Classification conceptuelle et ontologie de domaine pour l'intégration sémantique des données

Ismail Jellouli, Département de Mathématiques et d'Informatique, Faculté des Sciences Dhar Mehraz BP 1796 Fès, Maroc, ismail.jellouli@yahoo.fr
Mohammed El Mohajir, Département de Mathématiques et d'Informatique, Faculté des Sciences Dhar Mehraz BP 1796, Fès, Maroc, m.elmohajir@belgacom.net
Esteban Zimanyi, Service ingénierie de l'informatique et de la décision ULB Bruxelles, Belgique, ezimanyi@ulb.ac.be

Résumé :
La multiplication des sources de données, souvent hétérogènes, au niveau du Web a suscité un regain d'intérêt pour l'intégration des données. Notre travail s'inscrit dans ce contexte. Nous proposons une solution pour rechercher les sources pertinentes au vu d'une requête utilisateur.
Cette solution peut être perçue aussi bien comme un composant interne d'un système de médiation que comme un système de recherche d'information spécialisé.
Les sources de données que nous considérons sont toutes du même domaine (le tourisme). Nous les avons annotées à l'aide de termes issus d'une ontologie de ce domaine. Nous avons proposé pour la recherche un algorithme basé sur les treillis de Galois. Cet algorithme permet une classification formelle des sources de données et les résultats qu'il retourne sont classés par ordre de pertinence.
Pour valider notre solution, nous l'avons implémentée à l'aide du langage Java pour un cas d'étude d'une trentaine de sources de données et d'une vingtaine de termes de l'ontologie.

Mots-clés :

Ontologie, Annotation, Classification, Treillis de Galois, Recherche d'information, Affinement des requêtes

Abstract:
The multiplicity of data sources, often heterogeneous, like the ones used for the web renews interest in data integration. In this context, we present a solution that searches multiple data sources and extracts relevant information as a response to a client query. Our solution can be either seen as an internal component of a mediation system or as a stand-alone information retrieval system.
The data sources we have considered are all related to the same domain; which is "Tourism". We annotated them according to this domain ontology. The search method we adopted used a clustering technique based on concept lattices. This algorithm allows formal classification of data sources and the results it returns are ordered by degree of pertinence.
To validate our solution, we implemented it as a Java application for a case study with about thirty data sources and twenty terms issued from the ontology.

Keywords:
Ontology, Annotation, Clustering, Concept lattices, Information retrieval, Query refinement

1 Introduction

L'utilisateur des systèmes d'information, tant au niveau du Web qu'au niveau de l'entreprise, se trouve face à une multiplicité de sources de données souvent hétérogènes, distribuées et autonomes. De ce fait, il est contraint d'assembler et de synthétiser les résultats retournés avant de pouvoir les exploiter. Dans ce contexte, l'utilisation des systèmes de médiation peut s'avérer une solution appropriée.

Un système de médiation classique a pour objectif de fournir un accès intégré aux données indépendamment de leur localisation mais tout en supportant leur hétérogénéité (Gardarin, 2002). Il se présente comme une couche intermédiaire entre l'application utilisateur et les sources de données (Wiederhold, 1992).

Avant d'interroger effectivement les sources de données, le médiateur doit déterminer celles qui, parmi elles, contiennent des informations pertinentes au vu de la requête de l'utilisateur. Cette fonction est assurée par un composant interne qui s'apparente beaucoup à un système de recherche d'information.

L'objectif de notre travail est de proposer une méthode pour l'élaboration de ce dernier composant en utilisant une technique de classification basée sur les treillis de Galois.

Au préalable nous associons des annotations à chacune des sources. Une annotation est un ensemble de termes issus de l'ontologie servant à décrire les sources de données. Nous traitons ce point dans la section 2.

La classification est basée sur les annotations. La section 3 présente notre solution pour la classification des sources à l'aide des treillis de Galois.

La section 4 explique l'algorithme utilisé pour la recherche des sources de données. L'idée est d'utiliser le treillis qui classe les sources de données pour rechercher les sources pertinentes pour une requête utilisateur. Si les résultats ne sont pas satisfaisants, l'utilisateur a la possibilité d'affiner sa requête (Section 5).

