible, étant donnée la nature des taxonomies rapprochées (Kefi et al. 2006). L'indépendance des techniques autorise cependant la sélection de celles qui sont les plus appropriées dans un contexte d'utilisation donné, ou en fonction des caractéristiques des taxonomies à rapprocher (domaine couvert, longueur des labels, …).

- la proposition de techniques différentes par rapport à leur degré d'automatisation. Les techniques que nous proposons se décomposent en deux catégories par rapport aux critères de précision et de rappel. Les techniques de la première catégorie permettent la découverte de mappings probables, i.e les éléments dont les mappings ont de fortes chances d'être pertinents. Leur précision est très grande au détriment du rappel (qui reste malgré tout significatif) (Reynaud et Safar (2006b)). Les techniques de la seconde catégorie génère des mappings dont le rappel est meilleur mais la précision inférieure. Cette distinction permet de considérer que les mappings proposés par la 1ère catégorie sont acceptables tels quels sans être validés par un expert. Le processus d'exécution peut-être complètement automatisé. En revanche, les mappings générés par la seconde catégorie de techniques correspondent plus à des suggestions de mappings potentiels faites à l'utilisateur. Le processus de découverte de cette catégorie de mappings doit être complété par une phase de validation à la charge de l'expert. Le recours à ces techniques dépend de la possibilité ou non d'effectuer cette phase de validation.

Nous nous focaliserons ensuite sur une des techniques de *TaxoMap*, spécifique au rapprochement de modèles ontologiques pauvres sémantiquement. Plus précisément, nous traiterons de l'utilisation de connaissances supplémentaires pour pallier l'insuffisance de connaissances contenues dans les modèles alignés. Nous présenterons la technique STR_W de *TaxoMap* basée sur l'exploitation de la structure de WordNet. Nous effectuerons une analyse comparative de cette technique par rapport à l'état de l'art, en particulier par rapport aux travaux récemment réalisés par Sabou et al. (2006) et Aleksovski et al. (2006a, 2006b).

2 Utilisation de connaissances de support

Pour identifier des mappings entre les concepts de deux ontologies, O_{Src} et O_{Tar}, de nombreux travaux portent actuellement sur l'utilisation de connaissances supplémentaires dites de "background" ou de support, représentées le plus souvent sous la forme d'une 3ème ontologie, O_{BK} (voir Aleksovski et al., 2006a, 2006b, Sabou et al., 2006, Reynaud et Safar, 2006a, 2006b). L'objectif de ces travaux est de compléter les techniques classiques d'appariement qui exploitent la structure ou la richesse du langage de représentation des ontologies, et qui ne s'appliquent plus quand les ontologies à apparier sont faiblement structurées ou se limitent à des simples hiérarchies de classification.

Nous présentons tout d'abord, de façon globale, l'approche générale commune à ces différents travaux et nous reviendrons ensuite sur les aspects qui les différencient.

2.1 Approche générale

Pour identifier l'existence d'un mapping de la forme (X_{Src} *relation* Y_{Tar}) où $X_{Src} \in O_{Src}$, $Y_{Tar} \in O_{Tar}$, et *relation* $\in T$, l'ensemble des relations exprimables entre deux concepts dans les ontologies considérées, l'approche générale suivie par ces différents travaux se décompose en 2 phases : l'ancrage et la dérivation.

L'**ancrage** consiste tout d'abord à apparier chacun des 2 concepts X_{Src} et Y_{Tar}, pris indépendamment l'un de l'autre, avec un ou des concepts de la 3ème ontologie (O_{BK}), c'est-à-dire, à identifier des mappings de la forme (X_{Src} *relation* X_{BK}) et (Y_{Tar} *relation* Y_{BK}) où X_{BK} et $Y_{BK} \in O_{BK}$ et sont appelés des *ancres* ou *points d'ancrage*.

La **dérivation** consiste ensuite à s'appuyer sur la structuration de O_{BK} pour rechercher s'il existe des relations entre les différents points d'ancrage identifiés, puis essayer d'en dériver des relations entre les éléments des ontologies à apparier.

L'ensemble T des relations utilisées dans ces différents travaux est l'ensemble $\{\leq, \geq, \equiv\}$ où $X \leq Y$ peut se lire suivant les cas comme *«X isA Y», «X part-of Y»* ou plus généralement *«X narrower-than Y»* et les mappings (X_{Src} *relation* Y_{Tar}) cherchés sont dérivés en utilisant des règles de la forme :

Si ($X_{Src} \leq X_{BK}$) et ($X_{BK} \leq Y_{BK}$) et ($Y_{BK} \leq Y_{Tar}$) alors dériver ($X_{Src} \leq Y_{Tar}$)

Si ($X_{Src} \geq X_{BK}$) et ($X_{BK} \geq Y_{BK}$) et ($Y_{BK} \geq Y_{Tar}$) alors dériver ($X_{Src} \geq Y_{Tar}$).

Ces règles utilisent aussi la relation d'équivalence, $\equiv$, en considérant que l'existence d'une relation de type $A \equiv B$ permet de rajouter les deux relations $A \leq B$ et $A \geq B$ et qu'inversement, le fait d'avoir pu dériver les deux relations $X_{Src} \leq Y_{Tar}$ et $X_{Src} \geq Y_{Tar}$ permet de dériver la relation $X_{Src} \equiv Y_{Tar}$.

Si l'on considère que l'appariement d'ontologies est une fonction sur 2 ontologies qui retourne un ensemble de relations entre leurs concepts, $f : (O_1,O_2) \rightarrow \{(X \textit{ relation } Y) \mid X \in O_1, \textit{relation} \in T, Y \in O_2\}$, l'approche générale suivie par ces différents travaux revient donc à faire globalement trois appariements d'ontologie. En effet, la phase d'ancrage comporte

deux appariements d'ontologies f(O_{Src}, O_{BK}) et f(O_{Tar}, O_{BK}) et la phase de dérivation, un appariement d'une ontologie sur elle-même f(O_{BK}, O_{BK}).

Pour effectuer la phase d'ancrage vers les éléments de O_{BK}, les auteurs s'appuient sur des heuristiques terminologiques simples qui travaillent sur les labels et les synonymes des termes désignant des concepts. Par exemple, en utilisant une mesure de type *edit-distance* et en considérant que si les labels de deux concepts ne se différencient pas par plus de deux caractères, les concepts considérés peuvent être reliés par une relation d'équivalence ou en utilisant une heuristique d'inclusion de labels qui consiste à dire que si tous les mots du label ou du synonyme d'un concept A se trouvent dans le label ou le synonyme d'un concept B, alors B sera considéré comme plus spécialisé que A ($B \leq A$).

A partir de ce schéma général, les travaux se différencient sur les caractéristiques des ontologies employées comme support et sur les stratégies de mise en œuvre des deux phases.

2.2 Les travaux d'Aleksovski et al.

L'idée de base qui sous-tend ces travaux est que l'ontologie de support O_{BK} est plus complète et plus détaillée que les deux ontologies à rapprocher, et qu'elle contient une description en compréhension du domaine des 2 autres.

Les deux phases d'ancrage et de dérivation sont réalisées globalement : l'ancrage consiste tout d'abord à essayer d'apparier chacun des concepts des 2 ontologies initiales (O_{Src} et O_{Tar}) avec les concepts de la 3ème (O_{BK}). La dérivation consiste ensuite à rechercher au sein de O_{BK} les relations qui existent entre les différents points d'ancrage identifiés, puis d'en dériver des relations entre les éléments des ontologies à apparier. De multiples points d'ancrage pouvant être trouvés pour un concept (certains trivialement faux d'ailleurs, quand l'heuristique d'inclusion de labels est employée brutalement sans garde-fou), si les points d'ancrage de 2 concepts X et Y sont reliés de façons similaires, cela renforce le lien entre les deux concepts. S'ils sont reliés de façon incompatible (ex $X^1_{BK} \leq Y^1_{BK}$ et $X^2_{BK} \geq Y^2_{BK}$) aucune relation de subsomption ne peut être inférée entre X et Y, mais cela révèle quand même que les 2 concepts ont un lien avec une certaine proximité sémantique.

Les auteurs soulignent le fait que des concepts issus de 2 ontologies différentes sont rarement équivalents mais partagent en fait un certain recouvrement sémantique, et qu'identifier de tels recouvrements est utile dans la tâche d'intégration.

Dans Aleksovski et al. (2006a**)**, les concepts à rapprocher sont des éléments issus de 2 listes de vocabulaires plats, non structurés. L'ontologie O_{BK} utilisée pour rechercher les dérivations est une ontologie représentant des points de vue multiples (ou aspects), ce qui permet d'identifier plusieurs dérivations entre 2 points d'ancrage, suivant les différents aspects. Un ensemble de mappings (Gold Standard) a été élaboré avec le concours manuel d'un expert. Puis, les auteurs ont réalisé 2 expérimentations : l'une, appelée lexical matching (LM), en recherchant directement des appariements entre les termes de O_{Src} et O_{Tar}, l'autre (Semantic Matching, SM) en recherchant d'abord les ancrages dans O_{BK}, puis les dérivations entre les paires d'ancres trouvées. Rien n'est dit de très précis dans ce papier sur le type de

relation définie entre les termes d'un appariement, ni sur la façon dont on les obtient ou les combine.

Bien qu'ils aient remplacé un simple problème de recherche d'appariement (de O_{Src} vers O_{Tar}) par un double problème (de O_{Src} vers O_{BK}, et de O_{Tar} vers O_{BK}), les auteurs observent une amélioration de la précision des mappings obtenus. Ceci peut s'expliquer par l'existence de multiples dérivations obtenues dans O_{BK}, qui permet d'identifier des proximités sémantiques non identifiables par de simples rapprochements terminologiques.

Remarquons aussi que l'étape d'ancrage de O_{Tar} vers O_{BK} n'a pas été effectuée automatiquement mais manuellement, ce qui introduit probablement un biais, car l'étape d'ancrage des termes de O_{Src} vers O_{BK} n'apparaît pas si simple. Avec les mêmes techniques terminologiques, les auteurs ont ancré moins de termes de O_{Src} dans O_{BK} qu'ils n'avaient trouvé d'appariements directs de O_{Src} vers O_{Tar} ! (moins mais mieux car la précision est légèrement meilleure).

Remarquons aussi que les 2 expérimentations ont été effectuées de façon exclusive, sans essayer de combiner les 2 approches.

Dans Aleksovski et al. (2006b**)**, les concepts à rapprocher appartiennent à 2 ontologies vraiment structurées par des relations du type « *X narrower-than Y* » et « *X Broader-than Y* » ($\{\leq, \geq\}$) et l'O_{BK} contient des relations de type *is-a* et *part-of*.
Ces 2 relations permettront d'inférer des relations de type *narrower-than*, dans la recherche de dérivation entre 2 ancres en utilisant les règles suivantes :
Si (X_{BK} *isA* Y_{BK}) alors dériver ($X_{BK} \leq Y_{BK}$) et si (X_{BK} *part-of* Y_{BK}) alors dériver ($X_{BK} \leq Y_{BK}$).

Les auteurs utilisent la fermeture transitive de relations :
Si (X^1_{BK} *isA* X^2_{BK}) et (X^2_{BK} *isA* X^3_{BK}) et .. et (X^{n-1}_{BK} *isA* X^n_{BK}) alors dériver ($X^1_{BK} \leq X^n_{BK}$).
Cette fermeture s'applique aussi aux relations *part-of* et peut mêler les relations *isA* et *part-of* ou au contraire imposer de n'utiliser les relations *isA* qu'après avoir utilisé tous les *part-of*.

De nouvelles expérimentations sont effectuées dans ce contexte, la 1ere en recherchant directement des appariements entre les termes de O_{Src} et O_{Tar}, et les suivantes par dérivation, en utilisant ou pas la fermeture transitive de relation, et sans imposer ou en imposant des contraintes sur l'ordre d'utilisation des relations *isA* et *part-of* lors de la fermeture. Pour pallier l'absence de mappings de référence, les évaluations de ces expérimentations ont été faites en choisissant au hasard 30 concepts de O_{Src} et en évaluant manuellement la correction des relations trouvées. La dernière technique de dérivation est celle qui donne les meilleurs résultats, toutes les relations identifiées ayant été jugées correctes.

2.3 Les travaux de Sabou et al.

A l'opposé des travaux précédents, les auteurs considèrent qu'il n'existe pas a priori une ontologie qui soit plus complète et plus détaillée que les deux ontologies à rapprocher, et qui puisse seule servir de support. Ils proposent donc d'utiliser l'ensemble des ontologies accessibles sur le Web par l'intermédiaire du moteur de recherche sémantique Swoogle. Pour identifier l'existence d'un mapping de la forme (X_{Src}, *relation,* Y_{Tar}), les auteurs proposent de

rechercher à la volée les ontologies qui permettent l'ancrage simultané des deux concepts à apparier, puis de chercher s'il existe une dérivation entre les deux ancres dans les ontologies considérées.

L'approche peut paraître beaucoup plus coûteuse que la précédente puisqu'elle travaille séquentiellement sur toutes les paires de concepts possibles et impose a priori de rechercher plusieurs fois l'ancrage d'un même concept dans une même ontologie, théoriquement autant de fois que de concepts avec lequel on essaye de le mettre en relation. Mais elle permet d'identifier à la volée, sans choix manuel préalable, les ontologies susceptibles de servir de background même à un seul mapping et elle est parallélisable. De plus, l'approche est présentée comme complémentaire à d'autres techniques d'appariement et n'est donc utilisée que pour les concepts qui n'ont pas pu être appariés par ces autres techniques, donc sur un nombre limité de concepts.

Si aucune ontologie ne permet l'ancrage simultané des deux concepts à apparier, l'approche précédente peut être étendue récursivement en travaillant sur plusieurs ontologies à la fois. Les auteurs proposent ainsi d'ancrer X_{Src} dans une première ontologie, puis de rechercher, pour tous les concepts Y_{BK} en relation avec l'ancre dans cette ontologie s'ils sont en relation avec Y_{Tar} dans d'autres ontologies. Même si elle peut être parallèlisée, cette dernière stratégie est bien sur encore plus coûteuse que la précédente.

2.4 Utilisation de connaissances de support dans *TaxoMap*

Comme dans Aleksovski et al. (2006a, 2006b**)**, nous n'utilisons à ce jour dans notre approche qu'une seule ressource support O_{BK} identifiée manuellement au préalable, en l'occurrence WordNet. Bien évidemment, les ontologies à apparier étant très spécifiques et comportant des descriptions très fines du domaine d'application, avec des concepts très spécialisés, WordNet ne peut pas être considérée, et de loin, comme plus complète et plus détaillée que ces deux ontologies. Mais notre technique d'utilisation d'une O_{BK} n'étant comme dans Sabou et al. (2006) qu'une technique complémentaire à d'autres techniques d'appariement, la plupart des appariements portant sur les concepts très spécialisés auront été identifiés par les autres techniques et cette dernière ne s'appliquera que sur les concepts de X_{Src} non encore appariés.

Par rapport aux autres travaux, du fait de notre contexte d'application, nous limitons le nombre de relations à identifier, en ne cherchant à apparier les concepts de O_{Src} qu'avec des concepts de O_{Tar} considérés comme plus généraux, i.e. nous recherchons des mappings orientés de la forme $X_{Src} \leq Y_{Tar}$ et pas ceux de la forme $X_{Src} \geq Y_{Tar}$

Notre approche est donc la suivante : nous commençons par identifier manuellement avec un expert, le concept de WordNet noté $root_A$, qui sera le concept le plus spécialisé de WordNet qui généralise a priori tous les concepts du domaine des ontologies à apparier (*food* dans notre exemple). Puis nous réalisons l'ancrage dans WordNet de tous les concepts de O_{Tar} et de l'ensemble des concepts de O_{Src} non encore appariés.

Notre technique se différencie sur la recherche des dérivations. Au lieu de rechercher les dérivations entre les ancres des deux ontologies, nous recherchons d'abord les dérivations

qui mènent à la racine $root_A$ précédemment identifiée. Ces dérivations sont construites en recherchant dans WordNet les hypernymes de chacune des ancres, jusqu'à atteindre $root_A$ ou l'une des racines de la hiérarchie WordNet. Par exemple, le résultat de la recherche sur le concept cantaloupe donne les deux ensembles de généralisants suivants qui forment deux dérivations correspondant à deux sens différents du terme :

Sens 1: cantaloupe → sweet melon → melon → gourd → plant → organism → Living
Sens 2 : cantaloupe → sweet melon → melon → edible fruit → green goods → food

Seules les dérivations contenant $root_A$ sont retenues car elles correspondent au seul sens pertinent pour l'application. Un sous-graphe, T_{WN}, composé de l'union des concepts et des relations des dérivations sélectionnées (cf. FIG. 1) est alors obtenu. Il se compose du concept racine le plus général de l'application, $root_A$, des feuilles correspondant aux ancres des concepts issus des deux ontologies initiales (cercles sur FIG.1) et des généralisants intermédiaires extraits de WordNet qui peuvent, ou non, appartenir à l'une des deux ontologies.