Dans la dernière section, nous donnons un aperçu du prototype que nous avons développé pour le test et la validation de notre solution.

Nous terminons par une conclusion dans laquelle nous donnons un bilan de ce travail et les perspectives qu'il permet d'entrevoir.

2 Annotation des sources de données

L'annotation est une phase importante du processus de recherche d'information. Il s'agit d'associer aux sources de données des méta-données décrivant leurs contenus respectifs. Ces méta-données vont servir à la classification et à la recherche des sources de données pour répondre aux requêtes des utilisateurs.

Dans notre cas, les méta-données sont conformes à un vocabulaire issu de l'ontologie de domaine.

2.1 Le domaine d'application

Nous avons choisi le « tourisme » comme domaine d'application de notre solution. C'est un domaine qui se caractérise par une multitude de sources de données réparties (notamment sur le Web). Dans ce cas, un système de médiation pourrait avoir comme objectif de fournir à l'utilisateur final une réponse intégrée construite à partir des offres existantes.

2.2 L'ontologie « tourisme »

Nous avons utilisé un prototype d'ontologie qui comporte uniquement la partie terminologique (pas de partie déductive). Celle-ci se présente sous forme de trois arborescences : les destinations, les types d'hébergement et les activités (Cf. figure 1).

Figure 1 : Aperçu de l'ontologie « tourisme » comme elle apparaît sur l'éditeur *protégé*

2.3 Les sources de données

Nous nous inscrivons dans un contexte ouvert (e.g. le Web), c'est pourquoi nous supposons que les sources de données sont multiples, distribuées, hétérogènes et autonomes. Elles peuvent être structurées (relationnel, objet,…), semi-structurées (XML, HTML,…) ou plates (texte).

Nous avons privilégié pour notre prototype les données de type XML. Ce modèle a été retenu parce qu'il est de plus en plus utilisé comme modèle pivot dans les systèmes de médiation.

2.4 Le processus d'annotation

Le processus d'annotation se base sur les schémas des sources de données structurées, sur les tags pour les fichiers XML et sur le contenu pour les sources non structurées.
L'annotation consiste en l'attribution de termes spécifiques de l'ontologie aux sources de données.
Au préalable, une liste des termes-candidats susceptibles de servir pour l'annotation est extraite de l'ontologie. Il s'agit ensuite d'associer un sous-ensemble des termes de cette liste à chacune des sources de données.
Cette opération peut être faite de manière automatique ; nous l'avons implémentée pour les données de type XML, ou manuelle quand c'est l'administrateur qui choisit manuellement les termes de l'annotation.
Un exemple de source de données et de l'annotation correspondante est montré sur la figure 2.

3 La classification des sources de données

Comme nous l'avons proposé en introduction, nous allons utiliser la classification formelle à l'aide des treillis de Galois comme base de notre solution. Nous allons, dans ce qui suit, présenter les treillis de Galois puis leur application concrète à notre cas d'étude.

3.1 Les treillis de Galois

Soit deux ensembles **E** (ensemble des individus) et **E'** (ensemble des propriétés) et une relation binaire **R** entre ces deux ensembles. Le treillis de Galois correspondant au triplet **(E,E',R)** est un ensemble de classes dans lequel chaque classe est un couple **(X,X')** avec **X** un sous-ensemble de **E** et **X'** un sous-ensemble de **E'**. **X** et **X'** sont appelés respectivement *extension* et *intension* du couple **(X,X')**. Le couple **(X,X')** est également appelé *concept formel*. **(E,E',R)** est appelé *contexte formel*.

Exemple de source de données

```
<Destination>
        <Code>D001</Code>
<Nom>Aktaraz</Nom>
<Activité>
<sports>
<Tennis>
        4 terrains terre battue
</Tennis>
<Natation>
        Piscine, maître nageur
</Natation>
</sports>
<Visites>
<Musée>
        Musée de l'art moderne
</Musée>
<Monument>
        Vestiges Aramites
</Monument>
</Visites>
</Activité>
<Hébergement>
<Hotel>
<HotelEco>Hotel 2 étoiles </HotelEco>
<HotelLuxe>Hotel lumières 5 étoiles
</HotelLuxe>
</Hotel>
</Hébergement>
<Environnement>
<MilieuUrbain>
<Ville>Samadan</Ville>
</MilieuUrbain>
</Environnement>
</Destination>
```

Exemple d'annotation pour cette source : Ville, HotelLuxe, HotelEco, Tennis, Musée, Monument.

Figure 2 : Exemple de source de données et de l'annotation associée

Chaque couple **(X,X')** est complet. Cela signifie que **X'** contient seulement les propriétés partagées par tous les individus appartenant à **X** et symétriquement, les éléments de **X** contiennent les individus partageant toutes les propriétés appartenant à **X'**.

Il existe une relation d'ordre partiel « **<** » entre les concepts formels **(X,X')**. Soit $\mathbf{C_1 = (X_1,X_1')}$ et $\mathbf{C_2 = (X2,X2')}$, on a $\mathbf{C_1<C_2} \Leftrightarrow \mathbf{X_1} \subseteq \mathbf{X_2}$ et $\mathbf{C_1<C_2} \Leftrightarrow \mathbf{X_2'} \subseteq \mathbf{X_1'}$. Cette relation est aussi appelée

relation de *subsomption*. Cette relation d'ordre partiel permet de représenter le treillis de Galois sous forme de graphe dont les nœuds sont les concepts formels. Un arc entre deux concepts $\mathbf{C_1}$ et $\mathbf{C_2}$ signifie que $\mathbf{C_2}$ est un subsumant directe de $\mathbf{C_1}$ càd $\mathbf{C_1<C_2}$ et il n'existe aucun concept **C'** du treillis tel que $\mathbf{C_1<C'}$ et $\mathbf{C'<C_2}$.

3.2 Contexte formel : « Tourisme »

Dans notre étude, les objets sont des sources de données et les attributs sont un sous-ensemble des termes de l'ontologie « Tourisme ».

Nous considérons que nous disposons de 5 sources de données soit $\mathbf{E} = \{S_1,S_2,S_3,S_4,S_5\}$ et nous restreignons les termes candidats à 9 soit (en abréviations):

E' = {*VL*,*CN*,*HO*,*CG*,*PG*,*MT*,*YG*,*NT*,*TN*}

avec : ***VL*** pour ville, ***CN*** pour campagne, ***HO*** pour hôtel, ***CG*** pour camping, ***PG*** pour plage, ***MT*** pour montagne, ***YG*** pour yoga, ***NT*** pour natation, ***TN*** pour tennis.

La relation **R** est représentée sous forme matricielle dans le tableau 1 ci-dessous :

R	VL	CN	HO	CG	PG	MT	YG	NT	TN
S_1	1	0	1	0	0	1	0	1	0
S_2	1	0	1	0	0	0	1	0	1
S_3	1	0	0	1	0	0	1	0	1
S_4	0	1	1	0	0	1	0	1	0
S_5	0	1	0	0	1	0	1	0	0

Tableau 1 : Représentation matricielle de la relation R[1]

3.2 Treillis de Galois : « Tourisme »

Le treillis de Galois correspondant au contexte formel du tableau 1 est présenté dans la figure 3 ci-dessous :

[1] La valeur 1 de la ligne « 2 » et de la colonne « *HO* » signifie que la source 1 possède la propriété « *HO* », ou, en termes plus significatifs, que la destination « 1 » dispose d'un hôtel.