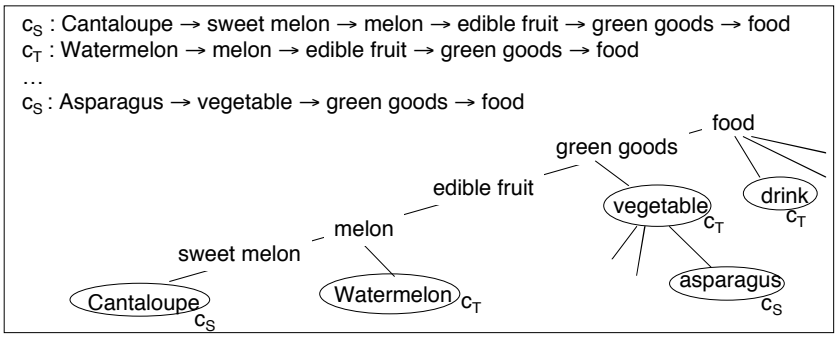


Fig 1. Un exemple de sous graphe T_{WN}

L'identification des mappings pertinents est faite en recherchant pour chaque concept c_S de O_{Src} le plus proche concept c_T de O_{Tar} qui apparaisse sur une dérivation menant à la racine $root_A$. Ainsi à partir du sous graphe de la figure ci-dessus, on peut dériver le mapping asparagus *isA* vegetable. Notre technique est comparable dans ces résultats à celle mise en œuvre dans Aleksovski et al. (2006b), si ce n'est que nous ne travaillons que sur les relations de type *isA* de WordNet (et pas sur les liens *part-of*) et que nous ne conservons que les mappings orientés de la forme c_S *isA* c_T.

Remarquons qu'aucune des techniques que nous venons de présenter ne permet de dériver de mapping sémantique pour le concept cantaloupe puisque aucun de ses ancêtres n'est une ancre d'un concept de O_{Tar} (tous sont des termes intermédiaires issus de WordNet). En revanche, on aimerait bien être capable de le « rapprocher » du concept Watermelon puisque ces deux concepts sont deux sortes de melon et donc sémantiquement très proches.

Ce rapprochement peut être effectué en utilisant sur le sous graphe T_{WN}, une mesure de similarité entre nœuds d'un même graphe. Nous utilisons la mesure proposée par Wu et Palmer (1994) selon laquelle la similarité entre deux nœuds c_1 et c_2 est fonction de leur profondeur, depth(c_i), $i \in [1,2]$, i.e. leur distance à la racine en nombre d'arcs, et de celle de leur plus petit ancêtre commun (*LCA*).

$$Sim_{W\&P}(c_1, c_2) = \frac{2 * depth(LCA(c_1, c_2))}{depth(c_1) + depth(c_2)}$$

L'intérêt de cette mesure est double. D'une part, elle plus précise qu'une mesure basée sur une simple distance des nœuds, car plus la profondeur du *LCA* de deux concepts est importante, plus les deux concepts partagent de caractéristiques communes et plus ils sont proches. D'autre part, une étude de ses propriétés, cf. Reynaud et Safar (2006b), nous a permis d'identifier et de mettre en œuvre une stratégie de recherche qui permet de retrouver très efficacement dans T_{WN} le concept c_T de O_{Tar} qui sera évalué comme le concept le plus similaire d'un concept donné c_S de O_{Src}.

Il est clair que les rapprochements effectués à partir d'une mesure de similarité de ce type ne permettent pas d'établir de mappings « sémantiques », c'est-à-dire reliant explicitement deux concepts par un lien de type *isA* ou *Eq*, et qui puissent être justifiés et prouvés par des mécanismes d'inférences cf. Sabou et al. (2006). Il est tout aussi clair qu'il serait dommage de ne pas exploiter l'information identifiée ! Nous proposons donc de retenir les rapprochements de ce type comme des mappings potentiels devant être validés par un expert et de les étiqueter par une nouvelle relation notée '*isClose*'.

Le choix fait dans *TaxoMap* consiste donc à rechercher pour chaque concept c_S de O_{Src} qui restait à apparier, le concept c_T de O_{Tar} qui lui est le plus similaire suivant la mesure de Wu et Palmer, qui sera noté c_{Sim}, et de construire le mapping potentiel associé c_S *isClose* c_{Sim} Puis nous extrayons, comme nous l'avons décrit plus haut, l'ensemble des mappings sémantiques lisibles sur les branches du sous graphe T_{WN}. Si un concept c_{Sim} apparaît relié à un même concept c_S à la fois dans un mapping sémantique et dans un mapping potentiel, nous ne conservons que le mapping sémantique.

Par exemple, le concept vegetable de O_{Tar} étant le concept le plus similaire du concept asparagus de O_{Src}, nous construisons le mapping potentiel asparagus *isClose* vegetable. Mais comme nous avons aussi pu construire le mapping sémantique asparagus *isA* vegetable seul ce dernier est conservé. Comme aucun mapping sémantique ne peut être construit pour le concept cantaloupe, nous conserverons le mapping potentiel cantaloupe *isClose* Watermelon.

Nous avons réalisé deux expérimentations de cette technique dans le domaine de la microbiologie. Dans la première, la technique a été utilisée directement sur tous les concepts de O_{Sr} en utilisant une technique d'ancrage du type inclusion de labels et dans la deuxième, la technique a été utilisée en complément d'autres techniques (donc sur les seuls concepts non encore appariés).

La non pertinence des résultats obtenus dans la première expérimentation est largement due à la longueur des labels des concepts de notre domaine (ex : home-style salad (reduced calorie mayonnaise with chicken)) qui ne sont bien sur pas reconnus directement par WordNet et qui sont incorrectement ancrés par l'heuristique d'inclusion de labels (les 3 ancres identifiés pour l'exemple précédent sont : salad, mayonnaise, chicken). En revanche, dans la deuxième expérimentation, comme les autres techniques de *TaxoMap* tirent justement parti de la longueur des labels pour exploiter leur similarité, la technique n'a du être appliquée que sur les seuls concepts non encore appariés, avec plus souvent des labels courts, et les résultats sont plus pertinents.

D'autres expérimentations ont été réalisées sur des taxonomies servant de test dans la communauté appariement. Elles ont montré que cette technique n'est pas adaptée pour

aligner des taxonomies dont le domaine d'application est trop large. En effet, la technique construit un sous-arbre à partir de tous les nœuds hypernymes de WordNet jusqu'à atteindre le nœud le plus général de l'application. Dans le cas d'un domaine très grand, le concept le plus général est un nœud placé très haut dans la hiérarchie WordNet, si ce n'est le nœud racine. T_{WN} est donc très gros. Il mêle des sens de termes différents et conduit à générer des mappings qui ne sont absolument pas pertinents. Des améliorations seraient possibles si plusieurs sous-arbres étaient construits, un par sous-domaine en supposant que les différents sous-domaines puissent être identifiés.

3 Conclusion et perspectives

Les différents travaux présentés montrent bien l'intérêt d'utiliser des connaissances supplémentaires pour la découverte automatique de mappings. La comparaison effectuée nous a permis d'identifier les similitudes et les complémentarités des différentes approches, ainsi que de possibles directions de recherche. Par exemple, nous pourrions essayer de valider les mappings potentiels identifiés dans notre approche en utilisant la technique proposée par Sabou et al. Inversement, ces mapping potentiels pourraient être utilisés dans les travaux de Sabou, dans la phase de recherche récursive pour diriger les recherches et ordonner les tests effectués.

Ces comparaisons nous ont permis de mettre en lumière quelques problèmes posés par le passage à l'échelle. Le recours à des connaissances externes est très intéressant lorsqu'on connaît précisément le contexte au sein duquel les éléments manipulés doivent être interprétés. Il est plus délicat lorsque le contexte est plus large ou inconnu au départ. Ainsi des techniques génériques, applicables quel que soit le domaine d'application et ayant recours de façon dynamique à de telles connaissances externes, sont face à un réel problème d'identification du contexte d'étude. C'est le cas lorsque la connaissance de support est obtenue via Swoogle ou lorsqu'on utilise WordNet.

Le deuxième problème identifié au travers des travaux présentés concerne le traitement des relations présentes dans l'ontologie supplémentaire utilisée. Comment tirer parti de toute la richesse des relations représentées dans une ontologie ? Doit-on s'autoriser à combiner des relations sémantiques différentes, si oui comment ? Comment interpréter les résultats obtenus suite à ces combinaisons ?

Références

Aleksovski, Z., Klein, M., Ten Kate, W., Van Harmelen, F. (2006a). Matching Unstructured Vocabularies using a Background Ontology, Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW'06), Springer-Verlag.

Aleksovski, Z., Klein, M., Ten Kate, W., Van Harmelen, F. (2006b). Exploiting the Structure of Background Knowledge used in Ontology Matching. ISWC'06 Workshop on Ontology Matching (OM-2006), Athens, Georgia, USA.

Kéfi, H. (2006). Ontologies et aide à l'utilisateur pour l'interrogation de sources multiples et hétérogènes. Thèse de doctorat de l'Université Paris-Sud.

Kéfi, H., Safar, B., Reynaud, C. (2006). Alignement de taxonomies pour l'interrogation de sources d'information hétérogènes. RFIA. Tours.

Lin, D. (1998). An Information-Theoretic Definition of Similarity. ICML, Madison, pp. 296-304.

Reynaud, C., Safar, B. (2006a) When usual structural alignment techniques don't apply. ISWC '06 Workshop on Ontology Matching (OM-2006), Poster, Athens, Georgia, USA.

Reynaud, C., Safar, B. (2006b). Structural Techniques for Alignment of Taxonomies: experiments and evaluation, In TR 1453, LRI, Université Paris-Sud, Juin 2006.

Sabou, M., D'Aquin, M., Motta, E. (2006). Using the Semantic Web as Background Knowledge for Ontology Mapping, ISWC'06 Workshop on Ontology Matching (OM-2006), Athens, Georgia, USA.

Wu, Z., Palmer, M. (1994). *Verb semantics and lexical selection*. Computational Linguistics. Las cruces, pp. 133-138.

Summary

This paper deals with the alignment techniques between ontologies performed by the *TaxoMap* system. We focus on the applicability on a large scale and we compare one of our techniques with close works using background knowledge.

Vers une plateforme de gestion des schémas XML

Sana Sellami*, Nabila Benharkat*, Rami Rifaieh**, Youssef Amghar*
* LIRIS, National Institute of Applied Sciences of Lyon, Lyon, France
{sana.sellami, nabila.benharkat, youssef.amghar}@insa-lyon.fr
**San Diego Supercomputer Center, University of California San Diego, USA
rrifaieh@sdsc.edu

Résumé. Les dernières années ont été témoin de l'émergence de l'intégration des données et des challenges inhérents à cette problématique. Parmi ces derniers sont concernés la gestion des schémas, l'évolution des schémas, le mapping et le matching des schémas. Plusieurs algorithmes ont été proposés pour comparer les schémas XML et établir les degrés de similarité entre les éléments. Plusieurs approches ont été définies pour traiter les expressions de mappings et différents outils ont été conçus pour générer le Mapping entre les schémas XML et d'autres formats de données. Nous avons proposé d'étendre le processus de matching des schémas XML à la gestion des contraintes en proposant un algorithme traitant l'aspect de conformité aux contraintes. Nous avons également défini le processus de mapping des schémas XML et nous avons proposé XME (XML Mapping Expression) un modèle d'expression de mapping et un ensemble d'opérateurs de transformation. Toutes ces extensions seront explicitées au travers de l'architecture ASMADE (Automated Schema Matching For Documents Exchange) qui permettra d'offrir un cadre d'application pour automatiser les transformations des instances des schémas et une plateforme pour la gestion des schémas qui résiste au passage à l'échelle. Nous présentons également un prototype de cette plateforme.

1. Introduction

Avec l'arrivée de l'Internet et du Web, le nombre de sources d'informations interconnectées ainsi que le nombre d'utilisateurs potentiels de ces sources a connu une augmentation exponentielle durant les dix dernières années. Le monde informatique regorge ainsi de données aux formats très hétérogènes, autrement dit utilisent des modèles différents pour la représentation de l'information, qu'il est nécessaire d'intégrer pour construire des applications. Dans un tel contexte, le besoin d'intégration se fait de plus en plus sentir.

De nombreuses technologies ont permis de faire communiquer des applications relevant de systèmes d'informations différents permettant ainsi d'atteindre un niveau d'intégration qu'il s'agit toutefois d'étendre et d'améliorer. En dépit de nombreux outils disponibles sur le marché, le problème d'intégration des données demeure entier en particulier dès que l'on passe à un contexte fortement dynamique et à large échelle.

Nous nous positionnons ici dans le cas de l'intégration des données et nous cherchons à améliorer la transformation des documents XML. Nous nous sommes donc intéressés dans ce travail à étudier un aspect d'optimisation des algorithmes de matching et plus précisément l'algorithme EXSMAL par Rifaieh et al. (2005), pour le passage à l'échelle, en réalisant son extension aux contraintes. Nous avons également défini le processus de Mapping des schémas XML pour lesquels il serait possible de réutiliser telles quelles les technologies déjà existantes et nous avons proposé un modèle d'expression de mappings et un ensemble d'opérateurs de transformation pour couvrir tous cas utiles de génération d'éléments cibles

de schémas XML à partir d'un ensemble de fonctions de mapping. Ces deux processus qui se suivent sont des pré-requis à l'intégration et la transformation de documents XML pour la réalisation d'un prototype offrant une plateforme de gestion des schémas qui résiste au passage à l'échelle.

Le présent article est organisé comme suit. Dans la partie qui suit, nous présenterons un état de l'art des travaux existants concernant l'aspect des contraintes dans le processus de matching et la technique de mapping. Dans la partie 3, nous exposerons les approches proposées qui sont l'extension du processus de matching à la prise en compte des contraintes, la proposition du modèle XME (XML Mapping Expression) et l'architecture ASMADE qui regroupera ces deux propositions. Nous terminerons finalement par une conclusion et quelques perspectives.

2. Etat de l'art

L'automatisation de l'intégration des données et le matching des schémas sont des problématiques qui ont fait l'objet de plusieurs études. Nous avons distingué deux catégories d'approches : le matching et le mapping de schémas XML. Le problème essentiel dans la mise au point de ces techniques est lié à l'hétérogénéité et à la diversité des sources d'informations.

2.1 Etat de l'art sur le Matching des schémas

Plusieurs travaux ont été réalisés afin de fournir des algorithmes de Matching gérant les correspondances ou incompatibilités des schémas. Nous nous sommes spécialement intéressés aux algorithmes de matching qui traitent les contraintes. Dans la littérature, Cupid Madhavan et al. (2001) traite les contraintes de schémas telles que les types de données et les rangs de valeurs, les cardinalités, etc. Similarity Flooding proposé par S.Melnik et al. (2002) traite des contraintes qui sont basées sur les clefs primaires, clefs uniques, valeurs non nulles ou encore contraintes référentielles qui se reportent aux clefs étrangères et contraintes de cardinalité. Dans le cadre de notre travail, nous avons travaillé sur l'algorithme EXSMAL Rifaieh et al. (2005). Nous avons cherché à étendre cet algorithme à la gestion des contraintes en réalisant une étude sur les différentes contraintes reliées aux schémas XML comme décrites par Buneman et al. (2001).

2.2 Etat de l'art sur les modèles et outils de mapping

Dans une seconde catégorie, le processus de mapping consiste à déterminer la vraie relation sémantique entre les éléments. C'est un processus qui suit le processus de matching. Plusieurs modèles ont été proposés, à cet effet, pour exprimer comment les éléments matchés (mis en correspondance) peuvent être mappés (transformés). Nous avons dégagé plusieurs modèles de mapping existants tels que les approches basées sur les valeurs de correspondances par Miller et al. (2000), Rifaieh (2004) ou encore sur des opérateurs de transformation proposés par Boukottaya et al. (2004), Zerdazi et Lamolle (2005). Nous avons également réalisé une étude comparative (Sellami, 2006) entre les différentes plateformes et outils de mapping existants sur le marché et nous avons identifié les limites de chacun d'eux.

Nous avons conclu que tous les outils ne réalisent pas simultanément le matching, mapping et la génération de code.

3. Les approches proposées

La première motivation de notre travail est de fournir une plateforme qui enchaîne les processus de Matching et de Mapping afin de minimiser les interventions humaines, assurer la gestion des schémas et fournir une méthodologie de partage de données qui résiste au passage à l'échelle permettant de comparer et d'intégrer un très grand nombre de sources d'information.