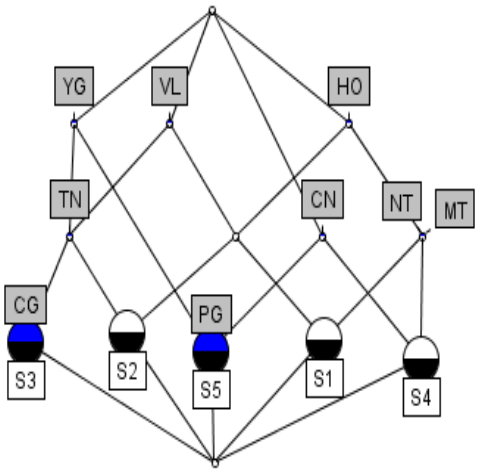

Figure 3 : Treillis[1] de Galois correspondant à la relation R

Le treillis de Galois établit une classification des sources de données. Chaque concept formel du treillis représente en fait une classe. Par exemple, le concept $(\{S_2,S_3\},\{VL,YG,TN\})$[2] met dans la même classe les sources de données **2** et **3**. Ces deux sources sont dans cette classe parce qu'elles sont les seules à partager les propriétés ***VL***, ***YG*** et ***TN*** i.e., que ce sont des destinations situées en ville et qu'on peut y pratiquer le tennis ou se détendre durant des séances de Yoga. Le treillis établit aussi une hiérarchie entre les classes. On peut y lire que le concept formel $(\{S_2,S_3\},\{VL,YG,TN\})$ est plus particulier que $(\{S_1,S_2,S_3\},\{VL\})$[3] dans le sens où il possède plus de propriétés c'est ce qui se traduit par le fait que $\{S_2,S_3\} \subset \{S_1,S_2,S_3\}$. Nous constatons que cette hiérarchie est à double sens, i.e., le treillis est une sorte d' «arborescence » à deux « racines » : $(\{S_1,S_2,S_3,S_4,S_5\},\phi)$ et $(\phi,\{VL,CN,HO,CG,PG,T,YG,NT,TN\})$ que nous appellerons respectivement ***top*** et ***bottom***.

Le déplacement vers ***top*** correspond à la généralisation et celui vers ***bottom*** à la spécialisation**.**

Le treillis est construit à l'aide de l'algorithme de construction incrémentale de Godin (Godin, Missaoui et al., 1995).

---

[1] Le treillis de la figure est un treillis d'héritage càd qu'un nœud hérite les propriétés des nœuds qui le subsument et les individus des nœuds qui lui sont subsumés. Par exemple le nœud étiqueté ***TN*** a comme extension $\{S_2,S_3\}$ et comme intension {***VL*** , ***YG***, ***TN***}

[2] Le nœud étiqueté par ***TN*** dans la figure 3

[3] Le nœud étiqueté par ***VL*** dans la figure 3

4 Recherches des sources de données

4.1 Requête utilisateur et treillis de Galois

Notre idée est de considérer la requête utilisateur comme une nouvelle source dont les propriétés sont les termes de la requête. Cette « source » va être ajoutée au treillis de manière incrémentale (Godin, Missaoui et al., 1995.) Cet ajout transformera le treillis ; de nouveaux nœuds vont être ajoutés et d'autres seront modifiés.

Exemple de requête

Supposons que l'utilisateur formule la requête suivante : ***Campagne***, ***Hôtel***, ***Montagne***, ***Tennis***. Il est à la recherche de destinations en campagne situées en montagne et où il peut loger dans un hôtel et pratiquer son sport favori le « tennis ».

Cette requête va être représentée comme suit: ({***Q***},{***CN***,***HO***,***MT***,***TN***}). Elle peut s'apparenter à une nouvelle source ***Q*** qui a les propriétés : ***CN***, ***HO***, ***MT***, ***TN***.

Le treillis, après ajout de la requête, va se transformer comme ci-dessous (figure 4):