3.1 Extension du processus de matching à la gestion des contraintes

Nous avons cherché à réaliser l'extension de l'algorithme, développé au sein de l'équipe, EXSMAL (voir Rifaieh et al., 2005), algorithme de matching semi automatique, aux contraintes. La solution proposée dans EXSMAL permet de découvrir les correspondances sémantiques entre les différents schémas des messages EDI en prenant en compte des descriptions des types de données et de leurs éléments. En revanche, elle ne traite pas les contraintes, les statuts et la cardinalité associée à chaque élément de message. Cependant, La recherche des correspondances entre les schémas XML lors de la phase de matching prend en compte divers types de contraintes. La prise en considération et le traitement des contraintes est un des aspects d'optimisation des algorithmes de matching afin de fournir des correspondances plus précises entre plusieurs schémas. En effet, ces contraintes de matching peuvent affiner la sémantique des éléments des schémas XML. Ainsi, les informations sur les éléments peuvent être utilisées comme des « discriminants » pour déterminer la probabilité que deux éléments correspondent. Nous avons donc proposé une classification de ces contraintes. En première catégorie, nous avons défini les contraintes intrinsèques qui sont prédéfinies et se rapportent généralement aux informations liées aux éléments des schémas. En seconde catégorie, nous avons défini les contraintes de processus qui encodent une connaissance supplémentaire sur le domaine et les correspondances sémantiques entre les schémas dans le domaine. Elles sont typiquement spécifiées par l'utilisateur ou par les experts du domaine. Nous avons également proposé un algorithme traitant l'aspect de conformité du matching aux contraintes et déterminant le degré de violation des contraintes de matching.

3.2 XME : XML Mapping Expression

Nous avons proposé par la suite XME (XML Mapping Expression) un modèle d'expression de mapping qui permet d'exprimer théoriquement des règles de transformations utilisables pour exploiter les correspondances générées par le processus de matching. Dans ce modèle, nous respecterons la spécificité du modèle d'expression de mapping défini par Rifaieh (2004) en précisant davantage la génération des éléments/attributs cibles afin d'offrir un modèle applicable sur tous les cas de mappings entre plusieurs schémas et leurs éléments.

Nous avons défini et décrit ce modèle à partir d'un ensemble d'opérateurs de transformation pour couvrir les différents cas de Mapping facilitant ainsi la génération du Mapping entre les éléments.

3.3 ASMADE (Automated Schema Matching For Documents Exchange)

Nous avons pu offrir, suite à ces propositions, une nouvelle dimension à l'architecture ASAMDE (Automated Schema Matching For Documents Exchange) par Benharkat et al. (2006) en réalisant son extension à la gestion des contraintes et la transformation de documents. Cette architecture permettra d'offrir un cadre d'application pour automatiser les transformations des instances des schémas et une plateforme pour gérer des schémas à large échelle. Nous avons conçu et réalisé un prototype d'ASMADE en intégrant le code source de l'algorithme EXSMAL. Ce prototype offre de nouvelles spécificités telles que l'utilisation du drag et drop des objets représentant des schémas, des fonctions de Mapping, des conditions de Mapping, le choix des schémas multiples.

4. Conclusion et perspectives

Ce travail vise à offrir une nouvelle vision sur les processus de matching et de mapping des schémas XML prenant en compte le passage à l'échelle de ces techniques. Nous avons proposé une extension du premier processus à la gestion des contraintes et nous avons contribué à l'extension du second pour permettre la transformation des documents XML. Nous estimons par ailleurs que, même si les différents contributions ont un impact sur l'intégration et la gestion des données, il faut intégrer dans des travaux futurs l'algorithme de gestion des contraintes (intrinsèques et de processus), améliorer le prototype de ASAMDE dans le but de fournir, à long terme, une plateforme et une méthodologie de partage et gestion des données qui résiste au passage à l'échelle.

Références

J.Madhavan, P. A. Bernstein and Rahm (2001).*Generic Schema Matching with Cupid.* Proceedings of the 27th VLDB Conference, Rome, Italy, pp.49-58.

S.Melnik, H.Garcia-Molina and E. Rahm (2002). *Similarity Flooding: A versatile Graph Matching approaches*. Proceeding (ICDE), San Jose, Californie, USA.

Rami Rifaieh, Uddam Chukmol, Aïcha-Nabila Benharkat (2005). *A Matching Algorithm for Electronic Data Interchange*. (TES): 34-47

Renée J.Miller, Lara M.Haas, Mauricio A.Hernandez (2000*). Schema Mapping as Query Discovery*. Proceeding of the 26th VLDB Conference, Caire, Egypte,

Rami Rifaieh (2004). *Utilisation des ontologies contextuelles pour le partage sémantique entre les systèmes d'information dans l'entreprise*. Thèse de doctorat, INSA de lyon,.

A.Boukottaya, C.Vanoirbeek, F.Paganelli, O.Abou Khaled (2004). *Automating XML Transformations: A conceptual modelling based approach* . Media Research Group, Lausanne, Switzerland.

Amar Zerdazi, Myriam Lamolle (2005*). HyperSchema XML: Un modèle d'intégration par enrichissement sémantique de schémas XML*. Université Paris 8.

Sana Sellami (2006). *Conception et réalisation d'un outil de génération automatique de mapping pour la transformation de documents XML*. Mémoire de PFE.

Nabila Benharkat, Rami Rifaieh, Herzi Khaled, Youssef Amghar (2006). *ASMADE: Automated Schema MApping for Documents Exchange*. Software Engineering and Data Engineering (SEDE), Los Angeles, Californie .

Peter Buneman, Weinfei Fan, Jerome Simeon, Scott Welnstein (2001). *Constraints for Semistructured Data and XML*. Proc SIGMOD Record, vol 30, No1.

Summary

Information systems design is increasingly multi-source. These data sources are heterogeneous, distributed and accessible by the Web or by an enterprise network. The heterogeneity problem is the principal difficulty in this design. We can point up problems such: schema integration, generating Mapping expression and data transformation. For this purpose, several algorithms have been proposed to compare the XML schemas and to establish similarity degrees between the elements. Many approaches have been defined to treat the Mapping expressions and different tools have been proposed for generating the Mapping between XML schemas and other data formats. In our work, we are interested in schema Matching and we propose to extend the matching process to constraints management. We define an algorithm which takes into consideration the matching constraints. We propose also XME (XML Mapping Expression) which is an expression model of mappings and we describe a collection of transformation operators. These extensions take part in our platform for document Exchange (ASMADE) that automates matching and generates data transformation instances.

A scalable approach for large-scale Schema Mediation

Khalid Saleem and Zohra Bellahsène

LIRMM - UMR 5506, Université Montpellier 2, 34392 Montpellier Cedex 5 - France
{saleem, bella}@lirmm.fr

Abstract. Nearly all schema-matching systems compare two schemas at a time and aim for quality matching with significant human intervention. For a limited number of schemas (less than 10) of small size (less than 100 nodes), the matching and integration processes give acceptable performance. The novelty of our method is fourfold. First, we support fully automated schema matching. Second, we generate approximate mappings and also build an integrated schema at the same time. Third, we show that our approach is scalable to hundreds of schemas, having large size. Fourth, we utilize tree mining techniques, clustering of similar node labels, for schema matching, which is a budding research approach.

Keywords : XML schema tree, schema matching, schema mapping, schema mediation, large scale schema integration, tree mining.

1 Introduction

The process of schema matching relies in discovering correspondences between similar elements of two schemas. There are different types of schema matching approaches, which have been studied at length by different researchers [2–5, 8, 12], to demonstrate their benefit in different scenarios. In today's application domain of data manipulation, schema matching is of central importance. The reason is the Web, which is expanding with enormous speed, and newer ways to store and extract data, are flourishing in parallel.

There are numerous issues in the semantic matching of schemas. The quality of match results depend on the algorithms (syntactic/ semantic) used, the strategy applied for combining the algorithms' results and the criteria for selecting the best match [2, 4, 8]. Besides mapping quality, performance is also very important. One has to trade-in between quality and performance, depending upon the application domain. Semantic Web, by definition, offers a large-scale dynamic environment where individual service providers are independent. In such a situation the mappings can never be exact, rather they are approximate [3].

The motivation behind our work is to explore the matching and integration of a large set of schema trees, using scalable syntactic and semantic matching

and integration techniques with performance. Our target is to develop an automated schema matching framework with approximate mapping and integration. We consider schemas as rooted, labelled trees, and nodes ordered using pre-order traversal. This supports the computation of contextual semantics in the tree hierarchy. The individual semantics of node labels have their own importance. We utilize linguistic matchers, to extract the concepts hidden within them and cluster together the similar labels. The contextual aspect is exploited by tree-mining techniques [15], applied on target search space presented by the related cluster of similar labels. Thus providing the automated approximate schema matching and integration in a large-scale scenario. Tree mining techniques by definition extract similar sub tree patterns from a large set of trees and predict possible extensions of these patterns. The pattern size starts from 1 and is incrementally augmented. There are different techniques, which mine rooted, labelled, embedded or induced, ordered or unordered sub-trees. The first basic function of tree mining is to find sub-tree patterns that are frequent in the given set of trees, which is similar to schema matching activity that tries to find similar concepts among a set of schema trees. Our main contributions are:

- Matching, merging and the creation of a mediated schema with semantically approximate mappings, automatically in one algorithm, which provides a scalable performance.
- Use of tokenisation, abbreviation and synonym matching of label tokens; thus clustering similar label
- Extension of a tree mining data structure [15] to schema matching, using related similar label cluster. Intuitively label cluster provides the target nodes cluster i.e., minimized target search space.
- Ability to produce element level 1:1 mappings [12], as well as 1:n (leaf mapped to non-leaf) and n:1 (non-leaf mapped to leaf) mappings.
- Experiments with real XML schema instances (OAGIS, XCBL)[1] showing scalable performance applicable for a large scale scenario and comparison with COMA++, showing that our approach is qualitatively similar.

The reminder of the paper is organized as follows. Section 2 gives the detail of our approach for large scale schema mediation. Section 3 presents the prototype implementation of our approach with a running example. In Section 4 we enumerate the experiments and their results. Section 5 outlines the related research in schema matching and integration and Section 6 concludes the paper.

2 Our Approach : Scalable large-scale Schema Mediation

Schema Mediation can be defined as integration of as set of input schemas into a single schema, with mappings between input schema elements and the mediated schema elements. The elements participating in respective correspondences are considered to represent similar concept. The similarity factor is calculated by

[1] oasis-open.org and www.xcbl.com

using a range of syntactic algorithms [12] like edit distance, n-gram etc. and semantic strategies using external oracles like Wordnet dictionary or some high level ontology [11]. Another important technique in semantic similarity measuring is the element's context within the schema i.e., elements placing in the schema and their relation to the neighbouring elements. The benefit of this aspect has been very successfully demonstrated in structure oriented matching tools [2, 9].Our approach for matching schemas is driven by hierarchical tree structures of schemas since our data model for mediation is XML schemas.

2.1 Semantic Label Matching

In any matching system the first operation is the element/node label comparisons. There are numerous algorithms which syntactically compare similarity of two label strings. We use the linguistic tokenization technique augmented with semantic method of synonym and abbreviation handling. Using this approach, we create clusters of similar labels used in the set of schemas to be integrated. Each participating node is linked to its respective label in the clusters. Intuitively this minimizes the search space of possible mappable target nodes. We consider each label as a combination of tokens and this combination represents some inherent concept. Semantic similarity between labels is either equivalence or partial equivalence [5], as shown below:

a. Equivalence: Concept (lx) = Concept (ly) Similar
b. Partial Equivalence: Concept(lx) $\cong$ Concept(ly)
   i. More Specific or Is part of : Concept(lx) $\subseteq$ Concept(ly)
   ii. More General or Contains : Concept(lx) $\supseteq$ Concept(ly)
   iii.Overlaps : Concept(lx) $\cap$ Concept(ly)

Example 1: Consider labels AuthorName and WriterName. Since Author and Writer are synonyms and Name is shared, conceptually they are equivalent, thus AuthorName = WriterName. Similarly, AuthorLastName is $\subseteq$ of AuthorName, as LastName is conceptually part of Name. Conversely, AuthorName is $\supseteq$ of AuthorLastName. MiddleLastName and FirstNameMiddle are *overlapping*, as their $\cap$ results in {Name, Middle}.

2.2 Tree-mining Aspect

Semantic Schema tree matching requires the comparison of concepts which are structured as schema nodes. Each nodes's contextual placement in the schema tree enhances the semantics of the concept.

Example 1: In Figure 1, node author/*name* and publisher/*name* are similar labels but their contexts are different, which makes the two nodes conceptually disjoint. thus in a XML tree, the combination of the node label and the structural placement of the node produce the concept.

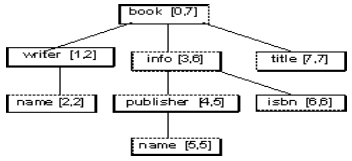


Fig. 1. XML schema tree with labels and [node number, node scope] for each node

For node placement, we have to order the tree nodes. We order each schema tree using pre-order traversal (Fig. 1), since XML schemas also follow the same ordering method. The creation of data structure containg list of all distinct node labels, extracted from the input schema trees, and nodes linked to respective labels, along with node placement detail (Fig. 3) has been adapted from tree mining research [15]. In our approach the distinct label is taken as node labels with in same schema tree with different contextual placement i.e., different parent nodes. Therefore the count for a label, with similar string value, in labels list, is the maximum number of occerance of that label in a certain schema tree, for whole set of input schema trees.

2.3 Definitions

Definition 1 (Schema Tree) : A schema is a rooted, labelled tree [15]. We call it a *schema tree*. A schema tree, S=(V,E) is a directed, acyclic, connected graph, with V={0,1,. . . ,n}, a set of nodes, and E={(x,y) | x,y ∈ V}, a set of edges. One distinguished node r ∈ V is designated the root, and for all x ∈ V, there is a unique path from r to x. Further, l:V → L is a labelling function mapping nodes to labels in L={l1,l2,. . . }, and VT:L → Vi is a function, which returns a set of nodes Vi⊆V for each label l∈ L i.e., set of nodes with labels similar to label l.
Definition 2 (Node Scope) : Since the schema tree is ordered (Section 2.3), nodes x ∈ V are numbered according to their position in the de pre-order traversal of the tree S (for example, the root is numbered 0). Let T(x) denote the sub-tree rooted at x, and let y be the rightmost leaf (or highest numbered descendant) under x. Then the scope of x is defined as scope(x)=[x,y]. Intuitively, scope(x) is the range of nodes under x, and includes x itself (Fig. 1). The count of nodes in T(x) is y-x+1.
Definition 3 (Node Semantics): Node semantics for node x ∈ V is given as Cx. It is a combination of the semantic concept of the node label C(lx) with its contextual placement in the tree TreeContext(x) [15].

Cx: x→(C(lx), TreeContext(x))

C(lx) is the composition of individual meanings of tokens making up the label l of node x. TreeContext of a node x is calculated using the node number and scope.

2.4 Scope Properties

Scope properties give us the contextual placement of a node in the tree and are explained in detail in [15]. The properties are simple integer operations.

Unary Properties, given a node x[X,Y]; P1) Leaf Node(x) : X=Y and P2) Non-Leaf Node(x): X < Y. For **Binary Properties**, given x [X,Y], xd[Xd,Yd], xa[Xa,Ya], and xr[Xr,Yr]; P3)Descendant (x,xd) - xd is a descendant of x : Xd>X and Yd≤Y, P4) Descendant Leaf (x,xd) - (combination of Property 1 and 3) : Xd>X and Yd≤Y and Xd=Yd, P5) Ancestor (x,xa) - (complement of Property 3) xa is ansector of x : Xa<X and Ya≥Y and P6) Right Hand Side Node (x,xr) with Non-Overlapping Scope - xr is Right Hand Side Node of x : Xr>Y.

Example 2 : In Figure 1, Property 1 for node **title**[7,7] defines it as a leaf because the node number equals the number of its rightmost child. Property 2, for **publisher**[4,5] defines it is a non-leaf node, as its number is less than the number of its rightmost child. Properties 1 and 2 detect simple and complex elements in an XML schema. Another task can be taken as, to find in the tree nodes matching of author/**name**. In Figure 1 there are two nodes called **name**: [2,2] and [5,5]. Given synonymy between words **author** and **writer**[1,2], we perform the descendant node check on nodes [2,2] and [5,5] with respect to **writer**[1,2]. Node [2,2] is a descendant of [1,2], using Property 3, and node [5,5] is not a descendant of [1,2]. Similarly, property 5 produces Ancestor([4,5],[5,5]) which holds for **publisher**[4,5] and **name**[5,5] and Ancestor([0,7],[4,5]) holds for **book**[0,7] and **publisher**[4,5].

2.5 Assumptions

a) Schemas in the same domain contain the same domain concepts, but differ in structure and concept naming.
b) In one schema different labels for the same concept are rarely present.
c) Only one type of matching between two labels is possible. For example, author is a synonym of writer.
d) We select the input schema with the highest number of nodes as the initial mediated schema. Since each node represents a concept, this covers the maximum number of concepts. This minimizes the addition of new concepts (nodes not present in the mediated schema) to the mediated schema and improve performance.
e) We perform semantic comparisons between the labels of the mediated schema and the labels not present in the mediated schema (based on assumption b). This minimizes the target search space for similar labels.
f) A node from the input schema is only matched to the cluster of similar label nodes present in the mediated schema.

3 Prototype with Running Example

The prototype application, developed in Java, works in three steps. First, schema trees are input to the system as a stream of XML, where **PreMapper** module calculates scope and node number for each of the nodes in the input schema trees. A listing of nodes and a list of distinct labels for each tree is constructed. In secont stage, **Label** module, a linguistic matcher identifies semantically distinct node labels in the labels list. It uses abbreviation table and tokenizes the labels. Then it derives the meaning for each individual token and combines these meanings to form a label concept. The comparisons of labels are based on similar token sets or similar synonym token sets.