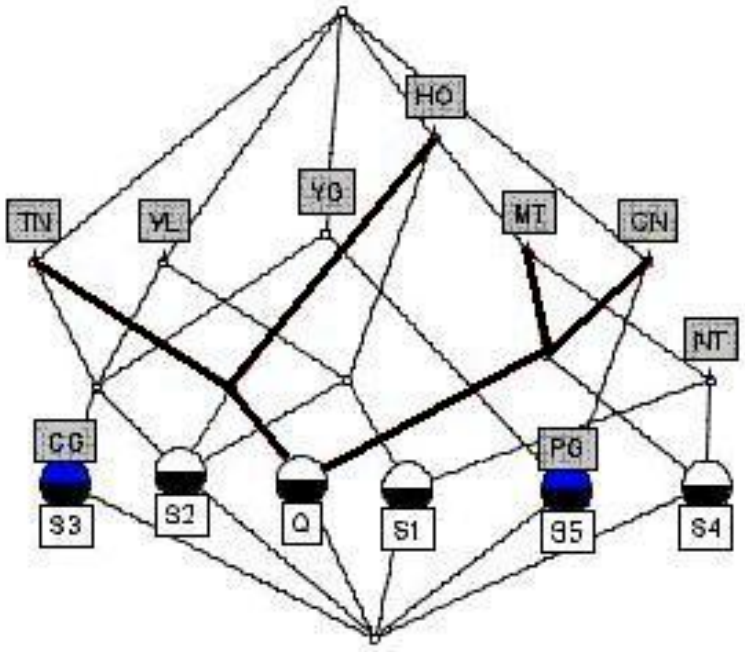

Figure 4 : Treillis de Galois après insertion de la requête

4.2 Algorithme de recherche

A présent, il s'agit de voir comment répondre à la requête en utilisant ce nouveau treillis. Il faut d'abord remarquer que dans tous les cas, il existe un nœud qui contient exactement les termes de la requête dans sa partie intension. Ceci provient des propriétés du treillis de Galois. En effet, il y a une nouvelle instance qui n'existait pas dans le treillis (la « source » Q dans notre exemple). Si la requête de l'utilisateur, i.e., l'intension n'est contenue dans aucune paire extension/intension (concept formel) alors la nouvelle « source » est la seule à correspondre à cette intension et il y aura donc forcément un nouveau nœud constitué de la paire : (Numéro source, termes de la requête). Dans la cas contraire, i.e., il existe déjà une paire qui a comme intension les termes de la requête alors cette paire va être modifiée par l'adjonction de la nouvelle source dans sa partie « extension ». Dans tous les cas, on va se retrouver avec un nœud qui a comme intension exactement les termes de la requête. Ce nœud est évidemment unique du fait des propriétés du treillis. Ce dernier, étant complet, il ne peut y avoir de nœud avec la même intension et avec une extension différente.

Cette constatation est fondamentale car elle est à la base de l'algorithme de recherche. En effet, c'est ce nœud unique dont nous venons de parler qui va servir de point de départ pour l'exploration du treillis à la recherche des sources pertinentes. C'est d'ailleurs le nœud qui contient les sources les plus pertinentes. Le parcours du treillis va continuer de manière ascendante jusqu'au sommet (***top***). A chaque fois, il y aura dans la partie extension des nœuds visités de nouvelles sources pertinentes dans le sens où ils partagent des propriétés avec la requête. Le nombre de ces propriétés va diminuant, en remontant le treillis et les sources rencontrées sont de moins en moins intéressantes.

En résumé, le treillis va nous fournir la liste des sources de données par ordre décroissant de pertinence.

Nous donnons ci-dessous, une présentation plus formelle de cet algorithme :

Recherche (Requête, Treillis) {

1. Insérer dans le treillis la requête considérée comme source de donnée fictive.
2. Parcourir le treillis à partir de ***bottom*** jusqu'à trouver le nœud qui a comme intension les termes de la requête.
3. Ajouter les sources de données retrouvées dans l'extension de ce nœud dans une liste FIFO.
4. Remonter dans le treillis à partir de ce noeud; ajouter à chaque fois les sources de données non encore rencontrées dans la liste FIFO.
5. Cette liste va contenir les sources de données par ordre de leur insertion et qui correspond en même temps à leur ordre de pertinence.

}

Illustration :

Dans notre exemple[1] (figure 4), la requête utilisateur va générer les réponses suivantes :

Au premier niveau du nœud (**{*Q*},{*CN*,*HO*,*MT*,*TN*}**), il n'y a aucune réponse ; la partie extension ne comportant aucune source réelle.

Au second niveau, les réponses fournies seront les sources S_4 et S_2. La source S_4 a en commun avec la requête les propriétés ***CN***, ***HO*** et ***MT***, i.e., que cette destination est dans une campagne située en montagne, qu'elle offre un hôtel mais qu'elle n'offre pas la possibilité de pratiquer le tennis. La source S_2 est située en campagne et dispose d'un terrain de tennis.

Au troisième niveau, on retrouve les sources S_1, S_3 et S_5. La destination **1** offre un hôtel et la possibilité de jouer au tennis, la destination **3** offre un terrain de tennis et la destination **5** est située en campagne.

On remarque, qu'en effet, les sources retournées sont toutes pertinentes dans le sens où elles possèdent au moins une propriété souhaitée par l'utilisateur et qu'elles ont été fournies en ordre décroissant de pertinence.