Example 4 : Consider labels "POShipDate" and "PurchaseOrderDeliverOn". In the abbreviation table PO stands for purchase, order and in the synonym table 'deliver'='ship' and 'on'='date'. This implies that the two labels have similar token sets.

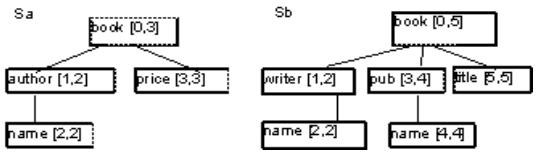


Fig. 2. Two input Schema Trees

In third step,**NodeMapper** module first select the input schema tree with the highest number of nodes and designate it as the initial mediated schema (Sec. 2.6). Next it takes each schema in turn and merges it to the mediated schema. This requires matching, mapping and merging. Concepts from input schemas are mapped to the mediated schema. The algorithm traverses the input schema pre-order depth-first, mapping parents before siblings from left hand side. If a new concept is found, with no match in the mediated schema, a new concept node is created and added to the mediated schema (Def. 4). It is the right most child leaf node added to the node in the mediated schema to which the parent of current node is mapped. This new node is used as the target node in the mapping. The algorithm combines node label similarity and contextual positioning in the schema tree, calculated with the help of properties defined in Section 2.5. following is a running example of two schemas as given in Fig. 2.

Fig. 2 shows two trees after executing PreMapper. A list of labels created in this traversal is shown in Table 2a. The two nodes with the same label 'name' but different parents are shown to be disjoint. The last entry in the list is a label 'ROOT' for the root node of mediated schema. A matrix of size um is created, where u is the number of schemas and m the number of distinct labels in all

schemas i.e., the length of the label list (Table 2.b). In the matrix each row represents an input schema tree. Each non-null entry contains the node scope, parent node link and the mapping, which is initially null. Matrix columns are ordered according to the order of nodes in the label list.

The larger schema tree Sb, Fig. 2, is selected as the initial mediated schema Sm. A list of size m (Tab. 2.b), is created to hold Sm, assuming the same column order as in Table 1.a and 1.b. The node mapping algorithm takes the data structures in Table. 1 as input, and produces mappings shown in Table. 2.b and the integrated schema in Table 2.c. In the process, the input schema Sa is directly mapped to mediated schema Sb. The mapping is taken as the column number (Table 2.b column number) of node. Saving mappings as column number gives us the flexibility to add new nodes to mediated schema tree. Scope values of some existing nodes are affected because of addition of new nodes, but column numbers of all previous nodes remain the same. Thus intuitively none of the existing mappings are affected. All mappings in this case are handled as given in Contributions Section.

NodeMapper for input schema tree Sa (Table 1.b row 1) starts from node with

Table 1. Befor Node Mapper Execution

a. Labels List

0	1	2	3	4	5	6	7	8
author	book	name	name	price	pub	title	writer	ROOT

b. Input Schema Nodes' Matrix : Row 1 is Sa and Row 2 is Sb

1,2,0	0,3,-1	2,2,1		3,3,0				
	0,5,-1	2,2,1	4,4,3		3,4,0	5,5,0	1,2,0	

b. Initial Mediated Schema

	1,6,0	3,3,2	5,5,4		4,5,1	6,6,1	2,3,1	0,6,-1

*Each column entry is node number, scope, parent node number

label book and scope [0,3]. Since it is a non-leaf node with only one similar node in the mediated schema tree Sm (Sb in this example) i.e., node 1 at column 1. So mapping **1** is added in column 1 for Sa records; the mapping Sa[0,3]→ Sb[0,5] (Table 2.b). Information regarding mapping is also saved as '1.0' i.e., label 0 of schema tree 1. Next node to map from Sa is author[1,2], similar to writer[1,2] in Sb. Both nodes are internal nodes and the ancestor map check returns true since parent nodes of both are already mapped. The resulting mapping for label 0 is **7**. For label 2 'name', there are two possibilities, label 2 (column 2) and label 3 (column 3). Descendant(name,author) is true for node in column 2 and false for 3 from P3. Hence **2** is the correct map. The last node in Sa is price[3,3]. There is no node in mediated schema tree with a similar label, so a new node is added to mediated schema, recorded by an entry in the column with label 'price' in the mediated schema list (Table 2.c). This node is created as the right most

Table 2. After Node Mapper Execution

a. Labels List

0	1	2	3	4	5	6	7	8
author	book	name	name	price	pub	title	writer	ROOT

b. Mapping Matrix : Row 1 is Sa and Row 2 is Sb

1,2,0,**7**	0,3,-1,**1**	2,2,1,**2**		3,3,0,**4**				
	0,5,-1,**1**	2,2,1,**2**	4,4,3,**3**		3,4,0,**5**	5,5,0,**6**	1,2,0,**7**	

b. Final Mediated Schema

	1,7,0, 1.0,2.0	3,3,2, 1.2,2.2	5,5,4, 2.4	7,7,1, 1.3	4,5,1, 2.3	6,6,1, 2.5	2,3,1, 1.1,2.1	0,7,-1

*Column entry is node number, scope, parent node, map info

sibling of node in the mediated tree to which the parent node of current input node is mapped i.e., node with label 'book'. The scope and parent node link is accordingly adjusted for the new node. And a mapping is created from input node to this newly created target node.

The running example has demonstrated that how our technique can create a mediated schema with approximate mappings from a given set of 2 schema trees. In our experiments we have run this technique to mediate with scalable time performance.

4 Experiments

We examine both the performance and quality of schema matching. Performance is evaluated as the number of schemas or nodes processed versus the time required for matching, merging and mapping (Fig. 3 and 4). Quality of matches is compared with the match results produced by COMA++ [2]. The experiments are performed on a PC, Pentium 4-M, 1.80 GHz, 768 MB RAM, running Windows XP was used in this evaluation. We selected three sets of schema trees from different domains:

1. Dom. 1 : BOOKS (Synthetic Schemas 176 - Avg. nodes/schema 8, Max/ Min nodes 14/ 5, DL* 22)
2. Dom. 2: OAGIS (Real Schemas 80 - Avg. nodes/schema 1047, Max/ Min nodes 3519/ 26, DL 10132)
3. Dom.3 : XCBL (Real Schemas 44 - Avg. nodes/schema 1678, Max/ Min nodes 4578/ 4, DL 14003)

DL: Distinct labels : Nodes' labels with different strings and labels with similar string but different parent in same schema tree.

Our experiments show that the execution time for our program depends upon the number of schemas, which are being integrated, and appears to be quadratic in the number of nodes compared. Secondly it is the size of distinct labels in each

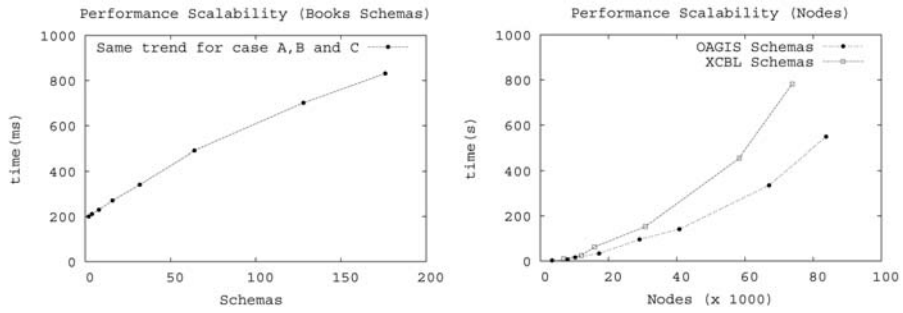


Fig. 3. a) Books Schema Integration b)Comparison of OAGIS/XCBL Schema

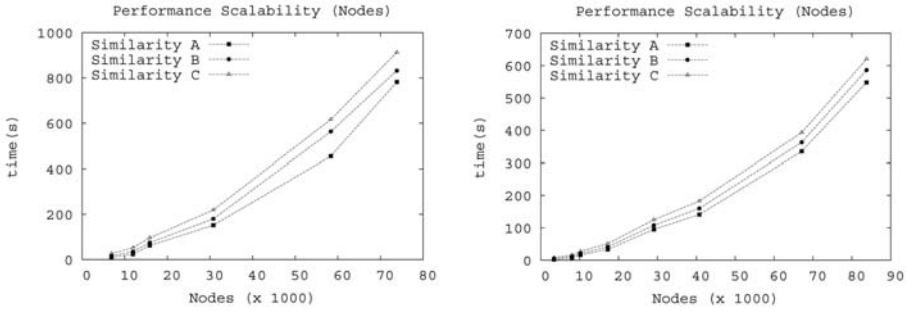


Fig. 4. a)Integration of OAGIS Schemas b) Integration of XCBL Schemas

experiment, which effected the time for similarity measure of labels (A: labels with same string, B: similar token set and C: similar synonym token set).

5 Related Work

Most of the research on schema matching and integration focus on achieving quality mapping between two small or medium size schemas [2–5, 8, 12]. Recent work on matching two large schemas has been presented in [10, 13] regarding large biomedical schemas. The other aspect of large scale scenario where the number of input schema is large have been addressed, for relational OO data model in [1], XML DTD schemas in [7] and for integration of obtaining an integrated interface for a set of web search forms in [6, 14].

6 Concluding Remarks

We presented a novel schema integration idea, which has shown very promising results for large scale schema integration. It is similar to other tools using

linguistic techniques like tokenization, use of abbreviations and synonym oracles. At present it uses very limited linguistic methods but the match quality is equivalent to other current tools as shown in comparison with COMA++. Our method uses the optimistic top down pre-order match traversal (parents are mapped before children, and left sub-tree is traversed before right sub-tree), since our assumption is that we utilize it in a domain specific environment. This helps in using the structural semantics of nodes for better quality matching. To improve performance we adapted a technique from tree mining including the clustering of similar node labels. This minimizes the target search space for a node match and gives better performance. Selection of the largest schema tree as the mediated schema further enhances the matching process. We report on experiments with up to 80 schemas ,which took 587 seconds to match, merge and create a mediated schema with mappings from input schemas to the mediated schema.

References

1. D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. The momis approach to information integration. In *ICEIS (1)*, pages 194–198, 2001.
2. H. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621, 2002.
3. A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Y. Halevy. Learning to match ontologies on the semantic web. *VLDB J.*, 12(4):303–319, 2003.
4. J. Euzenat and et al. State of the art on ontology alignment - technical report. *Knowledge Web*, 1.2, 2004.
5. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *Semantic Interoperability and Integration*, 2005.
6. B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *KDD*, pages 148–157, 2004.
7. M.-L. Lee, L. H. Yang, W. Hsu, and X. Yang. Xclust: clustering xml schemas for effective integration. In *CIKM*, pages 292–299, 2002.
8. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, pages 49–58, 2001.
9. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.
10. P. Mork and P. A. Bernstein. Adapting a generic match algorithm to align ontologies of human anatomy. In *ICDE*, 2004.
11. N. F. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.
12. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
13. E. Rahm, H. H. Do, and S. Massmann. Matching large xml schemas. *SIGMOD Record*, 33(4):26–31, 2004.
14. W. Su, J. Wang, and F. Lochovsky. Holistic query interface matching using parallel schema matching. In *ICDE*, 2006.
15. M. J. Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae 65*, pages 1–20, 2005.

Recherche de recouvrements dans une collection de schémas de bases de données

Ravi Ramdoyal, Anne-France Brogneaux, Julien Vilz, Jean-Luc Hainaut*

*Laboratoire d'Ingénierie des Bases de Données - Université de Namur
Rue Grandgagnage 21 - B-5000 Namur (Belgique),
rra, afb, jvi, jlh@info.fundp.ac.be,
http://www.fundp.ac.be/facultes/info/recherche/databases/

Résumé. Cet article présente nos pistes de recherche concernant l'analyse de recouvrements dans une collection de schémas de bases de données. Pour ce faire, nous introduirons les notions de redondance, de similarité et de pattern. Nous présenterons aussi quelques approches d'extraction de redondances à partir de schémas de bases de données. Cette problématique s'inscrit dans le cadre de ReQuest, une démarche explorant la possibilité de concevoir des systèmes d'information en exploitant de façon optimale les informations contenues dans les interfaces homme-machine.

1 Introduction

La démarche ReQuest (Brogneaux et al., 2005) vise à permettre la production plus rapide et à moindre coût de sites de commerce électronique de qualité, répondant de manière plus satisfaisante aux besoins des clients. L'un des aspects originaux de cette démarche repose sur l'implication active des utilisateurs finals dans l'expression de leurs besoins et l'élaboration du cahier de charges de leur nouveau site de commerce électronique.

A cette fin, les utilisateurs sont invités à dessiner eux-mêmes (éventuellement avec l'aide d'un analyste) un prototype simple de l'application qu'ils souhaitent, sous la forme d'interfaces homme-machine de type formulaire. Chaque interface de ce type est composée de blocs fonctionnels véhiculant de l'information et permettant de réaliser des tâches (composant de login, formulaire d'enregistrement, caddie électronique, ...). Le contenu informationnel de chacun de ces blocs fonctionnels peut être perçu comme une vue sur une base de données en cours de réalisation.

Dans la pratique, on peut constater que les informations qui se trouvent dans les blocs d'interface présentent une redondance élevée. Par exemple, les caractéristiques signalétiques d'un client (comme son nom ou son login) peuvent se retrouver dans plusieurs blocs fonctionnels. Dès lors, si on arrive à intégrer ces informations en identifiant et en résolvant les redondances, on obtient un schéma normalisé qui représente un sous-ensemble important du schéma conceptuel de la base de données. C'est dans cette optique que nous nous intéressons plus particulièrement au problème de l'identification de ces redondances. Nous laisserons donc de côté l'aspect "résolution", lequel est abordé dans Vilz et al. (2006).

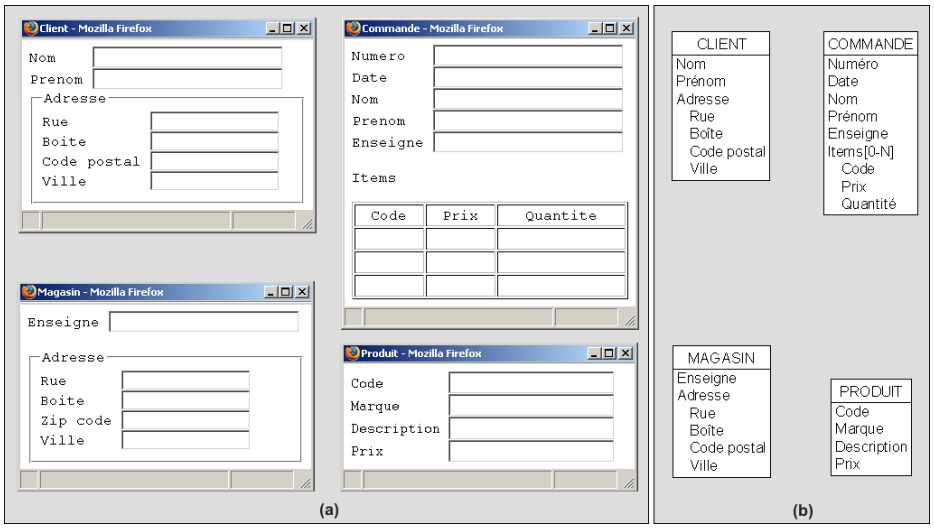


FIG. 1 – *Représentation de blocs fonctionnels d'interface sous la forme de types d'entité.*

2 Problématique

Comme les blocs d'interface possèdent une structure arborescente de champs, on peut les représenter sous la forme de types d'entités. En effet, ces derniers possèdent eux-mêmes une structure arborescente d'attributs. Une *redondance* s'exprime alors par deux ensembles d'attributs similaires.

Considérons l'exemple suivant pour illustrer le processus d'identification et de résolution de redondances (on se limite à la similarité du nom des attributs) : un utilisateur souhaite produire un système d'information simple destiné à gérer la commande de produits dans une chaîne de magasins. Dans ce contexte, un *client* passe une *commande* dans un *magasin* pour plusieurs *produits*. La figure 1 illustre le processus d'acquisition d'information : (a) l'utilisateur commence par dessiner les blocs fonctionnels d'interface dont il a besoin, puis (b) ceux-ci sont représentés sous la forme de types d'entité. La figure 2 illustre alors (a) la mise en évidence des différentes redondances de noms d'attributs, ainsi que (b) la résolution de ces redondances pour obtenir un schéma intégré.

Si le critère de nommage est le plus utilisé dans divers domaines afin de mettre en évidence des similitudes dans des arborescences (Chi et al., 2005), il n'est toutefois pas le seul : en effet, des attributs provenant de types d'entités différents peuvent aussi avoir la même taille, le même type de valeur, la même description sémantique, et ainsi de suite.

De ce constat, il ressort qu'on peut mesurer la *similarité de deux attributs* sur base d'une suite d'*indicateurs* (nom, taille, etc.). Pour chaque indicateur, on définit un *indice de similarité* compris entre 0 (absolument différent) et 1 (absolument identique). La similarité des deux attributs est alors une moyenne pondérée des indices de similarité pris en compte, et est elle-

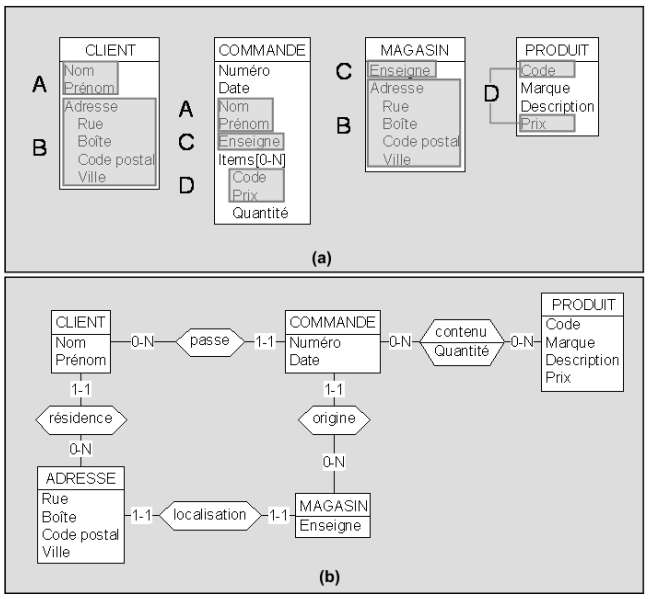


FIG. 2 – Identification et résolution de redondances de noms d'attributs. *(a) Les blocs d'interfaces sont représentés sous la forme de types d'entités. (b) Les différentes redondances de noms d'attributs sont mises en évidence. (c) En résolvant les redondances, on obtient un schéma intégré.*

même comprise entre 0 et 1. Par extension, nous définissons également la *similarité de deux groupes d'attributs* comme la moyenne des similarités d'attributs pris deux à deux.

Nous définissons alors un *pattern* comme une *classe d'équivalence* définie par la relation de similarité de groupes d'attributs. La similarité des membres d'un même pattern pris deux à deux doit atteindre au moins un certain seuil. L'instance représentative de chaque classe d'équivalence est calculée et validée sur base des membres de la classe. Par la suite, nous assimilerons un pattern à son instance représentative.

Afin d'extraire les patterns pertinents d'un schéma conceptuel pour traiter leurs similitudes, il faut donc suivre les étapes suivantes :

1. recherche des redondances
2. dérivation des patterns
3. validation des patterns par les utilisateurs
4. validation des patterns par l'analyste
5. résolution

Dans la suite, nous présentons donc quelques approches d'extraction de patterns que nous avons expérimentées.

3 Différentes approches d'extraction de patterns

3.1 Recherche de patterns arborescents

La première approche que nous avons expérimentée est abordée dans Vilz et al. (2006). Elle consiste à considérer chaque type d'entités comme un arbre et à procéder à la recherche des différents sous-arbres communs. Parmi les algorithmes développés pour résoudre ce problème complexe (Chi et al., 2005), nous avons opté pour l'algorithme FreqT exposé dans Asai et al. (2002). Sur base d'une approche de type "rightmost expansion" (Zaki, 2002), ce dernier produit une liste de patterns avec une structure arborescente ; nous appliquons alors la règle de pertinence suivante à la liste obtenue :

Soient P_i et P_k des patterns arborescents, où P_i est un sous-arbre de P_k conservant les relations père-fils. Soient I_i et I_k les ensembles des arbres (correspondant aux types d'entités) contenant des instances de P_i et P_k. On conserve alors P_i si et seulement si I_k est un sous-ensemble strict de I_i.

Cette règle est illustrée à la figure 3. Les patterns P_2 et P_3 y sont des sous-arbres du pattern P_1 ; les types d'entités E_1, E_2, E_3 contiennent des instances de ces classes d'équivalence. Comme I_1 et I_2 sont confondus, mais que I_1 est un sous-ensemble strict de I_3, on ne conserve que P_1 et P_3.

Cet algorithme est de complexité raisonnable (Asai et al., 2002) mais ne met pas en évidence toutes les similitudes souhaitées. Nous avons donc expérimenté une approche plus exhaustive.

3.2 Génération exhaustive de patterns non arborescents

Cette seconde approche consiste à examiner chaque type d'entités d'un schéma afin de générer tous les patterns qu'il contient par groupe d'attributs ayant le même parent. On n'obtient donc pas de pattern sous forme arborescente, mais sous forme de série, ce qui ne constitue pas une perte d'information, car la structure arborescente peut se retrouver récursivement.

Toutefois, cette génération étant exhaustive, il y a un grand nombre de patterns générés qui ne sont pas pertinents car ils présentent eux-mêmes des redondances. On propose donc d'appliquer successivement les règles suivantes à l'ensemble des patterns découverts :

1. *nombre d'instances* : il faut que les patterns possèdent au moins deux instances pour être significatifs ;
2. *largeur* : on conserve le pattern P au détriment du pattern R si et seulement si :
   (a) le pattern P recouvre le pattern R (i.e. P possède tous les éléments de R),
   (b) chaque instance de P recouvre une et une seule instance de R (i.e. chaque instance de P possède les éléments d'une et une seule instance de R),
   (c) le pattern P et le pattern R possèdent le même nombre d'instances.
3. *généricité* : on conserve le pattern P au détriment du pattern R si et seulement si :
   (a) le pattern R recouvre le pattern P,
   (b) chaque instance de R recouvre une et une seule instance de P,
   (c) le pattern P possède des instances non recouvertes par R.

Ce type de génération met en évidence plus de similitudes que l'approche précédente, mais il est de complexité exponentielle. Nous proposons donc une approche plus efficace.

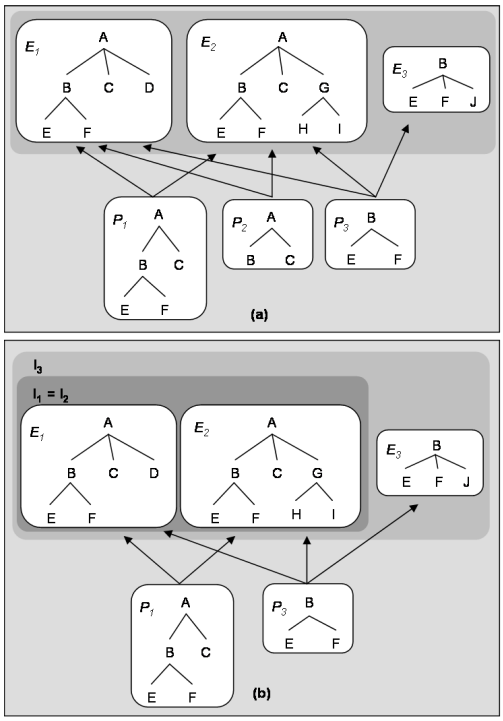


FIG. 3 – Elimination des patterns non pertinents. *(a) A partir des types d'entité E_1, E_2 et E_3, on extrait les patterns P_1, P_2 et P_3, où P_2 et P_3 sont des sous-arbres de P_1. (b) Les ensembles d'instances I_1 et I_2 sont confondus, alors que I_1 est un sous-ensemble strict de I_3. Selon la règle de pertinence, on conserve donc P_1 et P_3, et on élimine P_2.*

3.3 Génération sélective de patterns non arborescents

Cette méthode consiste également à générer des patterns par groupe d'attributs ayant le même parent, mais en prenant cette fois-ci les types d'entités deux à deux afin d'en générer les patterns communs. Au fur et à mesure de la génération deux à deux, les nouveaux patterns communs sont confrontés aux patterns déjà trouvés, selon les filtres en largeur et en généricité qui ont été présentés précédemment. Cette approche est en cours d'expérimentation.

4 Conclusion

Les résultats de la démarche ReQuest ont permis de déterminer et d'extraire les informations nécessaires contenues dans les interfaces conçues par l'utilisateur pour créer un schéma conceptuel du domaine de l'application. La méthode utilisée consiste tout d'abord à rechercher des redondances parmi les types d'entité extraits des blocs fonctionnels d'interface. Les pat-

terns qui en sont dérivés doivent alors être validés par les utilisateurs, puis par un analyste qui procédera ensuite à leur résolution. Dans la mesure où la démarche cible plutôt des systèmes d'information adaptés à de petites et moyennes entreprises, l'ensemble à partir duquel on extrait les redondances reste relativement petit, ce qui favorise la performance des algorithmes proposés.

Références

Asai, T., K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, et S. Arikawa (2002). Efficient substructure discovery from large semi-structured data. In *Proceedings of the Second SIAM International Conference on Data Mining*, pp. 158–174.

Brogneaux, A.-F., R. Ramdoyal, J. Vilz, et J.-L. Hainaut (2005). Deriving user-requirements from human-computer interfaces. In *Proc. of 23rd IASTED Int. Conf., Innsbruck, Austria*, pp. 77–82.

Chi, Y., R. R. Muntz, S. Nijssen, et J. N. Kok (2005). Frequent subtree mining - an overview. *Fundamenta Informaticae 66*(1-2), 161–198.

Vilz, J., A.-F. Brogneaux, R. Ramdoyal, V. Englebert, et J.-L. Hainaut (2006). Data conceptualisation for web-based data-centred application design. In *Proceedings of the Advanced Information Systems Engineering, 18th International Conference, CAiSE 2006, Luxembourg*, Lecture Notes in Computer Science, pp. 205–219.

Zaki, M. J. (2002). Efficiently mining frequent trees in a forest. In *KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, pp. 71–80. ACM Press.

Summary

This paper presents our work regarding the analyzis of redundancies amongst a collection of database schemas. We will start by introducing the concepts of redundancy, similarity and pattern. We will then also present a few approaches to extract redundancies from database schemas. This concern is a key part of ReQuest, a tool-supported methodology aiming at conceiving information systems by wisely exploiting the information conveyed by human-computer interfaces.

Creating and Maintaining Mappings for XML Data Integration

Xiaohui Xue, Zoubida Kedad

Laboratoire PRiSM, Université de Versailles, 45 avenue des Etats-Unis, 78035 Versailles
{xiaohui.xue, zoubida.kedad}@prism.uvsq.fr

Abstract. We address the problems of mapping creation and maintenance for XML data sources. Mappings describe the way instances of a target schema are derived from the data sources. We propose a mapping creation process that generates a set of alternative mappings for a target schema from the sources. We also propose a mapping maintenance process that, if changes occur in the schemas, adapts the mapping to keep it consistent with the new schemas. A prototype implementation is presented to support the process of mapping creation and it has been used to evaluate the performance of our algorithms.

1 Introduction

Nowadays, numerous modern information systems such as data translation systems, mediation systems and data warehouses need to use existing data from heterogeneous and distributed data sources. In this context, applications' needs are represented by a *target schema* and *mappings* are defined between the target schema and the source schemas to express the way instances of the target schema are derived from the sources.

Manually creating the mappings is a difficult task. The designer must have a thorough understanding of not only the data sources, but also the semantic links between the sources and the target schema. The amount of metadata to manage may be very important when there are a large number of data sources. It is therefore necessary to provide a support for mapping creation. Both the applications and the sources may frequently evolve their schemas. These changes may make the mappings invalid and it is necessary to maintain them for the new schemas.

Many research works have been devoted to mapping creation (e.g. Kedad and Bouzeghoub 1999, Miller et al. 2000, Popa et al. 2002) and mapping maintenance (e.g. Bouzeghoub et al. 2003, Lee et al. 2002, Velegrakis et al. 2003). Our work is among these contributions. We propose a mapping creation process that produces a set of alternative mappings for a target schema from multiple data sources. We also propose a mapping maintenance process. If some changes occur in the schemas, it adapts the mappings to keep them consistent with the new schemas. A prototype implementation is developed to support mapping creation and it has been used to evaluate the performances of our algorithms.

The rest of the paper is organized as follows: Section 2 presents the existing works on mapping creation and mapping maintenance; Section 3 and 4 describes our approaches for mapping creation and mapping maintenance respectively; the prototype for mapping creation and the evaluation of the algorithms is presented in Section 5; Section 6 concludes de paper.

2 Related Works

Many research approaches (Chen and al. 2003, Fletcher and Wyss 2006, Kedad and Bouzeghoub 1999, Loscio and Salgado 2003, Miller and al. 2000, Popa and al. 2002, Soukane 2005, Su and al. 2001, Zamboulis 2004) and industrial solutions (Adeptia Integration Service, Altova MapForce and Stylus XML-to-XML Mapper) have been proposed for automatically or semi-automatically generate mappings.

Among these existing works, some research approaches (Fletcher and Wyss 2006, Miller et al. 2000, Popa et al. 2002, Su et al. 2001) and all the industrial solutions are proposed for the context of data transformation. These works consider one target schema and one source schema and their objective is to transform data from the source representation to the target representation.

The other approaches (Chen and al. 2003, Kedad and Bouzeghoub 1999, Loscio and Salgado 2003, Soukane 2005, Zamboulis 2004) consider the context of data integration in which there are several data sources. The two approaches (Chen et al. 2003, Zamboulis 2004) consider that the target schema is generated from the integration of the source schemas. The two approaches generate mappings according to the LAV approach. Three approaches (Kedad and Bouzeghoub 1999, Loscio and Salgado 2003, Soukane 2005) generate mapping for a target schema from multiple source schemas and the generated mappings contain joins between different sources. The approaches (Kedad and Bouzeghoub 1999, Soukane 2005) consider relational schemas. The approach (Loscio and Salgado 2003) considers schemas in XML and they consider that there are some similarities between the structures of the target and the source schemas. Our objective is to propose an approach to support mapping creation in a context of data integration; we consider that the schemas are expressed using XSD (XML Schema) and we do not make any assumption about the similarities between them.

Some approaches (Bouzeghoub et al. 2003, Lee et al. 2002, Loscio and Salgado 2004, McBrien and Poulovassilis 2002, Velegrakis et al. 2003, Yu and Popa 2005) have been proposed to maintain mappings automatically when the schemas evolve. The approaches proposed in (Bouzeghoub et al. 2003, Loscio and Salgado 2004, Velegrakis et al. 2003) are proposed in the context of an existing mapping generation methodology and the proposed mapping maintenance process can be considered as an incremental execution of this methodology. These approaches assume that all the intermediate results of the mapping generation are provided. They define, for every change they consider, the actions of re-computing the intermediate results and then the final results of the mapping generation process. The approach presented in (Velegrakis et al. 2003) compares the performance of mapping maintenance and the performance of mapping creation. The result shows that the mapping maintenance gives better performance. These experimentations have been made for scenarios of about 30 elements in the source schema.

The approaches presented in (McBrien et Poulovassilis 2002, Velegrakis et al. 2003) consider that the evolution of a schema itself is expressed by a mapping and the process of mapping maintenance consists in combining the original mapping with the mapping expressing the schema evolution to obtain new mappings. These approaches consider mappings generated for one source schema and they assume an existing mapping creation process to generate mapping expressing the schema evolution.

The approach proposed in (Lee et al. 2002) maintains mappings between relational schemas for some considered changes. The maintenance is performed using some existing

meta-data about the source such as all the possible joins between source relations and the inclusion relations between different source instances, etc. Our objective is to propose a mapping maintenance process for mappings expressed in XQuery without making an assumption about the way the mappings are generated.

All the existing works on mapping creation and maintenance use semantic correspondences. Semantic correspondences involve elements of different schemas and state that these elements represent the same concept. For example, a semantic correspondence can be defined between "professor" and "teacher" to state that they represent the same meaning. Transformation functions can be used in semantic correspondences. Providing support to schema matching is a difficult problem and is addressed by many research works (e.g. Rahm and Bernstein 2001, Reynaud et al. 2001). The generation of semantic correspondences is beyond the scope of this paper.

3 Mapping Creation

Our mapping creation process (Kedad and Xue 2005) considers that both target and source schemas are described using XML Schema and there is a set of semantic correspondences between the target schema and the source schemas. It produces a set of mappings to populate the target schema from the instances of the source schemas; every one represents an alternative way to define the target from the sources.

To handle the complexity of mapping creation, the target schema is decomposed into subtrees. Mappings are first created for each of these subtrees; they are called *partial mappings*. Mappings for the whole schema are then obtained from the combination of the partial mappings. Mappings are expressed in an abstract language and they can be translated into XQuery.

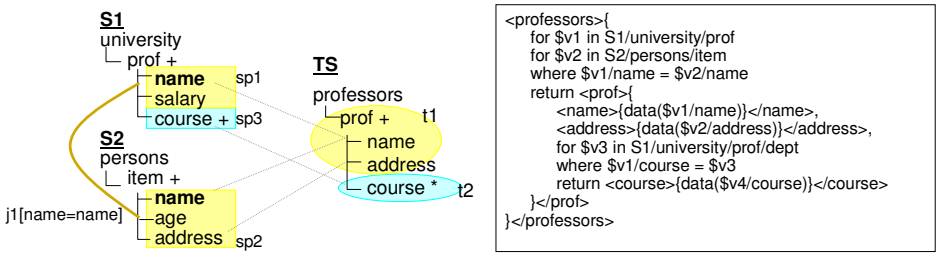


FIG. 1 – Example of a mapping for one target schema from two source schemas

Our algorithm for the decomposition of the target schema produces several subtrees. Every subtree can have any depth but it has to satisfy the following conditions:

- its root is either a multivalued element or the root of the target schema;
- all the other nodes of the subtree are descendents of r and are monovalued;
- there is at least one text node in t (t may contain a single node).