[1] Les arcs en **gras** du graphe de la figure 4 représentent les étapes de la recherche

5 Affinement des requêtes

Une amélioration possible de cet algorithme concerne l'affinement des requêtes. Il se peut, en effet, que la requête retourne peu ou pas de résultats.

L'affinement se traduit souvent par l'enrichissement sémantique de la requête (Bidault, Froideveaux et al., 2002) ; il s'agit de transformer la requête en remplaçant ses termes par des termes plus généraux ou plus spécifiques.

C'est ce principe que nous avons appliqué ici. La partie taxonomique de l'ontologie est organisée sous forme de hiérarchie et l'enrichissement va consister en l'ajout des termes liés aux termes de la requête par des liens de généralisation/spécialisation. L'idée est d'ajouter les termes subsumants i.e., les ancêtres dans l'arborescence dans le cas de la généralisation et les termes subsumées i.e., les descendants dans l'arborescence dans le cas de la spécialisation.

Dans le cas de la généralisation, si l'utilisateur exprime une requête contenant le terme « hôtel de luxe », par exemple, celle-ci va être enrichie par l'ajout du terme « Hôtel » et va retourner par conséquent toutes les destinations annotées par ce terme. A l'opposé, si l'utilisateur introduit pour la formulation de sa requête le terme « hôtel », sa requête va être enrichie par spécialisation par les termes « hôtel de luxe » et « hôtel économique ».

6 Implémentation

Pour tester la validité de notre solution, nous l'avons implémentée à l'aide du langage Java. Notre prototype a été testé sur un jeu d'essai comprenant une trentaine de sources de données. Nous avons retenu, par ailleurs, une vingtaine de termes-candidats (Cf. 2.4) pour l'annotation.

La première partie de notre programme porte sur l'annotation des sources de données. Il a fallu, dans un premier temps, choisir une représentation interne des objets manipulés à savoir, l'ontologie et les sources de données. Nous avons adopté, pour le faire, des structures à base de HashSet[1].
Comme nous n'utilisons que la partie taxonomique de l'ontologie, celle-ci est représentée simplement par un ensemble de termes-candidats retenus pour l'annotation et stockés dans un HashSet. Ces termes alimentent une structure arborescente qui va servir pour l'affinement des requêtes. Les sources de données quant à elles sont stockées dans une structure dédiée persistante. Elle est basée, elle aussi, sur les HashSet et contient pour chaque source les métadonnées

[1] Le HashSet est uns classe Java qui permet de stocker des objets sans les dupliquer, son intérêt réside dans la possibilité d'accéder directement à ces objets.

correspondantes. Cette structure est alimentée au fur et à mesure de l'exploitation du programme lors de l'ajout de nouvelles sources.

Une autre structure représente le treillis de Galois dans lequel les sources de données sont classifiées. En interne, le treillis est un graphe dont les nœuds représentent les classes et les arcs représentent les relations de subsomption (généralisation/spécialisation) qui lient les classes.

Toutes ces structures (ontologie, sources de données et treillis) sont sérialisées[1].

L'utilisateur a la possibilité d'ajouter des sources de données. L'ajout peut être automatique ou manuel.

Dans le cas de l'ajout automatique, le processus se déroule ainsi :

- L'utilisateur choisit un fichier[2] représentant une source de données ;
- Le programme extrait les tags qui vont servir comme propriétés de la source (Cf. 2.4) ;
- L'objet sérialisé « source de données » est mis à jour (Ajout de la nouvelle source) ;
- L'objet sérialisé « treillis » est mis à jour par insertion incrémentale (Godin, Missaoui et al., 1995) de la source ;

Dans le cas de l'ajout manuel, c'est l'utilisateur qui annote la source ajoutée et le programme exécute les autres étapes du processus expliqué ci-dessus (Mise à jour du treillis notamment)

Le programme implémente, par ailleurs, la fonction de recherche. Pour ce faire, l'utilisateur doit formuler une requête constituée d'un ensemble de termes de préférence appartenant à l'ontologie « tourisme ». C'est pour cela que le programme offre une interface permettant de naviguer dans l'arborescence de l'ontologie et de choisir ainsi les termes appropriés (figure 5). Une fois la requête soumise, le programme retourne le résultat qui est l'ensemble des sources classées par ordre de pertinence (figure 6). Il implémente à cet effet l'algorithme de la section 5.