Consider the example of *FIG. 1*. There is one target schema, two source schemas and some semantic correspondences that express the equivalence between elements of different schemas (represented by dotted lines). The target schema is first decomposed into two target subtrees *t1* and *t2*. In each subtree, all the elements except the root node are monovalued.

Once the decomposition of the target schema is completed, there are two tasks left for mapping generation. First, we search for mappings for every target subtree, called partial mappings. Since all the elements are monovalued in a subtree, this consists in finding some equivalent nodes in the sources that satisfy the cardinalities constraints regardless their hierarchical organization. The second step consists in generating a mapping for the whole target schema from the partial mappings, which consists in finding the hierarchical relations between the different target subtrees.

Partial mappings are defined for each target subtree independently from the others. Our algorithm of defining partial mappings is composed of three steps: (i) identifying the portions of the sources (called *source parts*) that are relevant for the considered target subtree; (ii) searching for the joins to combine these source parts; (iii) and determining the partial mappings from the source parts and the joins between them. Each target subtree may have several partial mappings with different semantics.

Given a target subtree *t*, a source part *sp* for *t* in a source schema *S* is defined as a set of text elements such that:

- every element in *sp* is involved in a correspondence with an element in *t*;
- the elements of *sp* can be contained in the same subtree in *S* such that all the elements except the root element, of the subtree are monovalued;
- there is no other set of elements *c* such that $sp \subseteq c$ and *c* satisfies the two above conditions.

Consider the subtree *t1* in the example of FIG. *1*. There are two corresponding source parts *sp1* and *sp2* in the two source schemas; both satisfy the above condition.

Joins are inferred using the constraints defined in the source schemas. There is a join between two source parts of the same source schema if there is a reference from one to the other. A join is possible between two source parts of different schemas if there are two elements of each that are equivalent and one of them is defined as a key. The process searches for all the joins between all the source parts following these rules.

Between the source parts *sp1* and *sp2* for the subtree *t1* in FIG. *1*, there is a candidate join *j1* with the predicate *[name=name]*. This is because *name* is defined as a key in both of the two schemas and the two elements *name* of the two sources are equivalent.

The partial mappings of a target subtree are defined from the corresponding source parts and the joins between them. Consider a graph *G* for each target subtree *t* (called *join graph*) in which the nodes presents the source parts for *t* and the edges represents the inferred joins. Every partial mapping for *t* is defined as a connected acyclic sub-graph of *G* such that for every mandatory text element in *t*, there is an equivalent element in the sub-graph. The process searches for all the partial mappings for every target subtree.

For the subtree *t1* in FIG. *1*, two alternative partial mappings *m1* and *m2* are defined: *m1* defines *t1* from the elements *name* and *address* of *S2*; *m2* joins *S1* and *S2* with the join predicate *[name = name]* and defines *t1* using *name* and *address* from the result of the join. Following the same three steps, we also obtain one partial mapping *m3* that populates the subtree *t2* from *S1*.

Partial mappings are combined to generate mappings for the target schema. Each combination must satisfy the hierarchical relationship between *t1* and *t2*. There are two combinations in the running example: {*m1*, *m3*} and {*m2*, *m3*}. Only the second one satisfies the relationship between *t1* and *t2*; it defines both *t1* and *t2* using *S1*, which has a similar structure as the target schema.

The XQuery translation of the mapping is shown in *FIG. 1*. The translation is fully automatic. We defined a pattern for For-Where-Return clauses. Every partial mapping is translated into a FWR (For-Where-Return) clause according to this pattern. For each target subtree *t* and its parent subtree *t'*, the FWR expression of *t* is nested in the one of *t'* and a grouping condition is added in the FWR expression of *t* to group its instances with respect to the instances *t'*.

4 Mapping Maintenance

Our process for mapping maintenance (Xue 2006) considers mappings expressed in XQuery. We consider a set of changes that can occur in the schemas such as additions and removals of elements or constraints. For each change, the mapping is maintained in three steps. We first check if the mapping is invalidated by the change. If it is the case, we check if there is a solution to repair the mapping regarding the new schemas. If the mapping can be repaired, new mappings are generated. A maintenance algorithm is proposed for every considered change type.

We consider that the mappings are described according to the pattern shown in *FIG. 2*. The target schema is decomposed into subtrees following the rule defined in Section 3. For every subtree, the mapping contains a For-Where-Return clause. This clause is specified according to the pattern of *FIG. 2*. This pattern is the same one used for XQuery translation. Every `For`, `Where` and `Return` clause of the mapping is called a *component* of the mapping.

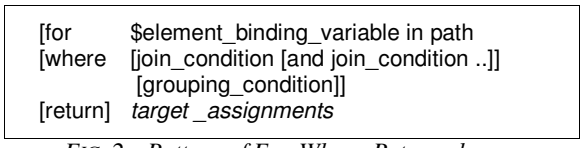


FIG. 2 – Pattern of For-Where-Return clauses

For every change in the schemas, some components of the pattern may be invalidated by the change. E.g. the removal of a source element can invalidate all the component of the pattern while the removing of a target element can only invalidate a `Return` clause. For every change of an element of a constraint, our process checks the appropriate components to see if the components use the element or constraint. The mapping is invalidated by the change if one of the components is invalidated by the change.

Consider again the example of *FIG. 1* and the three following changes: (i) removing the element *salary* from *S1*, (ii) removing the element *course* from *TS*, (iii) and removing the element *address* from *S2*. These changes lead to respectively the new schemas *S1'*, *TS'* and *S2'* shown in *FIG. 3*. The mapping is not invalidated by the first change because the change removes a source element that is not used in the mapping. The mapping is affected by the two others. After the removal of *course* from *TS*, the returned instances of the mapping do not conform to the new target schema anymore. After the removal of *address* from S2, the mapping has to find another element to replace *address* in the mapping.

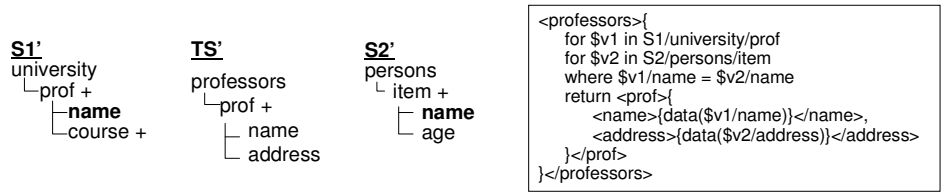


FIG. 3 – Three schema changes and a mapping produced by the maintenance process

For every component of the pattern that is invalidated by a change, an algorithm is used to search for the solutions to repair it regarding the new schemas. A mapping can be repaired if all its invalidated components can be repaired. For every found solution, a mapping is generated for the new schemas.

Consider our example of the two changes that invalidate the mapping. For the removal of the target element, repairing the mapping consists in removing the corresponding assignment. The mapping can be repaired and the new mapping is given in *FIG. 3*. After the removal of *address* from *S2*, the mapping can be repaired only if a substitution is found. In the example, there is no more information in the sources that concerns the addresses (i.e. no more semantic correspondence involving *address* of *TS*). The mapping can therefore not be repaired for the change.

5 Experimentation

A system prototype (Kostadinov et al. 2004) is developed in Java to implement the process of mapping creation and it has been used to evaluate the performance of the algorithms used in the approach.

We have run five scenarios to evaluate its performance. *TAB. 1* summarizes the main characteristics of these scenarios, such as the number of nodes in the target schema, the number of data sources and the number of nodes for each one, the number of correspondences between the sources and the target schema, and the number of the key definitions in the sources.

Scenarios	Target schema			Corresp-ondences	Source schemas				
	Depth	Nodes	Text nodes		Schemas	Nodes	Text nodes	Keys	Refs
Mediagrid	6	18	12	22	3	1674	825	412	413
Library1	5	18	14	26	6	56	30	9	1
Library2	5	18	14	30	6	62	35	10	1
ABC1	7	47	36	1300	50	1650	1434	0	0
ABC2	7	47	36	1300	50	1650	1434	58	0

TAB. 1 – Characterizing the scenarios

The first scenario is from the Mediagrid project[1] which proposes a mediation framework for a transparent access to biological data sources; it considers three biological sources SGD, GOLD, SMD and a target schema built by domain experts. The Library1 scenario contains six source schemas. The Library2 scenario is similar to Library1 but the overlap between the sources is more important (more correspondences are defined for the same number of text nodes in the target schema). The ABC1 and ABC2 scenarios contain 50 source schemas. They are similar, except that the ABC2 scenario contains 58 key definitions while ABC1 contain no key definitions.

We have run these different scenarios on a PC-compatible machine, with a 2.8G Hz P4 CPU and 516MB RAM, running Windows XP and JRE1.4.1. Each experiment is repeated five times and the average of the five is used as the measurement.

Scenarios	Execution time (s)			
	Load	Target Schema Decomposition	Partial Mapping Determination	Target Mapping Generation
Mediagrid	1.44	0.001	0.02	0.002
Library1	0.44	0.001	0.067	0.095
Library2	0.046	0.001	0.105	0.25
ABC1	0.98	0.001	0.06	1.997
ABC2	1.03	0.001	316	27

TAB. 2 – Characterizing the scenarios

The time needed for the main steps of our approach using the different scenarios are shown in *TAB. 2*. The loading time indicates the time to read the schemas and the correspondences into our internal representation. As expected, it is correlated to the size of the schemas and the number of their correspondences.

The target schema decomposition time indicates the time to decompose the target schema into target subtrees. We can see that the time needed to perform the task is negligible.

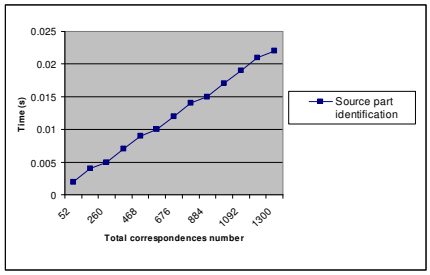
The partial mapping determination (pmd) time is proportional to the number of correspondences between target nodes and source nodes and the key and key references in the sources. The pmd time for Library1 which has 26 correspondences is smaller than the one of Library2 which has 30 correspondences; the two scenarios have the same number of sources and the same target schema. The pmd time for the ABC2 scenario which has 58 keys is largely greater than the one of the ABC1 scenario. This is because the number of keys of the ABC2 scenario makes the join graph very complex.

The target mapping generation (tmg) time indicates the time to find all the candidate mappings and to generate the target mappings. The tmg time is greater in ABC2 than in the other scenarios because in ABC2, most of the target subtrees have a lot of partial mappings (about 150), which leads to much more combinations to consider.

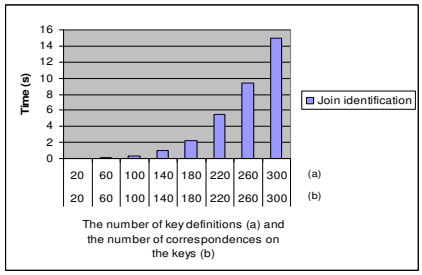
[1] Supported by the French Ministry of Research through the ACI Grid program, www-lsr.imag.fr/mediagrid/

Some evaluations for the partial mapping determination and the target mapping generation are shown in *FIG. 4*. The pmd time is decomposed into source part identification time, join identification time and partial mapping determination time. *FIG. 4* (a) shows the source part identification time with respect to the number of the semantic correspondences between the target schema and the source schemas. The measures are done using the ABC2 scenario and considering 52 to 1300 correspondences. This task is proportional to the number of correspondences and its time is almost negligible (about only 0.022 second for 1300 correspondences).

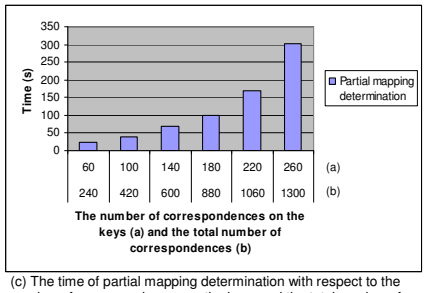
FIG. 4 (b) shows the time of join identification with respect to both the number of key definitions and the number of correspondences on the keys. We have considered the ABC2 scenario and we have successively increased both the number of key definitions and the number of correspondences involving keys. The time needed to perform this task is influenced by the two parameters. With 300 key definitions and 300 correspondences on the keys (which represents a complex case), the time for the join identification is about 15 seconds.



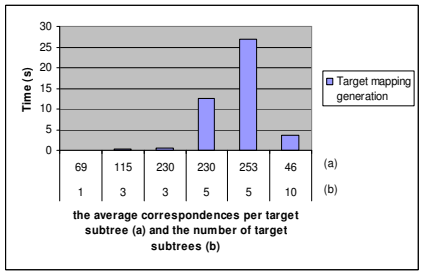
(a) The time of source part identification with respect to the number of correspondences



(b) The time of join identification with respect to the number of key definitions in the sources and the number of correspondences on the keys



(c) The time of partial mapping determination with respect to the number of correspondences on the keys and the total number of correspondences



(d) The time of target mapping generation with respect to the average number of correspondences per target subtree and the number of target subtrees in the target schema

FIG. 4 – Evaluating the time for the different steps of mapping generation

The time required for the determination of partial mappings for a given target subtree depends on the size of the corresponding join graph. *FIG. 4* (c) shows the time for partial mapping determination with respect to both the total number of correspondences and the number of correspondences for the keys. We have successively increased the values of these two parameters from 60 correspondences for the keys and 240 total correspondences to 260

correspondences for the keys and 1300 total correspondences. The other parameters of the scenarios used in this experiment are the same as the ABC2 scenario. We can see in the graph that a scenario having 880 correspondences among which 180 correspondences involving source keys takes about 100 seconds for the partial mapping determination.

The target mapping generation depends on the number of partial mappings and the structure of the target schema, that is, the number of parent-child relations between the target subtrees. *FIG. 4* (d) shows the time required for this task with respect to the number of the target subtrees and the average correspondences per subtree. We have increased both parameters using the same sources as for the ABC2 scenario. For example, in the case of 5 target subtrees and 230 correspondences per subtree, it takes about 12 seconds for generating the target mappings; note that this case is a complex one, since the scenario contains 50 sources, 1300 correspondences and 58 key definitions. The complexity of this process is exponential with respect to the number of partial mappings. It is possible to reduce this complexity using some quality criteria (for example, selecting the partial mappings that use the sources having a high confidence factor) or some heuristics (for example, selecting the partial mappings using a high number of sources).

6 Conclusion

This paper addresses the problem of mapping creation and maintenance for schemas in XML. Our process of mapping creation generates mappings to define one target schema from multiple source schemas. When the schemas evolve, our process of mapping maintenance allows automatically adapting the existing mapping to obtain mappings for the new schemas. We implemented the process of mapping creation and evaluated the performance of our algorithms.

To improve the performance of mapping creation, we are considering the use of some quality evaluation tools to eliminate some intermediate results that do not meet the requirements of the user. To achieve this goal, our system is already integrated in a platform that also provides tools for evaluating the quality of mappings considering user preferences (Kostadinov et al. 2005).

For the cases in which the cost of mapping generation is very low, mapping adaptation can be done by restarting the generation process. For all other cases, our objective is to propose adaptation algorithms that avoid restarting the whole generation process. We are currently implementing our algorithms for mapping maintenance to evaluate their performances and to characterize the size of the problem for which the adaptation is a better alternative.