Enfin, l'utilisateur a la possibilité d'affiner sa requête par généralisation ou par spécialisation au cas où les résultats retournés ne sont pas satisfaisants.

[1] Stockées de manière persistante
[2] Dans notre prototype, il s'agit d'un document XML

Figure 5 : Interface requête utilisateur

Figure 6 : Résultats de la requête

7 Conclusion et Perspectives

Dans ce travail nous avons réalisé un composant « recherche d'information » destiné à un système de médiation. Pour ce faire, nous avons fait appel à une technique de classification formelle basée sur les treillis de Galois. Notre prototype nous a permis de vérifier la validité de cette solution et la pertinence des résultats qu'il retourne.

Les extensions possibles pour ce travail se situent à divers niveaux. Il s'agit, entre autres, de l'implémentation, de la technique de classification et de l'annotation.

Au niveau de l'implémentation, il serait intéressant d'étendre notre prototype au Web, i.e., permettre aux utilisateurs d'accéder à l'application via le navigateur. Cela peut être réalisé grâce à la technologie J2EE notamment.

En ce qui concerne la classification, il va falloir tester notre solution dans un contexte plus réaliste où le nombre de sources à traiter est important (plusieurs centaines voire plusieurs milliers de sources). Il est à noter que des améliorations ont été apportées à l'algorithme de construction incrémentale des treillis de Galois que nous avons implémenté pour tenir compte de cette contrainte (Godin, Missaoui et al., 1995.) Par ailleurs, on peut explorer les nouveaux algorithmes de construction des treillis comme (Van der Merwe et Obiedkov, 2004) et des propositions pour combiner les techniques de classification statistiques avec les treillis de Galois (Valtchev et Missaoui, 2000).

S'agissant de l'annotation des données, une première amélioration serait la prise en compte, pour les sources de données, des formats de données dominantes du Web (HTML et PDF). On peut, par ailleurs, améliorer le processus d'annotation du prototype en s'inspirant des techniques utilisées par la recherche d'information (Manning, Raghavan et al., 2008) et des recherches sur les correspondances des schémas de données (Noy, 2004)(Rahm et Bernstein, 2001)(Shvaiko et Euzenat, 2005).

Références

Bidault, A., Froidevaux, C., Gagliardi, H., Goasdoué, F., Reynuad, C., Rousset, M.C. et al. (2002). Construction de médiateurs pour intégrer des sources d'informations multiples et hétérogènes : le projet PICSEL. *Revue I3 : Information-Intercation-Intelligence,* 9-59.

Gardarin, G. (2002). *XML des bases de données aux services Web*. Paris : Dunod.

Godin, R., Missaoui, R., Alaoui, H. (1995). Incremental concept formation algorithms based on Galois (concept) lattices, *Computational Intelligence*, 11(2), 246-267.

Manning, C.D., Raghavan, P., Shütze, H. (to appear 2008): *Introduction to Information Retrieval*, Cambridge University Press.

Noy, N. Semantic Integration (2004): A Survey on ontology-based approaches. *SIGMOD Record, Special Issue on Semantic Integration,* 33(4), December 2004, 65-70

Rahm, E., Bernstein, P. (2001): A survey of approaches to automatic schema matching approaches, *VLDB Journal*, 10, 334-350

Shvaiko, P., Euzenat, J. (2005): A survey of schema-based matching approaches. *Journal on Data semantics IV*, 146-171

Valtchev, P., Missaoui, R., (2000): Similarity-based clustering versus Galois lattice building: Strengths and weaknesses, *European Conference on Object-Oriented Programming*, Sophia-Antipolis: June 2000

Van der Merwe, D., Obiedkov, A., Kourie, D. G. AddIntent (2004): A New Incremental Algorithm for Constructing Concept Lattices, *Proceedings of ICFCA 2004*, Sydney, Australia: February 23-26, 372-385

Wiederhold, G. (1992), Mediators in the architecture of future information systems. *Computer*, 25, *38-49*.