References

Adeptia Integration Server: http://www.adeptia.com/products/ais.html

Altova MapForce: http://www.altova.com/products_mapforce.html

Bouzeghoub M., Lóscio B. F., Kedad Z., Salgado A. C. (2003), Managing the evolution of mappings. Proc. of CoopIS 2003, Catania, Italy, 22-37

Chen Y. B., Ling T. W., Lee M.-L. (2003), Automatic Generation of XQuery View Definitions from ORA-SS Views. Proc. Of ER 2003, Chicago, Illinois, 158-171

Fletcher G. H. L., Wyss C. M. (2006), Data Mapping as Search. Proc. Of EDBT 2006, Munich, Germany, 95-111

Kedad Z., Bouzeghoub M. (1999), Discovering View Expressions from a Multi-Source Information System. Proc. of CoopIS 1999, Edinburgh, Scotland, 57-68

Kedad Z., Xue X. (2005), Mapping Discovery for XML Data Integration. Proc. of CoopIS 2005, pp 166-182

Kostadinov, D., Peralta, V., Soukane, A., Xue, X. (2004), Demonstration: Système adaptif à l'aide de la génération de requêtes de médiation. Proc. of BDA ,Montpellier, France, 351-355

Kostadinov, D., Peralta, V., Soukane, A., Xue, X. (2005), Intégration de données hétérogènes basée sur la qualité. Proc. of INFORSID ,Grenoble, France, 471-482

Lee A. J., Nica A., Rundensteiner E. A. (2002), The EVE Approach: View Synchronization in Dynamic Distributed Environments. IEEE Trans. Knowl. Data Eng. 14(5): 931-954

Lóscio B. F., Salgado A. C. (2003), Generating Mediation Queries for XML-based Data Integration Systems. Proc. Of SBBD 2003, 99-113

Lóscio B. F., Salgado A. C. (2004), Evolution of XML-Based Mediation Queries in a Data Integration System. Proc. of ER Workshops, Shanghai, China, 402-414

McBrien P., Poulovassilis A. (2002), Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach. CAiSE 2002: 484-499

Miller R. J., Haas L. M., Hernández M. A. (2000), Schema Mapping as Query Discovery. Proc. of VLDB 2000, Cairo, Egypt, 77-88

Popa L., Velegrakis Y., Miller R. J., Hernandez M.A., Fagin R. (2002), Translating web data. Proc. of VLDB 2002, Hong Kong, China, 598-609

Rahm E., Bernstein P. A. (2001), A survey of approaches to automatic schema matching. Proc. of VLDB, Roma, Italy, 334-350

Reynaud C., Sirot J.-P., Vodislav D. (2001), Semantic Integration of XML Heterogeneous Data Sources. ¨Proc. of IDEAS 2001, Grenoble, France, 199-208

Soukane A. (2005), Génération automatique des requêtes de médiation avec prise en compte des besoins des utilisateurs dans un environnement hétérogène. PhD Thesis, Université de Versailles-Saint-Quentin-en-Yvelines, December 2005

Stylus XML-to-XML Mapper: http://www.stylusstudio.com/xml_to_xml_mapper.html

Su H., Kuno H. A., Rundensteiner E. A. (2001), Automating the transformation of XML documents. Proc. Of WIDM, 68-75

Velegrakis Y., Miller R. J., Popa L. (2004), Preserving mapping consistency under schema changes. VLDB J. 13(3): 274-293

Xue X. (2006), Automatic Mapping Generation and Adaptation for XML Data Sources, PhD Thesis, Université de Versailles, December 2006

Zamboulis L. (2004) XML Data Integration by Graph Restructuring. Proc. Of BNCOD, 57-71

Passage à l'échelle de la réconciliation de concepts et de la réconciliation de références : quelques points de comparaisons

Nathalie Pernelle*, Fatiha Saïs*

* LRI, Université Paris-Sud 11, F-91405 Orsay Cedex,
INRIA Futurs, 2-4 rue Jacques Monod, F-91893 Orsay Cedex, France
{prenom.nom}@lri.fr

Résumé. L'intégration de données provenant de différentes sources se confronte à deux problèmes majeurs : l'hétérogénéité des schémas et les variations dans les descriptions des instances. Nous avons comparé ces deux problèmes en se focalisant sur l'existence de méthodes permettant de faire face au passage à l'échelle et donc au traitement de données nombreuses et éventuellement distribuées.

1 Introduction

L'intégration de données provenant de différentes sources se confronte à deux problèmes majeurs : l'hétérogénéité des schémas ou des ontologies et les variations dans les descriptions des instances. De nombreux travaux proposent des méthodes permettant de réconcilier des ontologies en découvrant un ensemble de mappings possibles. Il s'agit alors de mettre en relation le vocabulaire de deux ontologies de façon à ce que leur structure mathématique, spécifiée par les axiomes de l'ontologie, soit respectée (Kalfoglou et Schorlemmer (2003)). Selon les approches, les relations possibles peuvent varier : il peut s'agir d'équivalences, de subsumptions, mais aussi de recouvrements ou de disjonctions. D'autres travaux se sont focalisés sur le problème de la réconciliation de références qui découle de l'hétérogénéité dans les descriptions des instances. Dans ce cas, il s'agit de décider si deux descriptions réfèrent ou non à la même entité du monde réel (savoir, par exemple, si deux descriptions décrivent le même hôtel ou la même personne).

Pour décider si deux concepts de deux schémas peuvent être réconciliés, les méthodes peuvent prendre en compte la similarité des concepts qui lui sont liés. De la même façon, pour décider si deux entités peuvent être réconciliés, certaines approches ont une approche globale exploitant les dépendances qui peuvent exister entre réconciliations de références. Par exemple, la réconciliation entre deux références à des articles influe sur la réconciliation entre les deux conférences où sont publiés ces articles.

Spécifier manuellement des mappings entre deux schémas est très coûteux et l'on est confronté au nombre croissant de sources de données en provenance du Web à intégrer. La recherche manuelle de réconciliations de références est encore moins envisageable. Par exemple, les sites comparateurs de prix (e.g. www.kelkoo.com) doivent traiter des millions de références de produits par jour. Ces méthodes doivent donc être aussi automatiques que possible. La mise

à l'échelle des méthodes de réconciliation de concepts et de réconciliations de références se confronte aussi à des contraintes sur le temps d'execution. Certains portails d'information se donnent pour règle de répondre à 30 requêtes en moins de 3 secondes. De plus, les données peuvent être distribuées dans un réseau pair-à-pair.

Le passage à l'échelle de telles méthodes suppose soit la possibilité de disposer d'un ensemble de connaissances permettant de filtrer les données à réconcilier avant de réaliser des traitements plus complexes pour diminuer l'espace des réconciliations possibles, ou de découper le problème en problèmes qui peuvent être résolus de manière indépendante. En ce qui concerne le problème de performance en temps, il est possible de concevoir un système parallèle pour l'exécution des réconciliations ou d'exploiter des méthodes itératives qui permettent de fournir un résultat approximatif après n itérations.

Nous proposons de présenter une comparaison possible des deux problèmes de réconciliation sur ces différents points.

2 Diminuer la taille de l'espace de réconciliation

Pour la réconciliation de références, l'espace de réconciliation représente l'ensemble des couples de références qui sont candidats à la réconciliation. Pour la réconciliation de schéma, l'espace considéré représente l'ensemble des couples de concepts et de relations des schémas (ontologies) à réconcilier. Dans le cas des approches exploitant les instances pour la mise en correspondance de schémas (Euzenat et Valtchev (2004), ou FCA-Merge de Stumme et Maedche (2001)), l'espace de réconciliation des schemas contient également l'ensemble des instances disponibles (appelées aussi objets ou références) des concepts des schémas à réconcilier.

2.1 Filtrage

Pour diminuer la taille de l'espace des réconciliations, les méthodes de réconciliations de références peuvent utiliser en prétraitement des techniques de filtrage. Ces techniques exploitent des connaissances du domaine pour limiter le nombre de couples candidats. Il peut s'agir de :

Méthodes dites de *blocking* : on ne considère que les paires de références qui possèdent une (ou plusieurs) caractéristiques communes, caractéristiques telle que le numéro de ISBN pour les livres, ou encore le nom de famille pour les personnes. Ces techniques ont été introduites par (Newcombe et Kennedy (1962)) et sont utilisées dans des travaux récents tels que Baxter R. (2003). Si il existe une telle caractéristique qui comporte m valeurs, l'espace des réconciliations est divisé par m.

Le corpus CORA (corpus de citations de publications scientifiques qui a servi de benchmark) comporte 6000 références d'articles, de conférences, de journaux et d'auteurs. L'espace de éconciliation concernant les articles et les conférences contient 2587 références. L'espace des réconciliations conteint donc 6692569 couples de références à comparer. Utiliser l'année comme caractéristique de filtrage, sachant que les publications existent sur 6 années différentes, permet de réduire cet espace de 21,8 %.

L'exploitation de connaissance du domaine telles que les disjonction entre classes (Sais et al. (2007)) : deux références qui appartiennent à des classes disjointes ne sont pas réconciliables. France telecom nous a fournit un corpus décrivant 562368 hotels. Les disjonctions entre classes d'hôtels de pays différents permttent de réduire l'espace des réconciliations de 67,8%.

L'exploitation de propriétés sur les sources telles que l'Unique Name Assumption : deux références issues d'une source de données qui possède la propriété d'UNA sont forcemment distinctes.

Ces techniques de filtrage permettent de filtrer l'espace en déduisant facilement des non réconciliations entre couples. Ce filtrage peut également se propager en cours de traitement. Ainsi, si une référence r1 d'une source S1 à été réconciliée à une référence r2 d'une source S2 qui possède l'UNA, toutes les autres possibilités de réconciliation de r1 avec une référence S2 peuvent être éliminées.

Ces méthodes nous semblent difficilement utilisables lorsque l'on s'intéresse à la réconciliation de schema. Même si deux noms de concepts d'une ontologie correspondent forcemment à des concepts distincts, cette propriété ne peut être propagée dans la mesure où un concept d'une ontologie A peut généraliser deux concepts d'une ontologie B.

2.2 Partitionnement des données pour partitionner l'espace de réconciliation

Le partitionnement consiste à diviser *l'espace de réconciliation* en plusieurs sous-ensembles (parties) de taille plus petite.

Partitions pour la réconciliation : l'espace de réconciliation est partitionné en plusieurs sous-ensembles de paires de références (ou de concepts) de taille plus petites de manière à ce que la couverture (le rappel) de la méthode de réconciliation ne soit pas diminuée.

Pour partitionner l'espace des réconciliations, on peut d'abord partitionner les références (ou les concepts). Pour assurer la non redondance et la non perte d'information, le partionnement doit satisfaire trois critères (Ozsu et Valduriez (1999)) :

Complétude : pour toute référence (resp. concept) il existe une partition P_i contenant cette référence (resp. concept).

Reconstruction : pour toute source S (resp. schéma S) partitionnée en un ensemble de parties P_i, il existe une opération de reconstruction telle que $S = \cup P_i$ pour tout P_i appartenant à l'ensemble des partitions. Cette opération de reconstruction est à définir en fonction du partitionnement effectué. Par exemple, dans le cas où les partitions sont des composantes connexes d'un graphe, l'opération de reconstruction est la fusion de graphes.

Disjonction : une référence (resp. concept) n'est présente que dans une seule partition.

Partitions pour la réconciliation de références : Le graphe G d'un ensemble de références I d'une source de données peut être représenté sous la forme d'un multi-graphe orienté étiqueté dont les sommets V_G sont des références et les arcs E_G sont des relations entre références.

$$G = \langle V_G, E_G, R_G \rangle$$

où

$$V_G = I, \qquad E_G \subseteq V_G \times R_G \times V_G$$

et

$$\langle i1, r, i2 \rangle \in E_G \; ssi \; \exists r, r(i1, i2)$$

Considérons que l'ensemble de références est représenté dans un multi-graphe G. Ainsi, l'ensemble des partitions dans l'ensemble de références correspond exactement à l'ensemble de composantes fortement connexes CFC que l'on peut former dans le graphe G. Une composante connexe pour une référence i représente le graphe contenant l'ensemble des références atteignables à partir de i par les relations instanciées.

Une fois que ces CFC ont été définies, elles sont ensuite enrichies par l'ensemble des valeurs atomiques associées aux différentes références par les attributs. Ces derniers sont très importants pour la mesure de la similarité entre les descriptions des références afin de pouvoir décider de leur réconciliation ou de leur non réconciliation. Ainsi, l'espace de réconciliation de référence est réduit à un ensemble d'espaces plus petits correspondant aux paires de composantes fortement connexes du graphe de références G, enrichies par les valeurs des attributs associés à chaque référence des CFCs. Lors de la réconciliation d'une paire de CFCs, l'ensemble des paires de références est le produits cartésien des deux ensembles de références contenues dans les deux CFCs . Bien sûr, certaines paires pourront ne pas être considérées par la suite compte tenu des connaissances du domaine et des informations renseignées.

Nous notons que la taille des composantes connexes est dépendant du niveau de redondance dans les sources (présence de l'UNA). Plus précisément, plus la source est redondante plus les composantes connexes de cette source sont petites. Ainsi, si une source qui décrit un ensemble de références bibliographiques possède l'UNA, chaque publication est reliée à un ensemble d'auteurs (3 en moyenne) auquels sont également associées d'autres publications qui font donc parties de la même composante connexe. En revanche, une source telle que CORA ne possède par l'UNA et donc chaque composante connexe se limite à la description d'une publication. CORA qui comporte 6000 références peut donc être découpées en 1295 CFC et donc en 1677025 espaces de 25 références en moyenne.

Partitions pour la réconciliation de concepts : Pour la réconciliation de concepts, les concepts de chaque schéma sont généralement organisés dans un graphe connexe où les sommets sont les concepts et où les arcs sont des relations. Dans le cas où il n'existe pas de mappings entre les concepts, comme le graphe des concepts est connexe, on ne peut pas définir de composantes connexes plus petites que le graphe lui-même. En revanche, le partitionnement de l'ensemble des concepts est possible quand des mappings préalables existent dans le cas de taxonomies où la seule relation sémantique considérée est la subsumption. Soient Sc1 et Sc2 deux taxonomies à réconcilier et m un mapping d'équivalence entre les deux concepts A et B calculé ou donné par un expert. Soient G_A et G_B les deux sous-arbres de Sc1 et Sc2 enracinés respectivement par A et par B. L'exploitation du mapping m nous permet de partitioner les deux schemas en quatre sous-arbres :

$$(Sc1 \setminus G_A), (Sc2 \setminus G_B), G_A, G_B$$

Avec ($\setminus$) exprime la difference entre deux graphes.

Ce mode de partitionnement peut être réitéré dans le cas où nous avons à disposition plusieurs mappings. Ainsi, l'espace de réconciliation est décomposé, et donc pour la réconciliation d'une paire de sous-arbre, l'espace de réconciliation est de taille plus petite.

L'ontologie du tourisme fournie par France Telecom comporte 210 concepts. L'espace des réconciliations qui serait créé avec une ontolgie de taille similaire contiendrait donc 44100 couples.

A priori la réconciliation de référence conduit plutôt à de très nombreuses partitions de petites taille tandis que la réconciliation de schema conduit plutôt à un espace de grande taille.

3 Une meilleure gestion du temps

3.1 Approximer

Le traitement de données nombreuses peut conduire à l'utilisation de méthodes permettant d'approximer un résultat grâce à l'utilisation d'algorithmes itératifs que l'on peut décider de stopper après n itérations. De telles méthodes ont pu être définies pour les deux problèmes de réconciliation.

Dans le cas de la réconciliation de schemas, Euzenat et Valtchev (2004) ont défini une méthode itérative utilisant une mesure permettant de comparer les entités (concepts ou relations) de deux ontologies décrites en OWL-Lite. Cette mesure prend en compte toutes les informations concernant un couple de concepts ou de relation (son label, ses propriétés éventuellement multivaluées, ses instances, ses généralisants).

Dans le cas de la réconciliation de référence, nous proposons une mesure de similarité définie pour traiter des données représentées en RDF et décrites par un schema RDFS+ (RDF auquel s'ajoute des opérateurs OWL et des règles SWRL). Nous avons pu prouver qu'un algorithme de calcul itératif utilisant cette mesure converge. Dans ce calcul, la similarité d'une paire de références est fonction de la similarité des paires de références voisines. Les connaissances du domaine sont exploitées afin de faire varier l'impact de la similarité de ces paires voisines, en particulier pour tenir compte du fait qu'une relation est une dépendance fonctionnelle. Ce calcul itératif est appliqué paire de composante connexe par paire de composante connexe.

3.2 Parallélisation de l'exécution de la réconciliation

Afin d'améliorer les performances en terme de temps d'exécution de la réconciliation, une des solutions est de concevoir un algorithme parallèle. Il s'agit de distribuer la tâche de réconciliation sur un ensemble de processus tout en assurant la non perte d'informations. Pour cela, un ensemble de partitions disjointes peuvent être définies comme dans la section 2.

Dans le cas où les partitions ne possèdent pas les propriétés définies dans la section 2(Omar et al. (2006)), propose d'utiliser deux fonctions qui sont importantes pour la distribution des tâches : la fonction "scope" qui permet de distribuer les différentes partitions sur les différents processus et la fonction "responsible" qui permet d'assurer la non redondance, cela en décidant quel processus est responsable de quelle réconciliation. Une définition plus formelle de ces deux fonctions est donnée dans (Omar et al. (2006)).

De la même façon, la parallélisation de la réconciliation d'ontologies est envisageable.

4 Données distribuées, P2P

Les méthodes de réconciliation se confrontent également au fait que les schemas peuvent être distribués. Certaines approches travaillent dans ce cadre et estiment que l'utilisation de mappings entre ontologies locales qui se connaissent est plus réaliste que la création d'une ontologie commune importante. Un système tel que SomeWhere (Adjiman et al. (2006)) est une approche complètement pair à pair dans laquelle chaque pair stocke localement ses propres axiomes et un ensemble de mappings. Il exploite les mappings pour répondre de façon correcte et complète à une requête.

Une telle approche peut être complétée pour permettre de découvrir et d'exploiter des réconciliations de références afin d'intégrer les données provenant de différents pairs lors d'une requête. Nous envisageons une méthode dans laquelle chaque pair connaît éventuellement un ensemble de réconciliations entre les données (références) qu'il possède et celles d'autres pairs (comme pour les mappings). Le traitement des réponses aux requêtes va alors pouvoir exploiter les réconciliations de référence stockées dans les pairs impliqués dans une requête de SomeWhere. L'objectif est de trouver un chemin de reconciliation entre une référence d'un pair et une référence d'un autre pair pour les couples de références pertinentes correspondant à des variables liées de la requête.

5 Conclusion

Nous avons brièvement présenté quelques points de comparaison entre le problème de la réconciliation de références et celui de la réconciliation de schema quand leur résolution se confronte au passage à l'échelle. Ces points de comparaison concernent essentiellement des approches globales, approches qui prennent en compte un nombre d'informations qui peut être important dans un calcul complexe. Différentes stratégies permettant de limiter la taille des données ou le temps de calcul doivent donc être mises en place si ces méthodes veulent passer à l'échelle.

Références

Adjiman, P., P. Chatalic, F. Goasdoué, M.-C. Rousset, et L. Simon (2006). Distributed reasoning in a peer-to-peer setting : Application to the semantic web. *Journal of Artificial Intelligence Research 25*, 269–314.

Baxter R., Christen P., C. T. (2003). A comparison of fast blocking methods for record linkage. In *ACM workshop on Data cleaning Record Linkage and Object identification*.

Euzenat, J. et P. Valtchev (2004). Similarity-based ontology alignment in owl-lite. In *ECAI*, pp. 333–337.

Kalfoglou, Y. et M. Schorlemmer (2003). Ontology mapping : the state of the art. In *Knowl. Eng. Rev., 18(1) :1–31*.

Newcombe, H. B. et J. M. Kennedy (1962). Record linkage : Making maximum use of the discriminating power of identifying information. pp. 563–567.

Omar, B., G.-M. Hector, K. Hideki, L. Tait, M. David, et T. Sutthipong (2006). D-swoosh : A family of algorithms for generic, distributed entity resolution. In *Technical Report, Stanford InfoLab*.

Ozsu, M. T. et P. Valduriez (1999). *Principles of distributed database systems (2nd ed.)*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc.

Sais, F., N. Pernelle, et M. Rousset (2007). Approche logique pour la réconciliation de références. In *Extraction et Gestion des Connaissances conference (EGC 2007)*.

Stumme, G. et A. Maedche (2001). Fca-merge : bootom-up merging of ontologies. In *17th IJCAI*, pp. 225–230.

Summary

We have compared the ontology mapping and the reference reconciliation problems when they have to deal with numerous and eventually distributed data.

Aligner automatiquement des ontologies avec oMAP

Raphaël Troncy*

*CWI Amsterdam, P.O. Box 94079, 1090 GB, The Netherlands
Raphael.Troncy@cwi.nl,
http://www.cwi.nl/~troncy/

1 Introduction

Nous nous intéressons aux ontologies décrites dans un même langage de représentation des connaissances (OWL) et nous proposons un cadre général nommé oMAP pour automatiquement aligner des ontologies OWL sur le web. **oMAP** (Straccia et Troncy, 2005a, 2006) permet de trouver les meilleures correspondances (avec leurs poids) entre des entités définies dans des ontologies, en combinant la prédiction de plusieurs classifieurs. Ceux-ci peuvent être terminologiques ou basés sur des techniques statistiques d'apprentissage, ou encore utilisent la sémantique des axiomes OWL pour établir des relations d'équivalence entre des classes et des propriétés définies dans les ontologies. oMAP est disponible à :
http://www.cwi.nl/~troncy/oMAP/.

Notre approche s'inspire des travaux formels menés autour de l'échange d'information et emprunte à d'autres approches comme GLUE (Doan et al., 2003) l'idée de combiner plusieurs composants spécialisés pour obtenir le meilleur résultat.

Théoriquement, un *mapping* est un tuple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ où $\mathbf{S}$ et $\mathbf{T}$ sont respectivement les ontologies source et cible, et Σ est un ensemble de règles d'appariement de la forme : $\alpha_{j,i}\, T_j \leftarrow S_i$ où S_i et T_j représentent respectivement une entité (classe ou propriété) de l'ontologie source et cible. Cette règle signifie que l'entité S_i de l'ontologie source correspond à l'entité T_j dans l'ontologie cible, et que la mesure de confiance associée à cet appariement est $\alpha_{j,i}$. Notons qu'aucune contrainte supplémentaire n'est posée sur ces mises en correspondance : un concept de l'ontologie source peut correspondre à 0 ou plusieurs concepts dans l'ontologie cible et réciproquement.

Aligner deux ontologies dans *oMAP* est un processus en trois étapes :

1. Nous devinons un ensemble Σ possible, et nous estimons sa qualité en fonction de la mesure de confiance associée à chacune des règles d'appariement qu'il contient.
2. Pour chaque règle $T_j \leftarrow S_i$, nous estimons le poids $\alpha_{i,j}$, qui dépend aussi de l'ensembleΣ, i.e. $\alpha_{i,j} = w(S_i, T_j, \Sigma)$.
3. Comme nous ne pouvons pas considérer tous les ensembles Σ possibles (il y en a un nombre exponentiel) pour ensuite choisir le meilleur, nous construisons itérativement cet ensemble et nous utilisons diverses heuristiques pour réduire la complexité.

De la même manière que GLUE (Doan et al., 2003), nous estimons le poids $w(S_i, T_j, \Sigma)$ d'une règle $T_j \leftarrow S_i$ en combinant les prédictions de différents classifieurs $CL_1, \ldots, CL_n$.

Chaque classifieur calcule un poids $w(S_i, T_j, CL_k)$, approximation de la règle $T_j \leftarrow S_i$ pour le classifieur CL_k. Une liste de priorité permet finalement de combiner toutes ces prédictions qui ne sont donc pas simplement moyennées (Figure 1).

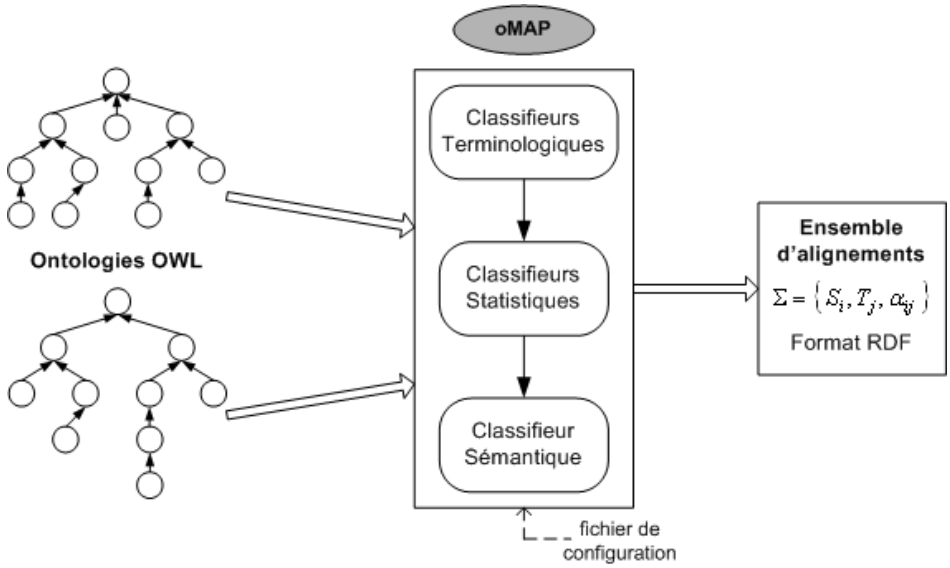


FIG. 1 – *Architecture générale de* oMAP. *Les différents classifieurs peuvent être combinés via un fichier de configuration.* oMAP *produit le meilleur ensemble d'alignements possibles, établissant une relation d'équivalence entre une entité i d'une ontologie S et une entité j d'une ontologie T avec la mesure de confiance α_{ij}.*

Nous décrivons dans la suite les différents classifieurs implémentés dans oMAP. Certains ne considèrent que la partie terminologique des ontologies alors que d'autres utilisent des méthodes statistiques d'apprentissage et peuvent utiliser les instances de l'ontologie. Finalement, un troisième type de classifieur fait appel à la structure et à la sémantique des définitions et des axiomes OWL.

2 Classifieurs terminologiques, statistiques et sémantique

Les classifieurs terminologiques utilisent le nom et les commentaires de chaque ressource ontologique. En OWL, chaque ressource est identifiée par un URI, mais peut également avoir des propriétés d'annotation attachées. Parmi celles-ci, la propriété `rdfs :label` est généralement utilisée pour fournir le nom usuel de la ressource. De plus, le multilinguisme des noms est supporté par l'attribut langue des littéraux RDF. Dans la suite, nous considérons que le nom d'une entité est donné par la valeur de la propriété `rdfs :label` ou par le fragment

de l'URI l'identifiant si cette propriété n'est pas spécifiée. Les classifieurs terminologiques de oMAP comparent le nom des entités, leur racine[1], ou calculent des mesures de similarité entre les chaînes de caractères (e.g. la distance de Levenshtein). Finalement, des dictionnaires ou ressources terminologiques tels que WordNet[2] peuvent être utilisés dans le calcul de la distance.

Les classifieurs statistiques considèrent les instances (ou individus) des ontologies. Les règles suivantes sont utilisées pour construire le texte correspondant à ces instances : nous considérons (i) le nom des individus nommés, (ii) la valeur pour les propriétés de type de données (`owl :DatatypeProperty`) et (iii) le type pour les individus anonymes et pour le co-domaine des propriétés de type objets (`owl :ObjectProperty`). Par exemple,, considérons les individus suivants :

Individual (x_1 type `(Workshop)`
  value `(label "Workshop DECOR")`
  value `(location` x_2`))`
Individual (x_2 type `(Address)`
  value `(city "Namur")` value `(country "Belgique"))`

Le texte u_1 pour l'individu x_1 sera donc (`"Workshop DECOR", "Address"`) et le texte u_2 pour l'individu anonyme x_2 sera (`"Address", "Namur", "Belgique"`). Les classifieurs populaires en apprentissage automatique tels que les algorithmes naïfs de Bayes ou kNN ont été implémentés dans oMAP.

Finalement, nous avons également proposés un classifieur considérant la sémantique des définitions OWL en étant guidés par leur syntaxe. Ce classifieur structurel est décrit dans (Straccia et Troncy, 2005a). Il est utilisé dans notre outil *a posteriori* dans la mesure où l'ensemble des prédictions des autres classifieurs lui sert d'entrée et qu'il cherchera à étendre Σ en ajoutant des règles $T_j \leftarrow S_i$ si aucun appariement concernant T_j est déjà présent dans Σ.

Tous ces classifieurs ont donc été implémentés et sont compatibles avec l'API d'alignement décrite dans (Euzenat, 2004), ce qui facilite leur composition. Le problème d'aligner des ontologies a déjà été rapporté dans de nombreux travaux (Shvaiko et Euzenat, 2005). Cependant, il est difficile de comparer théoriquement les différentes approches. Depuis trois ans, l'Ontology Alignment Evaluation Initiative (OAEI[3]) propose des compétitions et des jeux de tests pour mesurer les forces et les faiblesses des différents outils. Nous avons évalué oMAP avec les données des campagnes d'évaluation de 2004 et 2005 (Straccia et Troncy, 2005b). Nous présentons brièvement dans la suite nos derniers résultats.

3 Résultats

oMAP est disponible à : `http://www.cwi.nl/~troncy/oMAP`.
Il doit être utilisé en lançant la commande :

```
java -jar omap.jar -i %method% -r %renderer%
     -o %resultFile% %sourceOnto% %targetOnto%
```

où :

[1] Par exemple en utilisant l'algorithme de Porter.
[2] WordNet® `http://wordnet.princeton.edu/`.
[3] `http://oaei.inrialpes.fr`

- *method* est : `it.cnr.isti.OMapAlignment` ;
- *renderer* is : `fr.inrialpes.exmo.align.impl.renderer.RDFRendererVisitor2` ;
- *resultFile* est le nom du fichier résultat ;
- *sourceOnto* et *targetOnto* sont les URIs absolus des ontologies source et cible à aligner.

Les résultats du test *directory* sont donnés dans le tableau 1 :

oMAP	**OLA**	**Falcon**	**Dublin20**	**CMS**	**FOAM**	**ctxMatch**
34.43%	31.96%	31.17%	26.53%	14.08%	11.88%	9.36%

TAB. 1 – *Résultats pour le test* directory*, classés selon les performances des participants, données 2005*

Les résultats du test *benchmark* sont donnés dans le tableau 2 :

#	Name	Prec.	Rec.	Time
101	Reference alignment	0.96	1.00	20ms
102	Irrelevant ontology	0.00	NaN	
103	Language generalization	0.96	1.00	20ms
104	Language restriction	0.96	1.00	20ms
201	No names	0.88	0.38	20ms
202	No names, no comments	0.85	0.24	20ms
203	No comments	0.96	1.00	
204	Naming conventions	0.95	0.89	40ms
205	Synonyms	0.81	0.63	40ms
206	Translation	0.89	0.49	40ms
207		0.89	0.49	40ms
208		0.96	0.90	30ms
209		0.73	0.54	40ms
210		0.90	0.39	40ms
221	No specialisation	0.96	1.00	20ms
222	Flatenned hierarchy	0.96	1.00	20ms
223	Expanded hierarchy	0.96	1.00	20ms
224	No instance	0.96	1.00	20ms
225	No restrictions	0.96	1.00	20ms
228	No properties	0.92	1.00	20ms
230	Flattened classes	0.91	1.00	20ms
231	Expanded classes	0.96	1.00	20ms
232		0.96	1.00	20ms
233		0.92	1.00	20ms
236		0.92	1.00	20ms
237		0.95	1.00	20ms
238		0.96	1.00	20ms
239		0.85	1.00	20ms
240		0.87	1.00	20ms
241		0.92	1.00	20ms
246		0.85	1.00	20ms
247		0.87	1.00	20ms
248		0.85	0.24	50ms
249		0.85	0.23	50ms
250		0.05	0.06	50ms
251		0.85	0.25	50ms
252		0.85	0.24	50ms
253		0.85	0.23	50ms
254		0.06	0.06	50ms
257		0.00	0.00	50ms
258		0.85	0.25	50ms
259		0.85	0.24	50ms
261		0.03	0.03	50ms
262		0.00	0.00	50ms
265		0.00	0.00	50ms
266		0.00	0.00	50ms
301	Real : BibTeX/MIT	0.94	0.25	40ms
302	Real : BibTeX/UMBC	1.00	0.58	40ms
303	Real : Karlsruhe	0.90	0.79	40ms
304	Real : INRIA	0.91	0.91	40ms

TAB. 2 – *Résultats détaillés de oMAP pour le test* benchmark

4 Conclusion

Au fur et à mesure que le web sémantique grossit, de plus en plus d'ontologies sont créées. Le développement d'outils automatiques pour aligner ces ontologies devient donc fondamental. Nous avons développé et décrit **oMAP**, un cadre formel pour aligner des ontologies. oMAP peut être vu comme une boite à outils utilisant différents classifieurs pour estimer la qualité des règles d'appariement. Nous avons implémenté et testé les différents classifieurs sur les données indépendantes fournis annuellement par les campagnes OAEI 2004 et 2005.

Comme travaux futurs, nous prévoyons d'introduire des classifieurs supplémentaires (utilisant d'autres ressources terminologiques ou d'autres mesures de distance tels que la distance KL) et de multiplier les tests en les combinant. Si l'ajout de nouveaux classifieurs dans oMAP est relativement trivial théoriquement, un problème subsiste lorsqu'il s'agit de trouver la combinaison optimale de toutes les prédictions obtenues. Nous avons proposé une liste de priorité entre les différents classifieurs, mais celle-ci est forcément dépendante du contenu des ontologies (leur degré de formalité, leur complexité, etc.). Par conséquent, nous planifions de multiplier les variantes et les tests afin d'améliorer globalement les performances d'oMAP.

Remerciements

A Umberto Straccia pour nos nombreuses discussions et à ses commentaires constructifs.

Références

Doan, A., J. Madhavan, R. Dhamankar, P. Domingos, et A. Halevy (2003). Learning to Match Ontologies on the Semantic Web. *The VLDB Journal 12*(4), 303–319.

Euzenat, J. (2004). An API for ontology alignment. In *3^{rd} International Semantic Web Conference (ISWC'04)*, Hiroshima, Japon, pp. 698–712.

Shvaiko, P. et J. Euzenat (2005). A Survey of Schema-based Matching Approaches. *Journal on Data Semantics (JoDS) 4*, 146–171.

Straccia, U. et R. Troncy (2005a). oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies. In *6^{th} International Conference on Web Information Systems Engineering (WISE'05)*, New York City, New York, USA, pp. 133–147.

Straccia, U. et R. Troncy (2005b). oMAP: Results of the Ontology Alignment Contest. In *Workshop on Integrating Ontologies*, Banff, Canada, pp. 92–96.

Straccia, U. et R. Troncy (2006). Towards Distributed Information Retrieval in the Semantic Web: Query Reformulation Using the oMAP Framework. In *3^{rd} European Semantic Web Conference (ESWC'06)*, Budva, Montenegro, pp. 378–392.

Summary

This paper presents a method and a tool for automatically aligning OWL ontologies, a crucial step for achieving the interoperability of heterogeneous systems in the Semantic Web.

Different components are combined for finding suitable mapping candidates (together with their weights), and the set of rules with maximum matching probability is selected. Various classifiers, terminological, machine learning-based and a classifier using the structure and the semantics of the OWL ontologies are proposed in oMAP. Our method has been implemented and we provide some results of an evaluation from the 2005 Campaign tests of the Ontology Alignment Evaluation Initiative.